Diploma Thesis

Informatics

3D Particle Tracking Velocimetry using a Stereo Camera Setup

Technical Aspects of Multimodal Systems,
University of Hamburg

by

Till Bosselmann

and

Nils Erik Flick

December 2009

supervised by

Prof. Dr. Jianwei Zhang

Dr. Peer Stelldinger

Kurzfassung

Diese Arbeit beschreibt einen Ansatz zur Luftströmungsmessung über die Verfolgung kleiner Partikel mit Hilfe eines Stereokamerasystems. Von einem bestehenden Hardwareaufbau ausgehend, welcher bereits für 2D-Geschwindigkeitsmessung eingesetzt wurde, haben wir in Zusammenarbeit mit Airbus Deutschland GmbH ein neues Messsystem entwickelt. Dieses soll die Partikel im Raum lokalisieren sowie ihre Bewegung interaktiv visualisieren.

Ein konkretes Anwendungsgebiet ergibt sich in der Optimierung der Belüftung von Passagierkabinen in Flugzeugen über eine verbesserte Form und Positionierung der Lufteinlässe.

Abstract

In this thesis, we describe an approach for measuring fluid flows by means of particle tracking velocimetry with a stereo camera system. Building upon an existing hardware setup, originally used for 2D velocimetry, we developed a new system for depth recovery and interactive visualization of particle trajectories, in cooperation with our industry partner Airbus Deutschland GmbH.

The application is meant to assist in the optimization of ventilation in aircraft passenger cabins through optimal positioning and shaping of airflow inlets. Our system is to be used to visually evaluate inlet configurations (with a fully automatic evaluation as a possibility for the future).

“[. . .] the sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work – that is, correctly to describe phenomena from a reasonably wide area. [. . .]”

– John von Neumann [vN55]

Contents

Chapter 1

Introduction

Draught and insufficient air supply through lack of air circulation is one of the main sources of discomfort for aircraft passengers and crew, making the optimal design of the ventilation system a consideration of great importance in the design and configuration of aircraft cabins. Our thesis is motivated by the desire to find a flexible system for measuring and visualizing such currents, for the benefit of the cabin engineer.

For that purpose, we developed an optical system, which allows us to effect these measurements by tracking small tracer particles introduced via the ventilation system. This system is more flexible in the deployment and requires less hardware components than comparable optical and non-optical velocimetry measurement systems.

Summary of the system

Building upon an existing hardware setup, originally used for 2D velocimetry, we developed a new system capable of depth recovery.

The system captures the movement of tracer particles using a stereo camera setup. On the images, particle movement is visible as traces. These traces are segmented and the inter-frame location of the particles is extracted, using image processing. This is done separately for each camera, leading to two sets of 2D particle candidates, which are well localized in time.

From the two sets of particle candidates, all possible correspondences are considered and their hypothetical location in space is reconstructed. This is done for a number of successive frames, and a graph is built out of all possible continuations.

Paths in the graph are then converted into curves and filtered based on how closely they match evidence on the images. In order to keep the number of nodes in the graph manageable, a window with the size of a few frames can be used.

Cooperation and development process

While our velocimetry system is not limited to being used specifically in aircraft cabins, its development was motivated by this application. We developed a prototype for Airbus Deutschland GmbH, whose facilities we were able to use to conduct our experiments.

The requirements of the to be designed system were developed in cooperation with the engineers at Airbus, whom the system is intended to support in their work.

1.1 Prior work

Measuring velocity with the help of tracer particles is not a novel idea. Rather, it is a widely researched field with a long and extensive publication history. In the following, we will describe the categories in which most velocimetry methods fall, along with a few methods which are similar to the system we designed.

1.1.1 Particle velocimetry measurement

There are three general categories of methods which measure particle velocity. In the first category, Particle Image Velocimetry (PIV), a laser is commonly used to intersect the fluid in which the particles move. As particles pass through the plane, which is illuminated by the laser, two images are taken within a short period of time. On these images, cross correlation of image regions is performed, in order to obtain the direction and velocity of the flow. This approach is limited by the area which can be illuminated by the laser, as well as the acquisition and transfer rate of the cameras. While there are stereoscopic PIV setups ([Pra00]), it is more often used to obtain depthless flow information. As individual particles do not have to be matched, this approach is suitable for dealing with large numbers of particles.

In contrast, Particle Tracking Velocimetry (PTV) methods identify single particles, and track them from frame to frame. This makes PTV more suitable for experiments with a low number of particles. The lower number of particles allows depth reconstruction for each particle, provided correspondence can be established. Our system falls into this category.

The principle idea behind Particle Streak Tracking (PST) is to control scene illumination through a shutter system. It generates a train of pulses, during which the camera is exposed to the reflections of the tracer particles in the scene. As in PIT, only a single plane is illuminated, and the pulses are in synchrony with the exposure time of the camera, so that exactly one pair of pulses falls into each capture frame. In each pair, one pulse is longer than the other. This separates the image of each particle trajectory into two components with different length. Using this knowledge, it is possible to determine the (2D) direction of each particle. See [MMR01] for an application of PST for 3D velocity measurement of airflows on a single plane.

Aside from PIV, PTV, and PST, there are other velocimetry measurement methods, such as using arrays of hot-wire anemometers, or laser Doppler velocimetry. Hot wire anemometers work by measuring the change of conductivity of a wire, as wind cools it.

Laser Doppler velocimetry exploits the Doppler effect to measure the velocity of particles. A laser beam illuminates a volume. There is a change in wavelength as particles cross the beam, in accordance of the Doppler effect.

1.1.2 Similar approaches

Particle velocimetry has been used before to investigate fluid flows in aircraft cabins. Bosbach et al. describe such an application in [BKW09], where 2D PIV is applied to visualize air flows generated by the climate control in a mockup of an Airbus cabin. Like in our case, the setup has to be non intrusive, and the volume being investigated is considerably larger than in most other particle velocimetry applications.

Guezennec et al. ([GBTK94]) describe a two camera system which is similar to our own in some ways. Particles are separated from the background, possible paths are constructed in 2D, which are then matched and 3D reconstruction is performed. An adaptive Gaussian window is used to eliminate erroneous vectors.

While the general steps match our approach, there are significant differences in the way each step is being conducted: We do not assume to have perfect control of the experimentation environment, so that we can expect to achieve a complete background removal. As Guezennec et al record with a high framerate, they can use a scaleable template to find overlaying particles. Our system was designed to be able to operate with low framerates, making it necessary to find another way of locating particles in the images. The key difference, however, is that 2D curves are matched in order to obtain the actual paths, whereas we directly construct 3D paths.

Summary

Using neutrally buoyant particles to investigate flow structures is a well established approach, for which a wide selection of methods exists. They are usually applied to small scale volumes, however, and rely on manipulating the test environment in order to simply the detection of particles. There is little research on large scale, non intrusive applications of PTV, using cameras with a low framerate.

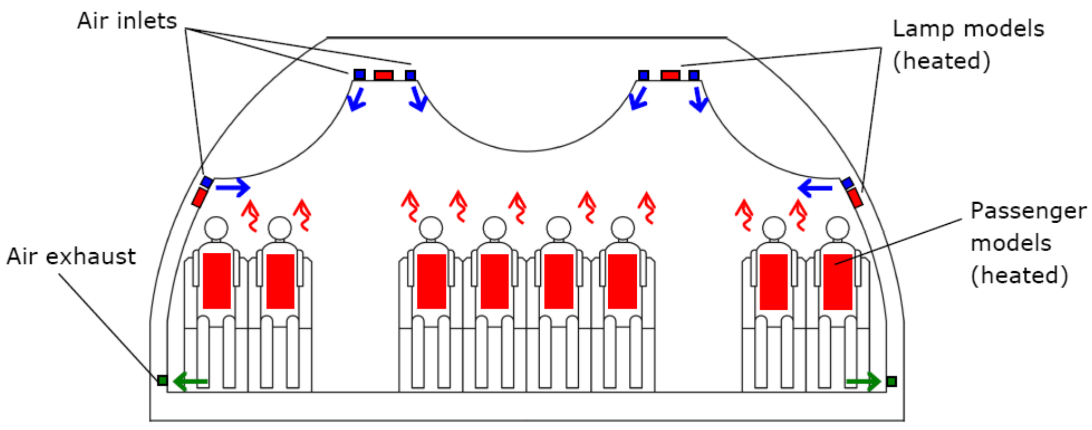


Figure 1.1: Cross section of a full scale aircraft cabin mockup, as it was used in our experiments. (Illustration from [BKW+06])

1.2 Experimental setup

In this section we will describe the hardware setup we used for conducting our experiments. It should be noted that our system is not restricted to this particular setup. Section 8.5 on page 184 discusses alternative arrangements.

As mentioned in the introduction, the purpose of this system is the measurement of airflow through airplane passenger cabins. For that purpose, we require a number of cameras, and a way to partially illuminate the cabin, so that the reflections of tracer particles may be captured. We did not conduct our experiments in real passenger cabins, but rather in a full scale mockup of one (figure 1.1).

1.2.1 Prior setup

This cabin mock up has been previously used for 2D PST. The setup consisted of two high resolution cameras, viewing a scene illuminated through a chain of halogen lamps, embedded into the floor of a mockup cabin. Tracer particles were generated with a soap bubble generator manufactured by sage action, inc.

According to the specifications, the soap generates uniformly sized bubbles, with an adjustable diametre ranging from 1.3mm to 3.8mm.The lifetime of bubbles ranges from one to two minutes and the generator produces between 300 and 400 bubbles per second.

Bubbles are generated by mixing helium with a soap lye and air. They are injected into the cabin mock up through the ventilation system. The bubbles are illuminated as they pass the conical illuminated area, and their reflection is captured by the two cameras. The

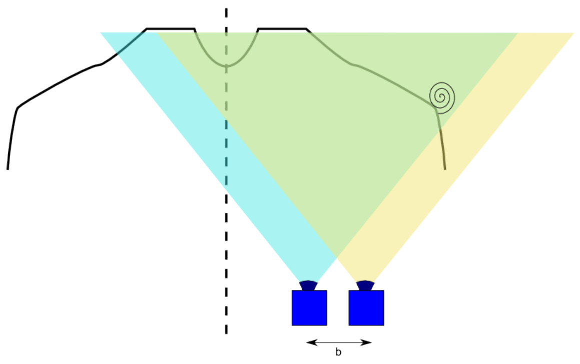


Figure 1.2: Our experimental setup. Two cameras observe one half of the mock up cabin.

cameras are spaced to cover the entire cabin with little overlap in their fields of view. Images are acquired through two separate ethernet connections with a computer running the image acquisition software provided by the cameras' manufacturer.

1.2.2 Our setup

We decided to reuse the hardware components from the existing setup for our system. While a stereo camera setup makes depth reconstruction more challenging than the use of three or four camera setups, as will be explained in chapter 3, the computationally expensive image segmentation and feature extraction has to be done for only two cameras. This makes using a stereo camera setup potentially faster and oligo camera approaches.

Figure 1.2 shows our changes to the previous experimental setup. We moved the cameras, so that only half of the cabin was covered in order to obtain overlapping fields of view.

The cameras operated with a frequency of about 7Hz. This frequency was chosen because higher frequencies led to loss of image frames. PTV with such a low sampling rate has, to our knowledge, not been explicitly addressed before. Rather, most PIV and PTV systems rely on high frequency sampling of image data. Lower temporal resolution means that there is less data to be examined. However, the temporal localization of particles on the images becomes increasingly difficult, as the sampling rate is reduced. For instance, cross correlation cannot be applied to measure the particle displacement, if the temporal distance is too large.

In order to synchronize the images from the two cameras, a hardware mechanism build into the cameras was used, connecting a strobe signal from one camera to trigger the other. Section 8.3, will offer some suggestions on how to deal with non-synchronized camera systems.

1.3 Outline of the thesis

This thesis begins by examining the underlying assumptions present in the assignment. We do so by anchoring the problem of measuring particle trajectories with a twin camera setup in its theoretical foundations in chapter 2. This entails the investigation of particle laden flows and measuring of light transport through digital optical sensors.

In the next chapter we take an geometric vantage point on the problem. The problem is thus reduced, and simplified. This has been a favourite approach in the stereo reconstruction community for a long time, as exemplified by the popular description of scene through edge and corner features.

Moving on to chapter 4, we devise a model which describes the scene in terms of geometry and discusses means of recovering it. The recovery is a multi-step process which starts with the removal of unwanted scene background. A preliminary segmentation tries to discriminate between regions of interest and background by classifying image regions. Contained within the regions of interest we search for certain points which are well defined, possess a known localization in space and time and correpond to specific events.

In chapter 4.5.1, we reconstruct particle trajectories through space and time. This is achieved by constructing curves from reconstructed space-time locations, and subsequently evaluating certain quality criteria.

Chapter 6 characterizes the implementation of the system in C++. After explaining important design decisions, we give an overview of the software architecture, comprising a detailed discussion of some main components.

The second-to-last chapter presents experimental results from all stages of the process.

In the last chapter, after discussing some possible future extensions, we conclude with a summary of the goals achieved in this work.

We assume that the reader has a firm mathematical-technical background and is familiar with real analysis, linear translation-invariant system theory and basic physics; prior knowledge about image processing, however, is not required.

At several points in the text, we will argue from *genericity*. For a precise definition, read [Lu76].

Chapter 2

Physical foundations

Before we begin introducing the methods devised for extracting relevant information from image series, and for the interpretation of that information, let us give some thought to the nature of the measurements.

Particle tracking velocimetry is a method based on indirect observations: the action of the fluid on small tracer particles is observed by integrating the light reflected from these particles.

This chapter is about the foundations of the particle tracking velocimetry setup; some facts are related and several conclusions are drawn from them; they are necessary to substantiate claims made about trajectories of particles and about image properties, in almost all of the following chapters.

It comprises the following parts:

- a condensed account of the usual mathematical-physical description of fluid flows, where we relate some results of interest and make the experimental setup plausible
- a short discussion of lighting, including considerations on how to ensure good contrast
- of measurements with digital cameras, since the object of study is measured by proxy of digitized photometric information.

As for the first part, there are several possibilities for following flows in optically transparent media with optical methods (see [Mer87]), most relying on suitably introducing another substance into the fluid.

The concepts of fluid dynamics are crucial for understanding the role of particles in the measurement of fluid flows, and also for the interpretation of the experiment: we must ascertain that the introduction of particles does not significantly alter the flow conditions being observed, and that the particles move with the fluid. We will conclude in the affirmative.

For the second part, despite its important role in the setup (for deciding how to light the scene), it proved out of the question to infer detailed scene information (or e.g. depth or timing) from the observed luminance distribution. In the first section of chapter 4, a model will be elaborated which allows the reduction of the scene to a simple geometrical description, although it is purely descriptive and does not yet show an obvious practical way of obtaining the description.

Finally, the chapter also contains a section on measurements with digital cameras and the noise they are subject to. Knowing the spectral characteristic of the noise is important for optimal feature detection.

2.1 Fluids

To justify the necessity of experiments, and their applicability, here is a short digest of fluid dynamics. Unfortunately, we cannot even scratch the surface of this vast topic here; however, we hope to give some reasons for the use of tracer particles. The paragraphs after the introduction of the Navier-Stokes equations, which concern themselves with particle-laden flows and measurements, contain some facts which can be gathered from the literature, as well as some conclusions we draw from them.

There is an abundance of literature on the subject. In the following, we refer to the treatments by Landau&Lifshitz ([LL59]), Lamb ([Lam32]), Batchelor ([Bat00]) and also Zeytounian ([Zey91]).

2.1.1 Introduction to fluid dynamics

Fluid dynamics, as a subdomain of fluid mechanics, is the domain of physics that deals with the motions of fluid media.

The term "fluid" describes certain states of matter. A fluid is a medium which is perfectly deformable (due to the mobility of the molecules), as opposed to a solid which is characterized by its rigid structure. Technically, a (Newtonian) fluid is sometimes defined as putting up no resistance to slowly applied shear stress, i.e. a force with no normal component with respect to a face of the material (cf. [Poz09], p.189, p.206)

Both being easily capable of flowing, liquids differ from gases in that they form a free surface (a gas will quickly fill any space available to it, because interactions between its molecules ideally are limited to elastic collisions). The study of fluid dynamics concerns itself with the common properties of both.

The movements of fluids are generally very complicated, and some aspects of fluid dynamics are still not very well understood, especially the phenomenon of turbulence, which also impedes accurate simulation.

The Navier-Stokes equations

Fluid flows are often taken to be governed by the Navier-Stokes equations: a system of nonlinear PDE which derive from the classical conservation laws when applied to a "continuum", i.e. it is valid when the particulate nature of matter can be, by and large, ignored [Bat00].

One obtains a vectorial equation describing the evolution of a velocity field $\vec{v}$, defined on a suitable subset of $\mathbb{R}^3$ – and to be taken with the appropriate boundary conditions (see [Zey91] p.75 and below).

$$\frac{\partial \vec{v}}{\partial t} = -(\vec{v} \cdot \nabla)\vec{v} - \frac{1}{\rho}\nabla p + \frac{\eta}{\rho}\Delta \vec{v}$$

ρ being the mass density, p the pressure. It is accompanied by the incompressibility equation, which is a good simplification of the actual behaviour at low velocities.

$$\nabla \cdot \vec{v} = 0$$

Let's quickly explain the meaning of the terms in above equations:

The whole equation is not only reminiscent of the Newtonian $\vec{F} = m\vec{a}$ (divided by the mass in the guise of ρ), but actually derived from its continuum mechanical formulation. The terms on the right hand side of the above equation are best explained via the corresponding forces:

- $-\frac{1}{\rho}\nabla p$ indicates that a pressure gradient exerts a force, inducing flow.
- $\frac{\eta}{\rho}\Delta \vec{v}$ is a dissipative term; it is the influence of viscosity, caused by interactions between molecules. $\nu = \frac{\eta}{\rho}$ is called the kinematic viscosity, as opposed to the dynamic viscosity η. It is a kind of diffusivity ($[m^2 s^{-1}]$). Leave it out to obtain the so-called Euler Equations.
- $-(\vec{v} \cdot \nabla)\vec{v}$ means that velocity is *advected.* It is caused by inertia, and responsible for the complicated behavior of the solutions.

In the following, we relay some relevant properties, most of the time strictly following the exposition in [LL59]:

An important consideration in the description of flows is *similarity*. How does the behavior change, for example, when the geometry of an obstacle is scaled? The laws remain unchanged, but some quantities will be affected.

There is a general way of expressing an equation relating n physical quantities expressible in terms of k fundamental units as a relation of only $n-k$ non-dimensional variables in the Buckingham Pi Theorem, which we do not state here (nor its elegant proof).

It suffices to say that there is a habitual choice of such non-dimensional variables in fluid dynamics, which is used together with the following non-dimensional scaling numbers.

Specifically, a steady ($\frac{d\vec{v}}{dt} = 0$) flow which depends on

- velocity $u[m \cdot s^{-1}]$
- kinematic viscosity $\nu[m^2 \cdot s^{-1}]$ and
- the scale $l[m]$ of the domain (always of the same shape)

is fully characterized by these quantities.

- One can combine them to one dimensionless number, named Reynolds number to honor Osborne Reynolds, who first used it: $Re = \frac{u \cdot l}{\nu}$
- When gravity is not neglected, another parameter is needed to describe the flow, the Froude number: $Fr = \frac{u^2}{lg}$
- When the flow is not steady but e.g. oscillatory, there is also a characteristic time interval τ (e.g. a period) and one more non-dimensional quantity can be defined, usually the Strouhal number: $St = \frac{u\tau}{l}$
- Finally, the Euler number takes care of the pressure. It involves a "characteristic" pressure difference a: $Eu = \frac{a}{\rho \cdot u^2}$

Flows which differ only in the scaling of measurement units are called *similar*.

To non-dimensionalize the equations, one converts velocities to $\vec{v} = \frac{\vec{v}}{u}$, the pressure to p and similarly the times $t = \frac{t}{\tau}$ and lengths (which also means replacing ∇ by $\nabla = l\nabla$) to express everything in terms of non-dimensional quantities and obtain nondimensionalized Navier-Stokes equations, something like:

$$St\frac{\partial \vec{v}}{\partial t} = -(\vec{v} \cdot \nabla)\vec{v} - Eu\nabla p + \frac{1}{Re}\Delta\vec{v}$$

One can see that the Reynolds number dictates the balance between the advective and diffusive terms. When it is small, diffusion is much stronger than advection: friction dominates over inertia and the latter becomes negligible when $Re \ll 1$. Note also ([Zey91] p. 247) that turbulent behavior has an onset at a specific Reynolds number, above which the solutions to the Navier-Stokes equations become unstable, i.e. sensitive to small perturbations, and below which it is laminar (layered). Actually, the situation is even more complicated.

Lastly, for example in [Poz09] one reads that "Turbulence is characterized by random motion in *both* time and space".

Fluid-solid boundaries

At boundaries, there is practically no *slip* (an observation which has often been contested, [Day90], but which seems to hold to a reasonable degree). I.e. at the boundary itself, the velocity field has no component tangential to the boundary, and of course also no normal component (because no fluid is moving through the boundary).

So-called boundary layers, of which there are several kinds, generally form at an interface which is not a free surface.

Stokes flow and boundary layers

When the Reynolds number is very small ($Re \ll 1$), inertial forces play almost no role and the flow, at short distances, is approximated by Stokes Flow (this is the case of the further assumption of a steady flow):

$$\eta \Delta \vec{v} - \nabla p = 0$$

The Navier-Stokes equations are thus reduced to a *linear* equation (system). Because of the boundary conditions, the velocity of the fluid at the interface equals that of the body falling through it.

We report, without further explanation, a result which can be derived from this: the Stokes Formula for the drag on a sphere moving slowly through the fluid (by which we would like to model the tracer particles, to be introduced below).

$$F = 6\pi\eta R v$$

The sphere (of radius R) experiences a drag proportional to its velocity. The velocity finally settles, under the effect of external forces (gravity), at an equilibrium value which is proportional to the difference in density compared with the surrounding fluid, a well-known result:

$$V_s = \frac{2}{9}\frac{\rho_p - \rho_f}{\mu} g R^2$$

So the equilibrium velocity is proportional to the density difference.

Of course, the particles in question are not sinking like the sphere in the above situation.

Instead, one would like to show that for an almost neutrally buoyant, small particle, there is no significant lag (since the total inertia is the same as that of an equal volume of surrounding fluid when the particle is neutrally buoyant), and that one can use them for accurately drawing out pathlines, which are curves obtained by integrating the time-dependent velocity field.

Definition of a pathline:

$$\frac{\partial}{\partial t}\mathscr{C}(t) = \vec{v}(\mathscr{C},t) \tag{2.1}$$

We assume in this work that the trajectories of particles follow closely actual pathlines of the time-dependent velocity field of the fluid, also turbulent motion up to a certain scale (the particles have, after all, a finite mass and surface.) The deviation will be neglected. A more than perfunctory discussion will have to wait.

But surprisingly, the transport of small particles by a fluid is still the subject of recent research (e.g. see [BCPP01] for neutrally buoyant spheres, and the references therein).

One can find interesting leads in publications on sedimentation of particle suspensions (ex. [Büh07] uses time-dependent flows).

2.1.2 Making flows visible and measuring them

Now one point is how to make the velocity field visible. which is of no concern in simulations nor usually in the abstract treatment of the movements of a body of fluid, but of practical importance for experiments in real environments .

One must also differentiate this from methods for assessing e.g. volumetric flow rates. Methods and goals are bound to be entirely different. Both might be important, e.g. to measure the influx of air at the source, at the same time!

Mechanical measurement

There are plenty of mechanical measurement methods (Pitot tubes, Anemometers) [CB76], but most of these have non-negligible effects on the flow and change the whole setting. Furthermore, each (expensive) unit deployed essentially measures what happens in a specific location only, and in some situations their application is not necessarily feasible.

Such probes offer either (0+1)-dimensional measurements, so to speak, of instantaneous velocities (or volume flow measurements). Particles on the other hand allow "1-dimensional" measurements and when they are deployed in great number, together with the right tracking method, essentially allow arbitrary sampling of the whole field.

These are reasons why optical methods enjoy great popularity. Our approach had to be scaled down a bit; there are no whole-field measurements like in PIV; instead, one uses a moderate number of particles and integrates over time to obtain averaged properties of velocity fields. On the other hand, it is true 3D.

Contactless measurement

At first, methods of flow *visualization* methods (cf the first part of [PVW] for overview) were employed chiefly for visual inspection [Mer87]. Often, the goal of an experiment still *is* a qualitative evaluation by a human expert but nowadays one also constructs computerized measurement systems atop of the methods originally intended for visualization.

Usually, the optical properties of a homogeneous, transparent medium do not change appreciably with the flow behavior. If they do, and the fact is exploited, one uses the name *Schlieren* – sometimes one can use thermally induced optical inhomogeneities. Apart from that, it is difficult to obtain dense, three-dimensional measurements in arbitrary media.

Smokes, which are dispersions of tiny solid particles in a gas, are also traditionally used for qualitative experiments. In water (for example), the choice of good tracers is by necessity different than in air. One can use dyes in the place of smoke.

Tracer particles

To resume the discussion on the use of particles for tracing out fluid motions (precisely, approximations to pathlines of the velocity field $\vec{v}(t)$), we want to mention that e.g. in [Mer87], one finds, with regard to tracer particles, that "the foreign material is swept along with the mean flow", which also supports the assumption.

What kinds of tracer particles are available? In PTV, as opposed to PIV (which often operates at microscopic scales and sometimes uses the properties of coherent light, which do not reduce to geometrical optics), the following is true: if one has only cameras at one's disposition, e.g. in large-scale experiments, one will want to use "macroscopic" tracer particles and one cannot use e.g. fog, whose constituent particles are so small as to be indiscernible in such a situation.

In principle, the particles do not even have to be introduced artificially, if there are already particles present, which further broadens the useful range of applications.

[Mer87] also asserts that neutrally buoyant particles "need not be extremely small" and explicitly mentions (p. 48, 2nd edition) Helium-filled bubbles as the *only* choice for tracers in air. We reproduce here (figure 2.1) a diagram of a bubble generator which was used at NASA. Similar "Soap" bubbles, albeit filled with a mixture of Helium and air, were also used in our experiments.

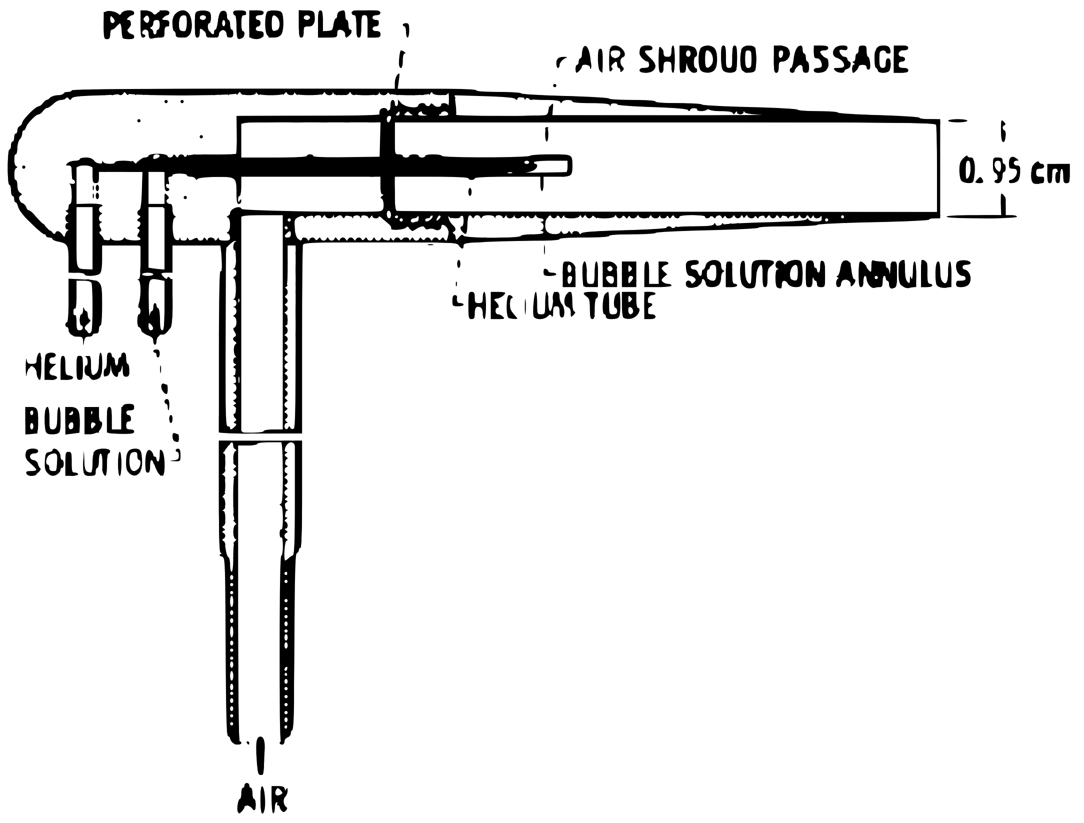


Figure 2.1: Diagram of a He bubble generator, reproduced from [CR75]

So we develop a method with isotropically shaped particles in mind, which is also a sensible approximation in other cases.

Just as it makes no sense for pathlines to cross at some moment in time, one can safely assume that particle trajectories do not cross under reasonable conditions. I.e. that because of the boundary layers and low inertia, particles do not collide. In fact, we never observed anything like a collision in the experiments.

The quality of the particles will be difficult to assess from the images; we did not attempt to do so. One can only hope that the average size/mix produced by the generator will be just right and the deviation from neutral buoyancy is very small on average. If one is willing to extend the setup, there is a laser-interferometry-based technique to be found in [NKKM08].

2.2 Repeatability of the experiments

An experiment which cannot provide the experimenter with any information relevant to similar situations (as measured, for example, in the accuracy of predictions based upon information gained from it) is not very useful.

Especially regarding the use case which drove the development (optimizing the ventilation of an aircraft cabin, which is a space where people move around in unpredictably), one needs to ask the question: what constitutes a repeatable experiment under possibly non-controllable and highly variable conditions?

The circumstances encountered "in the field" comprised cameras being positioned inside the observed volume (and not behind a viewing window) and an open system (sometimes the cabin door was left open, even when running an experiment, so air circulated not only through the ducts, but also through the door).

These are only the conditions under which the mock-up is tested; the intended use of the finished product is subject to even more vagaries.

The answer must lie in statistics; this theme will be taken up again in section 8.1.1 on page 180, near the end of the thesis – when the raw image series have been processed and interpretation and integration of results becomes possible. The best one can hope to obtain is a statistical description of the main patterns of the air flows inside. The statistical description, being built up from quantitative measurements, could be subjected e.g. to methods appropriate for vector field visualization, even though does not purport to describe the momentary velocity field.

The characteristics of the currents are often not accurately readable from just a few frames' worth of observations, even if one employed a "whole-field" method. One needs to trade time against the attainable confidence.

Life expectancy of particles

The fate of a single particle is not important at all.

As the helium bubbles have a life expectancy in the order of minutes, and the flow can be seeded only at specific places, there will certainly be regions which are more or less densely sampled. However, less particles result only in less samples, and on the whole (with a continuous stream of new particles, equilibrium is reached in a short time, of the order of the life expectancy of a particle:

let's call $\#p$ the average number of particles present and say that they are created with a constant rate g (say, in $[s^{-1}]$) and that their decay k is independent of their age (while probably not true, it's simpler this way).

This gives the well-known ODE (for convenience, let's formulate in the continuous domain)

$$\frac{\partial \#p}{\partial t} = g - k \cdot \#p \tag{2.2}$$

Solutions are $\#p(t) = \frac{g}{k} - s \cdot e^{-kt}$. If at $t = 0$ the process is started with $\#p(0) = 0$, one gets $s = \frac{g}{k}$, so $\#p(t) = \frac{g}{k}(1 - e^{-kt})$, which converges to $\frac{g}{k}$, with a "half-life" of $-\frac{\ln 2}{k}$ (then in seconds). With mean lifetime $-\frac{1}{k}$, because we just modeled the disappearance of particles as exponential decay. Which, if not actually true, is a welcome approximation.

Consequences for Sampling There is nothing to guarantee enough "mixing" for particles to reach every nook and cranny of the volume of interest (trivial example: particles won't reach regions with no circulation, i.e. near walls (cf. paragraph on boundaries; [Poz09] on the "Law of the Wall"), but also in other places where the consequently worse sampling (with a comparable level of unavoidable outliers) may constitute a real problem!

See section 8.1.1 on page 180 for all further discussion of these aspects.

2.3 Numerical simulations

One can discretize the Navier-Stokes equations and the domain in various ways and obtain a model which can be run on a computer. Nowadays, numerical simulations of fluid dynamics are gaining importance.

2.4 Lighting and photometry

A question which naturally comes up in the experimental setup is the influence of lighting conditions (positioning and nature of light sources) on the measurements.

Evidently, tracking something on an image series means that the object of tracking must be distinguishable from both background and noise (though not necessarily at all times): ensuring good contrast helps in detection tasks.

Part of working with images obtained with a *camera* should consist in understanding how the observed intensity distributions emerge. An image of a scene can often be thought of as being composed of *surfaces* with various optical properties (often only reflection is considered. See [Kaj86]).

2.4.1 Photometry and radiometry

In the computer vision literature, one finds a mix of radiometric (see [PDG05] for a reference) and photometric terms, the overall distinction between the two being that photometry concerns itself with spectrally weighted quantities (originally with respect to to the receptive properties of human eyes).

Therefore, we adopt the language of photometry, as is long-standing usage in astronomy, and not the language of radiometry, because most of the electromagnetic spectrum does not affect the sensors and we mostly rely on a model based on geometric optics and surfaces, which means that the wave nature of light is ignored.

See [Hor97] for the photometric foundations of image formation (he uses radiometric terms).

Definition of basic quantities

Quoting Horn ([Hor97]), one defines the irradiance (photometrically: the illuminance), as the power being transported (via electromagnetic radiation) onto an area element,

and the radiance (photometrically: the luminance) as the energy given off by a surface (reflected or otherwise), per steradian. The reflected part is called reflectance.

During the operation of a camera, an array of light-sensitive surfaces, usually CCD or CMOS devices (section 2.5), accumulates photons in the visible range of the spectrum and is scanned at regular intervals. In other words, it serves as a photometry performing device.

2.4.2 Lighting conditions

Going back to the experimental setup, how does one ensure good contrast?

In geometric optics, light reflected from surfaces is modeled as consisting of rays; one can easily see that the most advantageous positioning of a point light source for lighting the whole field of particles is near the cameras, because that way most light hitting a specularly or diffusely reflecting sphere is returned nearly in the direction where it came from and more light is collected at the sensors instead of being scattered into the scene.

In the present work, we use knowledge about light to improve the contrast; from other image processing tasks, one could think that lighting is important information, but we cannot work with more detailed information, since the particles' images tend to near the resolution limit and one cannot really infer anything from the luminance gathered from a particle.

It is advantageous to use a matte (Lambertian) background, if possible; any reflected images of particles would have to be eliminated using known scene geometry. Ideally, the background should not even emit any light visible to the image sensors and appear black. The option of redecorating a scene is not always available, though. Another option for improving contrast is the use of fluorescent particles.

The inverse square law is also important to note. It holds for point light sources and states that the illuminance decreases with the square of the distance between light source and illuminated surface.

[FS07] propose a way of obtaining depth information directly from defocus, although for dense scenes. This seems difficult here, and not likely to match the accuracy obtainable with stereo vision methods (but nevertheless worth an experiment!).

For similar reasons, it is not possible to use color coded illumination for the recovery of depth information. Such a procedure would require colorimetric calibration (which is difficult to obtain except under very controlled conditions, and time intensive), and when tracer particles aren't Lambertian scatterers, which they usually aren't; soap film bubbles for example exhibit a great deal of specular reflectance, the prospect of success is small.

Usable information is thus reduced to spatial geometry, and it is the role of chapter 3 to expound it.

2.5 Digital optical sensors

While a comprehensive discussion of digital optical sensors goes beyond the scope of this thesis, let us give a quick introduction into how they work, and what kind of noise one has to expect from them.

2.5.1 Photoconversion

Digital optical sensors often use p–n junctions as photoactive semiconductor diodes. Like all semiconductors, these diodes have a material dependent gap between their conductive and valance band. This band gap energy is defined as the required energy to overcome this gap and is given by:

$$E_{Gap}(T) = E_{Gap}(0) - \frac{\alpha^2}{T+\beta}$$

$E_{Gap}(0)$, α, and β are material dependent fitting parameters, while T is the temperature.

These diodes are exposed to photons, with the energy:

$$E_{Photon} = \frac{hc}{\lambda}$$

with h being Planck's constant, c being the speed of light, and λ being the wavelength of the light. If the energy E_{Photon} of incoming photons is greater than the gap energy E_{Gap} of the diode, light is absorbed.

In order to capture the visible spectrum of light, ranging from about 380 to 750 nm, a semiconductor material with fitting constants $E_{Gap}(0)$, α, and β has to be chosen. Silicon, which is commonly used for this purpose, has a band gap of about 1.11eV. Hence, light with a wavelength of up to $1100nM$ can be detected, easily covering the visible spectrum.

The rate of photon absorption in a region with thickness dx is proportional to the photon flux $\phi(x)$, with x denoting the distance ([Nak06]).

The kinds of image sensors are widely used in digital cameras: Charge coupled device (CCD) sensors, complimentary metal-oxide semiconductor (CMOS) sensors.

2.5.2 Image sensors

Image sensors consist of an array of photoactive capacitors, such as silicon photodiodes, as described above. These capacitors are known as pixels. Depending on the purpose of the sensor array, they may be one or two dimensional.

Incoming photons are converted into electric charge, as they hit the capacitors. After a certain time interval - the exposure time - has passed, the charges for each pixel are collected and transformed into voltage by a charge amplifier. There are several possible strategies for the collection of the charges.

The cameras we used in our experiments shift each charge for each row into a vertical register, and then down the 2D array into a horizontal register (HCCD register), which servers as input for a floating diffusion node. This node converts the charge onto voltage.

The required time for all charges to be transferred out of the horizontal register is called a frame and gives a lower boundary for the acquisition rate.

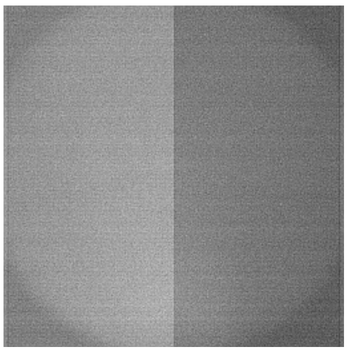
The two most common forms of such sensors are charged coupled devices (CCD) and complementary metal oxide semiconductors (CMOS) [CZZ+07]. In CCD setups, the pixels are connected to the charge amplifier through a transistor, while CMOS setups integrate a charge amplifier into each pixel.

Both CCD and CMOS sensors used in digital signal cameras operate by integrating charges as first described by Weckler in [Wec67].

2.5.3 CCD Noise

With the basic knowledge of the operation of CCD sensors, we can now review sources of noise, that affect them, and which complicate the analysis of images in chapter 4 on page 61. In [LFSK06] the authors distinguish between five sources of CCD noise: dark current noise, fixed pattern noise, shot noise. amplifier noise. and quantization noise.

Electrons can be generated spontaneously on the surface of CCD sensors through thermal vibration, even when it is not exposed to photons. This effect is known as dark current noise. It is dependent on both exposure time and temperature. Figure 2.2(a) on the following page shows a downscaled, normalized image taken in the dark with one of the cameras used for our experiments. The split in the middle is due to each half of the image being read out into a separate HCCD register and charge amplifier. On each half thermal noise can be detected, which increases with distance from the vertical read out registers. In addition vignetting affecting the whole image can be observed.


(a) Downscaled CCD image taken in the dark

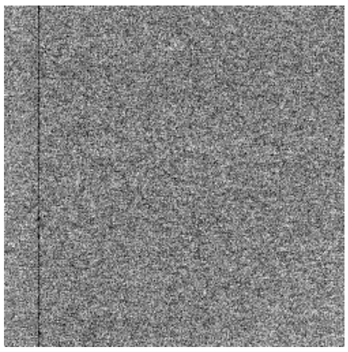

(b) Image section containing fixed pattern noise

Figure 2.2: Different forms of noise displayed by the CCD cameras, which were used in our experiments.

Unlike dark current noise, fixed pattern noise is non random, and always affects the same pixels. Figure 2.2(b) on the next page shows fixed pattern noise in form of a vertical line.

Shot noise is Poisson distributed and caused by random fluctuations in the arrival time of photons on the sensor surface. As the name suggests, amplifier noise is generated by the charge amplifiers. Quantization noise occurs during the analog to digital conversion, where ranges of analog input is mapped to the same digital output.

2.6 Conclusion

In this chapter, we discussed the backgrounds of particle tracking velocimetry from an experimental and theoretical perspective, encompassing fluid dynamics, the nature of photometric measurements and the devices one employs to perform them.

The use of small tracer particles to analyze fluid flow situations is widespread in engineering, because experiments still play a central role, for at least three reasons:

1. simple calculations are accurate enough only for very simple geometries
2. numerical simulations of fluid dynamics are not yet widespread enough
3. if a computer simulation is being planned, one will want to validate it.

We arrive at the conclusion that one can employ such a setup to gain useful information; however the photometric information is of little use – as [CZZ$^+$07] remark, no data post-processing can increase the information content without introducing information from elsewhere (external knowledge about a scene). It is difficult to know enough about the exact reflectance parameters of all the objects composing the scene.

Therefore, we had to find a way to disregard the useless information and develop a reconstruction method based on simple geometry.

It remains to see whether and how one can obtain measurements which are sufficiently accurate for practical purposes. That is the subject of the following text.

Chapter 3

Geometric preliminaries

This chapter deals with the geometric concepts which provide the basis for calibration and depth reconstruction. Starting from the basic pinhole camera model, we will discuss the calibration steps which are necessary to approximate a real world camera to the ideal pinhole model. This includes estimating the intrinsic and extrinsic parameters along with the parameters for the distortion model. As we are interested in obtaining depth information, we will investigate the epipolar geometry of dual camera setups. From there we will move on to the concept of rectification, and finally depth reconstruction through triangulation.

3.1 The camera model

Having covered the physical principles of image aquisition in section 2.5 on page 22, we now move on to a geometric model for cameras - the pinhole camera model. This model will be used throughout the thesis, and is the foundation for the other sections in this chapter.

While there are other camera models, such as the Camera Obscura, and the thin lens model, only the pinhole camera model is of relevance for this thesis. Figure 3.1 gives a schematic overview.

Optical rays can enter the camera through an opening, with a diameter known as , which is centered at the optical center C. In the pinhole camera model, this opening is restricted to c, which has the convenient effect of eliminating all aberrations, such as defocus and distortion. The rays intersect the image plane at distance f. This plane is perpendicular to the optical axis, which connects it with the optical center.

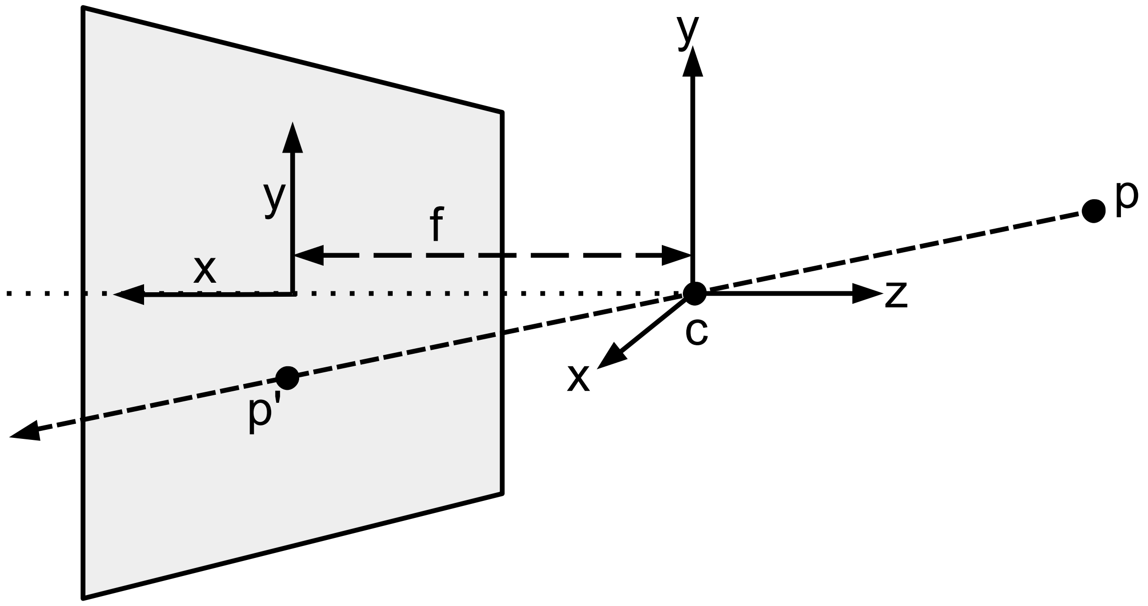


Figure 3.1: Illustration of the pinhole camera model. Point P is projected via central projection to P' on the image plane, which is at a focal distance f on the optical axis from the camera centre C.

A point P in the 3D world coordinate system is mapped to the image plane by means of perspective projection. This projection is given by [HZ04]:

$$(x,y,z)^T \rightarrow (f\frac{x}{z}, f\frac{y}{z}, f) \tag{3.1}$$

Evidently, all points are mapped to the same z coordinate, causing depth information to be lost. Its recovery is the motivation behind much of the remainder of this chapter. The other main issue is overcoming the mismatch between the model and real cameras. An infinitely small aperture is not practical for the latter. Rather, lenses are used in order to have a sufficiently large aperture for measurements while retaining focus. However, all lenses are subject to imperfections, which cause some degree of distortion.

3.1.1 Homogeneous coordinates

For each point in the scene, an optical ray can be constructed, which connects the point with the optical centre. The image coordinates of each point are given by the intersection of the ray with the image plane. A pixel in cartesian coordinates $(\frac{X}{Z}, \frac{Y}{Z})$ corresponds to the ray (X,Y,Z). A special case occurs for $Z = 0$. The 3D point $(X,Y,0)$ has no cartesian correspondence, and lies on a plane which is parallel to the image plane. Points on the former plane are known as points at infinity. Homogeneous Coordinates can be used for 3D as well, which is useful when dealing with projections between 3D spaces. This extension of euclidean geometry is known as affine geometry. In affine geometry, all lines intersect at points of infinity.

3.1.2 Projective geometry

Projective Geometry is an extension of standard Euclidean geometry. Whereas the later is well suited for dealing with Euclidean Transformations, that is rotations and translations, it is not invariant towards projective transformations. A point (x,y,z)in Euclidean coordinates can be transformed into Projective coordinates by adding an additional scale parameter $(x,y,z,1)$.Projective coordinates are invariant to scale, so $(\lambda x, \lambda y, \lambda z, \lambda) = (x,y,z,1)$. Transforming from projective to euclidean coordinates is done by dividing each coordinate by the scale factor. Therefore, all projective coordinates have the same corresponding Euclidean coordinate, as long as they only differ in the value of λ.

3.2 Stereo camera system

Recovering spatial information from images is a necessary subgoal in the construction of any PTV application.

In this section, we shall recall the geometric laws relating multiple views of a scene. These laws can be deduced from the pinhole camera model encountered in the preceding section (3.1 on page 28) where a scene is recorded by several cameras, and we will apply them to our stereo setup as a first step to spatial geometry reconstruction.

The laws of stereo vision are well-documented in the literature. There exists a number of excellent sources, [FLP01, HZ04] amongst them. The mathematics described therein constitute a necessary ingredient for the identification of the images of a point in space (namely, a surface feature of a visible object – or, in this case, a small particle in a body of fluid), as it is projected onto the cameras' image planes.

Stereo geometry describes a highly idealized aspect of the imaging system, an idealization of immense practical value for simplifying reconstruction.

3.2.1 Binocular vision

Binocular vision systems can be viewed as being descended from natural, biological antetypes. In the animal kingdom, we find many examples of naturally occurring binocular vision systems superficially resembling two-camera setups (with interesting exceptions: [MLKC91]).

However, the analogy ends right with the optics. No-one has been able to build a general-purpose vision system yet, since that would require understanding of the world (as B.K.P. Horn noted in the classic [Hor97] – that is true in 2009 as it was in 1987).

The fact that a human observer is able to pick out particle motions is therefore of little consequence in the conception of such a system. The limitations, however, are: even we are not able to discern the "depth component" of the motion of a particle which is far away, in the absence of e.g. lighting cues. This points to the baseline dilemma illustrated and partly quantified in sections 3.6 and 3.6 and in the sections on multi-camera setups – a tradeoff between number of cameras, distance between them, field of view.

More than two views There are, neither in the general formulation nor in our prototype of a system, any structural limitations precluding the inclusion of oligo-camera data sources instead of binocular only.

Multiple-view theory is really quite general and allows the incorporation of more cameras, the downside being a more complicated formulation of the relations between the views, and a higher computational load.

Skip to section 8.5 of this thesis for an exposition of the advantages of a tri-camera system. The benefits are possibly increased detection capabilities (subsection 3.6.1) and higher accuracy. With a two-camera system, however, no tensors of order higher than two appear. With more then 4 cameras [HZ04], one can reconstruct the whole scene from two by two (or three by three) views. The multiple-view tensor method, [HZ04] finds, becomes unavailable.

Incidentally, due to constraints in the experimental setup, we used only two cameras and the prototype we developed is partly built around that assumption (but could be upgraded for more cameras, e.g. also for an expanded field of view).

However, as our algorithm was built specifically around the limitations of two-view geometry in an explicit attempt to overcome them by

- building scene knowledge into the process
- exploiting the time series by integrating information over several frames.

The binocular situation will be emphasized in most of the text.

3.2.2 Incidence relations

Stereo geometry finds a natural formulation in the language of synthetic projective geometry, where any two lines meet, even parallel ones (*synthetic* projective geometry doesn't have a notion of parallelism; that arises only through the affine geometry).

Indeed, one advantage of using projective geometry is that no special cases need to be treated when a point or line lies at infinity. Another is that translations of objects in n-dimensional space become special rotations in $n+1$ projective space.

An optical ray is just the "join" of a pinhole camera's optical centre and the image point; its "meet" with the image plane constitutes the image point.

$$(O_i \wedge P) \vee (I_i) = P_i$$

Or the other way round, the image point is where the optical rays meet in a "degenerate" fashion (i.e. a special situation: their meet isn't empty).

$$(O_1 \wedge P) \vee (O_2 \wedge P) = P$$

The same situation can also be described, in the binocular case, by asserting that the views satisfy the Epipolar Constraint ([FLP01, Fau06] and others):

Since a line is determined by a vector and especially the epipolar line's parameters can be written as a vector, and in the two-dimensional projective geometry of the image plane, points are dual to lines (see e.g. the introductory chapters of [HZ04]).

An analysis of the situation (to be found in e.g. [LF95]) shows that the correspondence must indeed be linear: $l' = \underline{F}x$ (in this formula, one can interpret l' to be a vector perpendicular to the line). As a correspondence between image points, the linear transformation must thus have rank 2 only, because of the incidence which must be realized, and one obtains $x'^T l' = 0$ or $x'^T \underline{F}x = 0$, the so-called *Epipolar Constraint*.

[HZ04] also has a wonderful discussion of the various geometric error functions appropriate to stereo geometry. We will mostly use reprojection error (mean distance between the individual 2D points and their images after reprojection from 3D space).

3.2.3 Euclidean reconstruction

Views whose calibration data is known allow full Euclidean reconstruction, i.e. up to an Euclidean transformation, i.e. with angles and distances, no sooner than the correspondence between surface points can be established.

Intrinsic vs. extrinsic

The intrinsic parameters of a camera are those that do not change when the camera undergoes an Euclidean transformation. They are properties of the camera optics alone, while the extrinsic parameters of a camera in a scene describe its pose (translation and rotation) relative to that scene.

Both intrinsic and extrinsic parameters, as we know from the pinhole model, are given by linear transformations. The extrinsic parameters are Euclidean transformations which allow conversion between the camera-centered coordinate systems and into the world coordinate system. The intrinsic ones convert between the pixels on the actual image and the camera-centered coordinate systems used for measurements in scene space.

For the whole system, when all parameters are determined up to one of the rotation/translation matrices, the relative pose is known since the choice of world coordinate system is arbitrary. One can then interpret measurements geometrically.

There was originally no question of moving the cameras; but since one could feel the need to reposition them, we provided the software prototype with a repositioning module.

Calibration

To calibrate a stereo camera system means to determine the two sets of parameters mentioned above: the 6 degrees of freedom of the extrinsic parameters, the 5+2 degrees of freedom of the intrinsic parameters (projective-linear + distortion).

3.3 Calibration

Camera calibration gives us a way to estimate the parameters of a camera. Upon knowing these parameters, we can regain world coordinate information from projected points on the image plane. Calibration is therefore essential to computer vision and it comes at no surprise that there exists a wealth of literature on this subject. The accuracy of the depth reconstruction directly depends on the calibration, which made the selection of a suitable calibration algorithm important.

We are able to chose between various approaches to camera calibration, depending on the target application. In general, calibration techniques can be classified by the use of calibration objects. In his paper from 2000, [Z$^+$00], Zhengyou Zhang refers to calibration techniques without calibration objects as self calibration methods, whereas the use of such objects is labeled as photogrammetric calibration. We shall discuss one method from either category.

If no calibration objects are used, the world coordinates of some points in the scene has to be known. These points can then be used for calibration. As we will see, this approach allows camera calibration from a single image. In contrast, using a calibration object only requires knowledge about the geometry of the object. This makes it possible to easily conduct calibrations in new environments. The drawback is that more than one view is required for calibration, making the calibration process more time intensive.

3.3.1 Intrinsic and extrinsic parameters

In order to approximate a real camera to this model, two sets of parameters have to be determined. The first set describes the mapping of 3D optical rays to 2D pixel coordinates and is referred to as the intrinsic parameters. They are dependent on the camera (including objective/lens), but not on its position in the world coordinate system. They can be represented by the following matrix:

$$A = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.2}$$

Where $\alpha_{x,}\alpha_y$ represents the focal length in x and y direction (measured in pixels), γ the skew between the x and y axes on the image plane, and (u_0, v_0) the principal point, which is the centre of the computer of the the image frame.

The other set is known as the extrinsic parameters and described the position of the camera in the Euclidean Space. It consists of the $(3x3)$ rotation matrix R, and the $3x1)$ translation vector T.

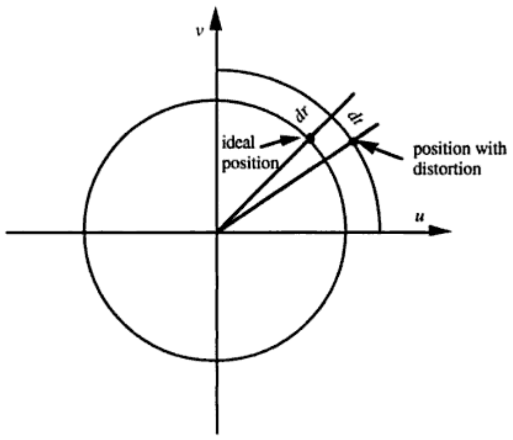


Figure 3.2: Effect of radial (*dr* and tangential *dt* distortion. Illustration taken from [ZR96]

Knowing both sets of parameters allows us calculate the image point p_i of a point p_w in the world coordinate system:

$$p_i = A[RT]p_w$$

3.3.2 Distortion

There are two main forms of lens distortion: tangential and radial lens distortion. Figure 3.2 shows both kinds. Radial distortion shifts each point from its ideal location $p = (x,y)$ along the optical axis, while tangential distortion displaces it along the tangent of the circle, which is centered at the principle point $c = (u_0, v_0)$, and which radius is given by the length of the line segment from c) to p. Unlike radial distortion, tangential is negligible in practice.

Radial distortion can either have a pincushion or a barrel effect, depending on whether it is positive or negative. Figure 3.3 on the following page shows both forms. Negative radial distortion is known as *barrel distortion* and causes points to be moved closer together, as distance from c increases. Analogical to this, *pincushion distortion* refers to positive distortion and leads to points being shifted further apart with growing distance from c . In general, radial distortion is caused by imperfect radial curvature of the lens.

3.3.3 Calibration after Tsai

In his seminal 1987 paper [Tsa87], Tsai outlined a calibration algorithm for determining both sets of camera parameters, as well as two parameters for the radial distortion. Tangential distortion is ignored. The algorithm requires seven non coplanar points with

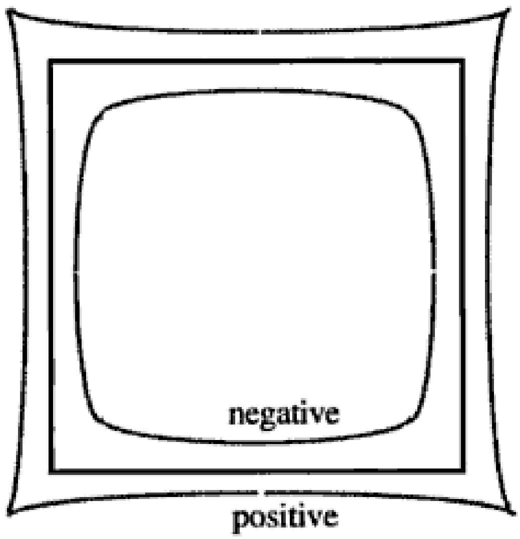


Figure 3.3: Pincushion (positive) and barrel (negative) radial distortion illustrated. Illustration taken from [ZR96]

known world coordinates, and proceeds in four steps, which we will outline in the following.

Initially, the transformation from the world coordinate system to the camera's 3D coordinate system is determined. That is, the rotation matrix R and translation vector T, which transforms a point (x_w, y_w, z_w) in the world coordinate system, to the 3D camera coordinates (x, y, z)

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T$$

We will come back to the general problem of determining the transformation between 3D coordinate systems in section 3.3.5 on page 43, albeit between systems for which the point coordinates are known. For now, let us continue with the description of Tsai's algorithm.

The next step is to map the 3D camera coordinates to 2D image coordinates (x_i, y_i) using the perspective projection 3.1 on page 28 (replicated here for convenience):

$$(x, y, z)^T \rightarrow (f\frac{x}{z}, f\frac{y}{z}, f)$$

This step yields the focal distance f.

Having obtained f, the next step is to estimate the radial distortion coefficients. We can wait with this (effectively ignoring distortion for the determination of the intrinsic parameters), because radial distortion is small, and we optimize all parameters in the final step. Distorted image coordinates (x_d, y_d)relate to undistorted image coordinates (x_u, y_u) by:

$$x_d + D_x = x_u$$
$$y_d + D_y = y_u$$

With D_x, D_y being defined as:

$$D_x = x_d(k_1r^2 + k_2r^4 + ...)$$
$$D_x = x_d(k_1r^2 + k_2r^4 + ...)$$

Tsai only uses one term for radial distortion, arguing that more introduce numeric instability for little gain in the quality of the undistortion. [Tsa87].

In the final step, image coordinates are mapped to pixels (x_f, y_f) by:

$$x_f = s_x d_x'^{-} 1 x_d + u_0$$
$$y_f = d_y^{-} 1 y_d + v_0$$

The distance between centres of adjacent sensor elements in X and Y direction is represented by d_x, and d_y, while the derivative of d_x' is given by: $d_x' = d_x \frac{N_c x}{N_f x}$. $N_c x$ stands for the number of sensor elements in a horizontal direction, and $N_f x$ is the total number of pixels in a line. $N_c x$. These two parameters should be provided by the manufacturer of the camera. As should d_x, and d_y, which cannot be relied upon,however, forcing the introduction of a scaling parameter s_x.

3.3.4 Zhang's calibration method

Zhang's algorithm uses a planar calibration pattern with a checkerboard texture to estimate the intrinsic matrix and four of the distortion coefficients. Figure 3.4 on the next page gives an example for such a pattern. At least two poses are required, but in practise many more are needed for accurate results. Depending on the used pattern, a large number of feature points have to be detected.

The calibration methods operates by estimating a homography between the calibration pattern and its image. This homography is then used to derive constraints on the intrinsic parameters, which are then used to estimate the calibration parameters through a closed

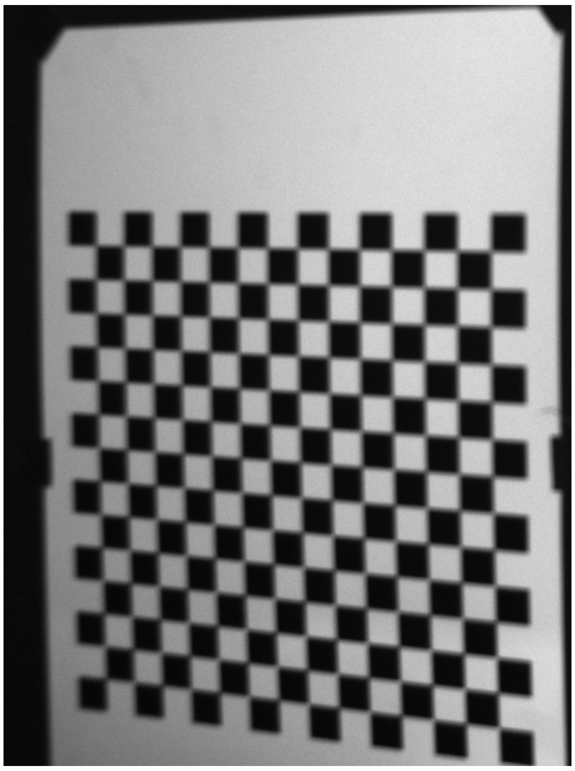


Figure 3.4: The calibration object which was used for the experiments.

form solution. Zhang presents a Linear Least Squares method to estimate the radial distortion coefficients. In a final step, all parameters are minimized together in a maximum likelihood estimation.

In the following, we will describe each step of the process in more detail.

In order to establish the homography, we first assume, without loss of generality, that the calibration object lies at $Z = 0$ in the world coordinate system. This allows us to simplify the projection relation by ignoring the translation and rotation related to the Z axis. In particular, that is the third component of the translation vector t, and the third column $r3$ of the rotation matrix R. The projection between a point $P_i = (u, v, 1)^T$ on the image plane, and a point $P_w = (x, y, z)^T$ in the world coordinate system is then given by:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A[r_1 r_2 t] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

With A being the intrinsic matrix, as defined in 3.2 on page 34, s being a scalar, and r_1, r_2 being the first and second column of the rotation matrix R. The desired homography

$H = A[r_1 r_2 t]$ is the product which relates the two points:

$$sP_i = HP_w \tag{3.3}$$

This homography is a $3x3$ matrix and can be estimated using non linear least squares approximation. The homography should satisfy (3.3). We refer to it in the following as $H = [h_1 h_2 h_3]$.

It is obvious from the above that

$$H = A[r_1 r_2 t] = [h_1 h_2 h_3] \tag{3.4}$$

Knowing that the rotational axes $r1$ and $r2$ are orthonormal, two constraints can be established from the homography for the intrinsic parameters:

$$h^T{}_1 A^- T A^{-1} h_2 = 0 \tag{3.5}$$

$$h^T{}_1 A^- T A^{-1} h_1 = h_2{}^T A^- T A^- 1 h_2 \tag{3.6}$$

We are now ready to estimate the intrinsic parameters through a closed form solution:

$$B = A^T A^- 1 = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma - u_0\gamma}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{v_0\gamma - u_o\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0^2}{\beta^2} + 1 \end{bmatrix} \tag{3.7}$$

We are now ready to extract the intrinsic parameters from B:

$$v_0 = (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2) \tag{3.8}$$

$$\lambda = B_{33} - [B_{12}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})]/B_{11} \tag{3.9}$$

$$\alpha = \sqrt{\lambda/B_{11}} \tag{3.10}$$

$$\beta = \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)} \tag{3.11}$$

$$\gamma = -B_{12}\alpha^2\beta/\lambda \tag{3.12}$$

$$u_0 = \gamma v_0/\beta - B_{13}\alpha^2/\lambda \tag{3.13}$$

With the intrinsic parameters known, it remains to calculate the external orientation of the camera. For this purpose equation 3.4 on the previous page can be used to yield:

$$r_1 = \lambda A^{-1} h_1 \tag{3.14}$$

$$r_2 = \lambda A^{-1} h_2 \tag{3.15}$$

$$r_3 = r_1 x r_2 \tag{3.16}$$

$$t = \lambda A^{-1} h_3 \tag{3.17}$$

The required number of calibration poses depends on whether skew can be neglected or not. If it can be assumed that $\gamma = 0$, two poses are required. Should skew be considered, at least three poses are needed. Of course, as with Tsai's method, many more poses are needed to compensate for noisy input data.

Estimating distortion

Thus far we have a calibration for a perfect pinhole camera. We must now compensate for distortion. Like Tsai, Zhang only considers radial distortion, and uses only the first two coefficients of the model. Radial distortion is assumed to be small enough that an useful first estimate of the intrinsic parameters is possible while ignoring distortion. In order to estimate the coefficients, we introduce (u, v) as undistorted pixel coordinates, and $(\tilde{u}, \tilde{v})$ as the measured coordinates, which are subject to distortion. In the same vein, (x, y), and $(\tilde{x}, \tilde{y})$ are distortion free and distorted image coordinates, respectively. We can now write as model for the radial distortion:

$$\tilde{x} = x + x[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]$$
$$\tilde{y} = y + y[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]$$

The distortion coefficients are denoted by k_i. Radial distortion is centered on the principle point. If we assume that $\gamma = 0$, and knowing that $\tilde{u} = U_0 + \alpha\tilde{x} + \gamma\tilde{y}$ as well as $\tilde{v} = v_0 + \beta\tilde{y}$, we arrive at a radial distortion model for pixel coordinates:

$$\tilde{u} = u + (u - u_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \tag{3.18}$$
$$\tilde{v} = v + (v - v_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \tag{3.19}$$

At this point we know the undistorted pixel coordinates (u,v) from having estimated the intrinsic parameters earlier. If we apply them to equations (3.18) and (3.19), we get two equations for each point:

$$\begin{bmatrix}(u-u_0)(x^2+y^2) & (u-u_0)(x^2+y^2)^2 \\ (v-v_0)(x^2+y^2) & (v-v_0)(x^2+y^2)^2\end{bmatrix}\begin{bmatrix}k_1 \\ k_2\end{bmatrix} = \begin{bmatrix}\tilde{u}-u \\ \tilde{v}-v\end{bmatrix}$$

We are now ready to optimize the all parameters together through maximum likelihood estimation. For this purpose, we use the m model points m_{ij} in the n images, and minimize the square error with their projections $\tilde{m}_{ij}$. The optimization term is given by

$$\sum_{i=1}^{n}\sum_{j=1}^{m} ||m_{ij} - \tilde{m}_{ij}(A, k_1, k_2, R_i, t_i, M_j)||^2$$

In order to solve this non linear problem, the Levenberg Marquardt Algorithm can be used, as described in [Mor77].

For our experiments, we calibrated the cameras using nearly 300 poses of the calibration object.

Figure 3.5 on the following page shows the reprojection error after calibration. Several outliers are clearly visible, resulting from failed corner detection. These were manually removed by us, but it would be conceivable to filter them automatically, using a Random Sample Consensus algortihm [FB87].

Summary

In this section, we defined the calibration parameters, which have to be obtained, before giving an overview over the two most common approaches to camera calibration. We discussed Roger Tsai's method as one example for the photogrammetric, along with our chosen method by Zhengyou Zhang.

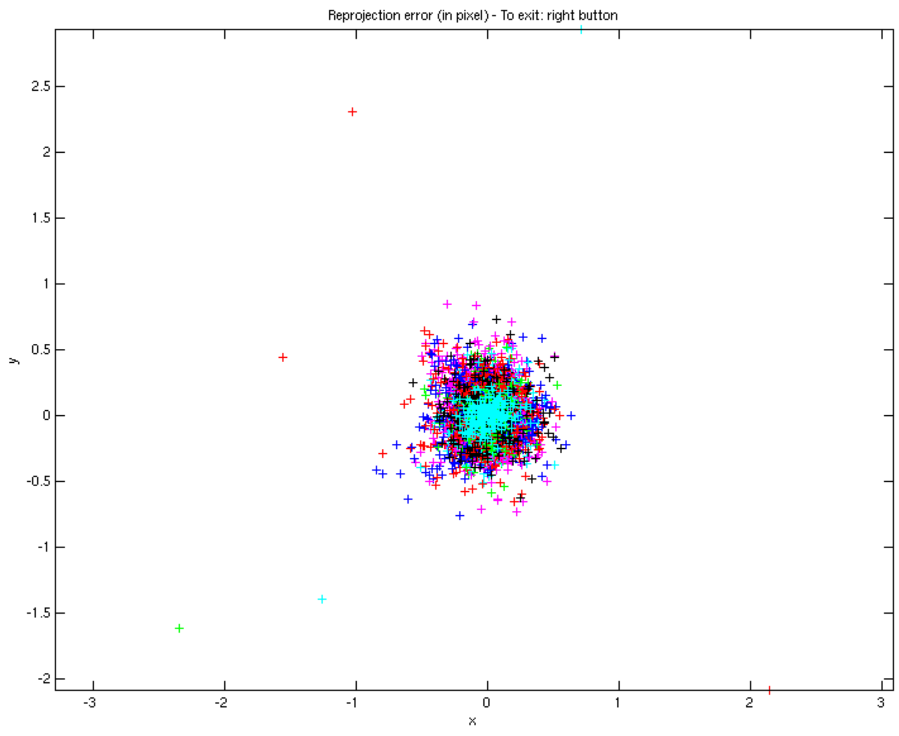


Figure 3.5: Reprojection error using 300 calibration poses after conducting calibration after Zhang.

3.3.5 Obtaining relative and absolute orientation

For stereo cameras to be of any use at all, we must know their relative positions. Without this knowledge, depth reconstruction is impossible. Further, it won't do to only know the positions of objects relative to the cameras. We must also know the positions of the cameras in the world coordinate system that is used by whomever conducts the experiments. In our case, the world coordinate system is the cabin, and we have to obtain the positions of the particles in it.

We used the Matlab Calibraton Toolbox [Bou08] to obtain the relative orientations of the cameras, using the same calibration pattern as before. Figure 3.6 shows the calibration result.

In principle, both problems can be solved in the same fashion if points with known locations in the world coordinate system are used, which is why we will restrict ourselves to the discussion of how to obtain the absolute orientation. Closed form solutions for this problem exists, (e.g. [H$^+$87] and [MB]. We used an implementation of [H$^+$87], which solves the problem using quaternions, and which will be presented in the following.

We are looking for a transformation RT which maps a point p_s in the scene coordinate system to a point p_c in the camera coordinate system.

$$p_s = RT\,p_c$$

This gives us six degrees of freedom, which means at least three non colinear points are required get the needed seven constraints.

The problem can be solved using a least squares approach. However, a closed form approach can also be used.

First the axes need to be constructed for each camera (referred to x_s, y_s, z_s, and x_c, y_c, z_c) . A unit vector in the direction of the x axis can be constructed from the normalized difference vector of a pair of corresponding points $p_{s,i}, p_{c,i}$:

$$x_s = \frac{p_{s,1} - p_{s,1}}{||p_{s,1} - p_{s,1}}||$$

The unit vector in the direction of the y axis must be perpendicular to x_s. We obtain it by calculating the tangential component:

$$y_s = \frac{p_{s,3} - p_{s,1} - ((p_{s,3} - p_{s,1})x_s)x_s}{||p_{s,3} - p_{s,1} - ((p_{s,3} - p_{s,1})x_s)x_s||}$$

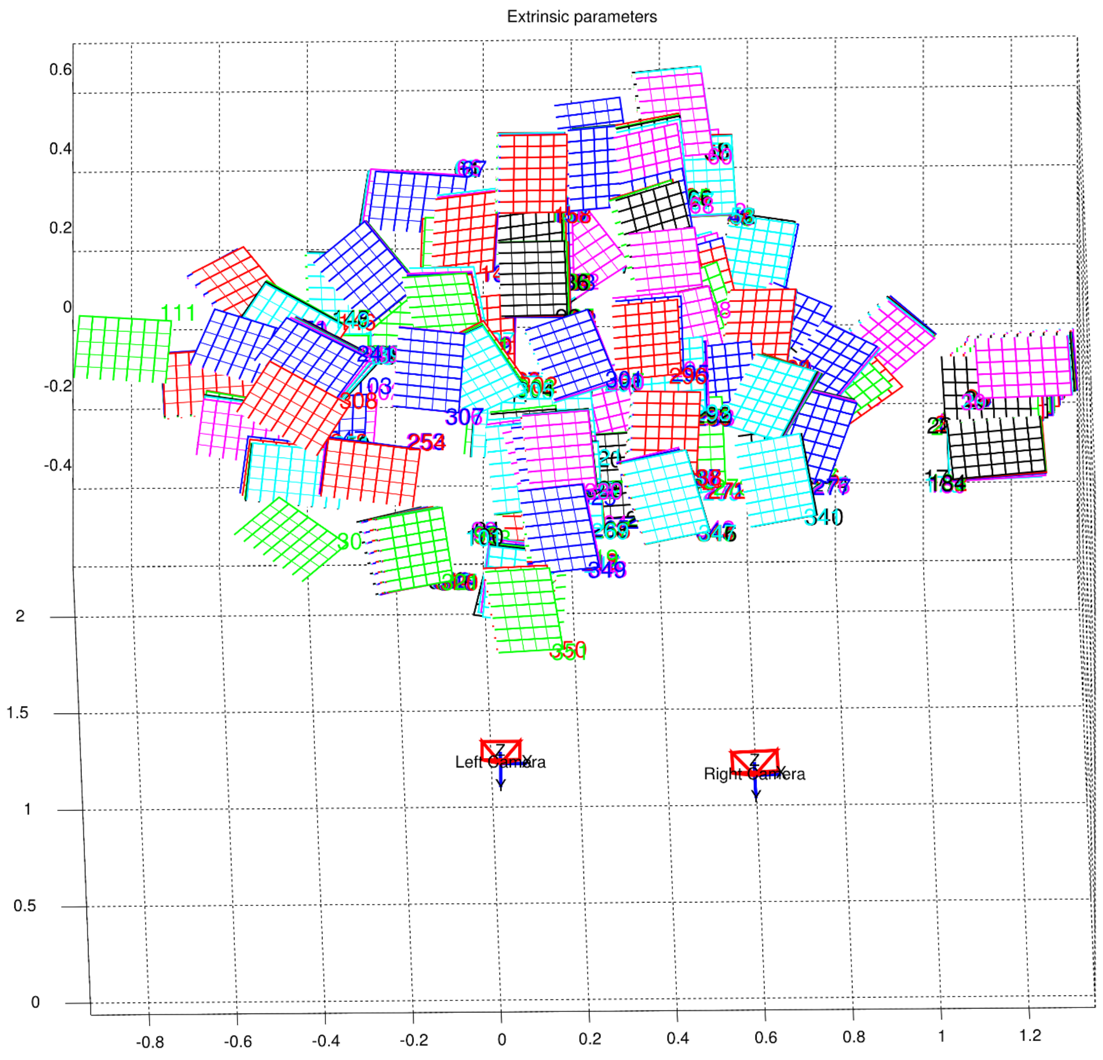


Figure 3.6: Relative orientation of the two cameras in experiment $V2$, obtained from 300 poses

The z axis needs to be perpendicular to both other axes, and is given by the cross product of both:

$$z_s = x_s x y_s$$

x_c, y_c, and z_c are generated in an analogous fashion. The vectors can be adjoined to the matrices $M_s = |x_s, y_s, z_s|$ and $M_r = |x_c, y_c, z_c|$.

We can express the rotation between the coordinate systems in terms of M_s and M_c:

$$R = M_c M_s^T$$

Solving this above allows us to calculate a first estimate for the rotation.

We now write the transformation between p_s and p_c as:

$$p_s = st(p_c) + t$$

This leaves the translation and scaling to be determined. In order to find the translation, we can exploit that rotation is a linear, length preserving operation. In order to find the translation, the centroids of the two point sets, $\bar{p}_w, \bar{p}_c$ is calculated. The translation is then given by:

$$\bar{p}_s = sR(\bar{p}_c)$$

The scale can be calculated by:

$$\sum_{i=1}^{n} p'_{s,i} R(p'_{c,i}) / \sum_{i=1} ||p_{c,i}||^2$$

Where $p'_{s,i}, p'_{c,i}$ are defined as the difference between the measurement point for that image, and its centroid.

3.4 Undistortion

As stated before, there are various models for dealing with lens distortion, i.e. non-(projective-)linear deviations from an ideal pinhole camera situation.

Let us mention, in passing, our own undistortion method, which is currently employed and is in principle compatible with the Matlab toolbox [Bou08]

3.4.1 Distortion like undistortion

We use a model for lens distortion which affects the image isotropically (when viewed from the “center point”), which is only reasonable for a lens sitting perfectly parallel to the sensor array. For that reason, [HS97] also introduces tangential terms to capture the non-radial effects. However, the tangential terms are often less important than the radial ones. [Zha99], relying on sources cited there ([Tsa87]), asserts that “it is likely that the distortion function is totally dominated by the radial components”, and goes on to say that further terms just cause numerical instability. This vindicates the design choice of going by only two coefficients and dealing with the remaining distortion in different ways (practically as in 6.3.3 on page 137 and theoretically as in the parts dealing with matching).

Following [Zha99], lens distortion is thus modeled by a one-dimensional power series in the radial coordinate r. Most authors use complicated expressions for the undistortion, some iterative, some closed-form.

If one wants to apply undistortion to a whole image (via image warping), one actually needs only the distortion function. However, in the effort for a more flexible stereo vision engine, we tried to adopt the same model for forward and reverse undistortion.

Let us reiterate the usual distortion expression and its meaning:

$$x_d = o_x + f(x_u - o_x) = o_x + (x_u - o_x) \cdot (1 + k_3 r^2 + k_5 r^4) \tag{3.20}$$

These are the usual radial terms ($r = \|\vec{x} - \vec{o}\|$, and o being the footpoint), analogous for y_d, which are taken from the Taylor Series expansion

$$f(r) \approx f(0) + f' \cdot r + \frac{1}{2} f'' \cdot r^2 + \dots \tag{3.21}$$

(quite cavalierly, we think, throwing away the even terms).

Observing the pincushion effect, the impression imposes itself that one could counter it almost precisely with a barrel distortion.

In other words, it would be desirable if one could use the same model for distortion and undistortion. Unfortunately, an inverse of a polynomial function (on a domain where it's unambiguous) needn't be a polynomial function. But the nice thing about Taylor series (if/where convergent) is that one can truncate them. The "inverse" of a polynomial function is a rational function, thus analytic, and its Taylor expansion exists.

$$f^{-1} \approx f^{-1}(0) + (f^{-1})' \cdot r + \frac{1}{2}(f^{-1})'' \cdot r^2 + \dots$$

What are these terms and how do they relate to the distortion coefficients? One calculates the following, suggesting that the previously lower-order terms need higher-order corrections (of arbitrarily high orders, in fact). Experiments (3.4.2 on the following page) demonstrate that even distortion with just a quadratic term cannot be countered with only a quadratic term.

$$\begin{aligned}
(f^{-1})' &= \frac{1}{f' \circ f^{-1}} \\
(f^{-1})'' &= \frac{-1}{(f' \circ f^{-1})^3} \cdot f'' \circ f^{-1} \\
(f^{-1})^{(3)} &= \frac{3}{(f' \circ f^{-1})^5} \cdot (f'' \circ f^{-1})^2 + \frac{-1}{(f' \circ f^{-1})^4} \cdot f^{(3)} \circ f^{-1} \\
(f^{-1})^{(4)} &= \frac{-15}{(f' \circ f^{-1})^7} \cdot (f'' \circ f^{-1})^3 \\
&\quad + \frac{10}{(f' \circ f^{-1})^6} \cdot (f'' \circ f^{-1}) \cdot (f^{(3)} \circ f^{-1}) \\
&\quad + \frac{-1}{(f' \circ f^{-1})^5} \cdot (f^{(4)} \circ f^{-1})
\end{aligned}$$

Using q for the "inverse distortion coefficients", and evaluating at 0,

$$\begin{aligned}
q_2 &= -k_2 \\
q_3 &= -k_3 + 2 \cdot k_2^2 \\
q_4 &= -k_4 + 5 \cdot k_2 k_3 - \frac{15}{3} \cdot k_2^3 \\
q_5 &= -k_5 + \\
\ldots
\end{aligned}$$

3.4.2 Results

For the tests, a 640x480 image was used; and the center of distortions was artificially positioned at (320,240) and a point was followed, originally $r = 310.7$ pixels from the origin.

Top to bottom, left to right, 3.7 on the next page shows:

1. Top left corner of the original image.
2. Distorted with $k_2 = 5 \cdot 10^{-4}$. Deviation now 47.6
3. Undistorted with $q_2 = -k_2$ only. Deviation 20.7
4. Undistorted with q_2 and q_3. Deviation 7.91
5. Undistorted with q_2 and q_3 and q_4. Deviation 4.3
6. Undistorted with up to q_5. Deviation 2.12, ca. 5% of Δr.
7. Distorted with $k_3 = -5 \cdot 10^{-7}$ (no quadratic term)
8. Undistorted with $q_3 = -k_3$ (here). It does not get any better with q4 or q5.

In spite of the images, we found that it was generally best to shun higher-order terms for distortion and undistortion alike, and that sometimes the 4th and 5th were of the same order of magnitude and were needed to counterbalance each other.

As a conclusion, we note that this undistortion converges too slowly this way, and one should stay with the traditional methods.

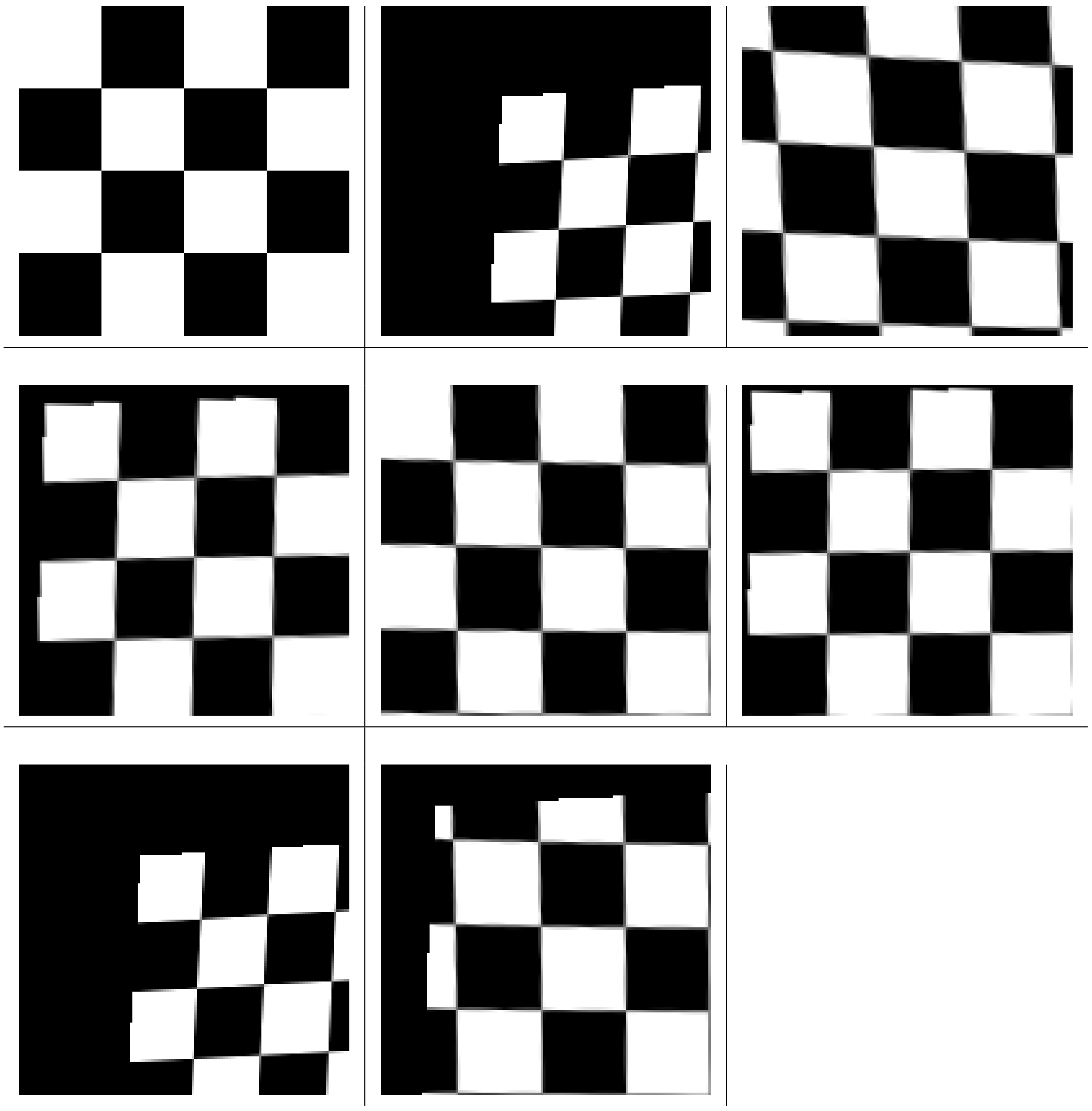


Figure 3.7: Distortion and undistortion. Legend on the facing page.

3.5 Rectification

When a multi-camera setup is used for depth recovery, the question of how to establish correspondences between 2D image points on the various camera image planes arises. We use epipolar geometry to generate a constraint, which limits the possible matches to a line. With the help of a rectification algorithm, arbitrary stereo camera configurations can be treated in the manner that is most suitable for the computational correspondence search. As the developed system is meant to be used for industry application, this performance enhancing step is necessary.

3.5.1 Epipolar geometry

Epipolar geometry describes the geometric relations in a stereo camera setup. Figure 3.8 on the facing page a) illustrates such a setup. The optical centers c_1, c_2 are separated by a baseline b. A point P is projected to the respective image planes at p_1, p_2, where the rays from it to the optical centres intersect the image planes. The optical centers themselves are projected into each other's image planes in the same fashion, at e_2, e_1. These points are known as the epipoles. A line from P through c_1 is seen on the image plane of the other camera as a line from e_2 through p_2.

This implies that if we know the location of either p_1 or p_2, we can determine the location of the other point up to a line. There exist such a line for every on the image plane. As these lines are not generally parallel, it is computationally expensive to compute the epipolar line for each point. If they were parallel, however, finding them would be trivial. This is the goal of Rectification, and will be the subject for the remainder of this section.

3.5.2 Determining the point projection matrices

As has been stated above, the goal of Rectification is to obtain parallel epipolar lines. In particular, they should be parallel to the scanlines of the respective image planes. This is achieved by rotating both image planes around their optical center c_i until they are coplanar and parallel to the baseline. Epipoles are moved into infinity in such a configuration, resulting in the desired parallel epipolar lines.

Rectification is possible with both known and unknown calibration data. As we have already obtained the intrinsic and extrinsic parameters, we will only consider rectification for the calibrated case. Let us refer to [FI08] for the other case.

If the epipolar geometry between the cameras is known, there are several ways of calculating the rectifying transformations. One approach, presented by Zhang in [LZ99], is based on the decomposition of the transformations into a projective transform, a similarity

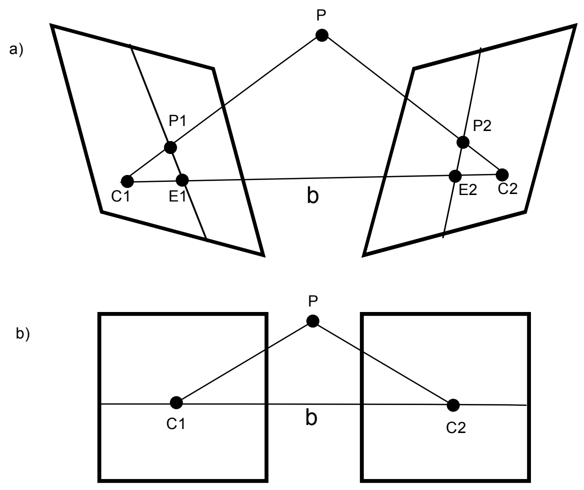


Figure 3.8: Figure a) illustrates the epipolar search problem for unrectified images. The epipolar lines are not parallel to the scanlines, making the matching problem non trivial. b) shows two image planes after rectification. The epipolar lines are parallel to the scanlines, allowing for easy matching

transform, and a shearing transform. It minimizes the image distortion, which is induced by the rectification.

We choose to use the algorithm developed by Fusiello et al., and published in [ATV00], because it is linear, and therefore more suitable for performance critical systems. In the following, the algorithm will be presented.

First off, we define a Point Projection Matrix(PPM) as the matrix P, which maps a point $p_i = (u, v, 1)$ on the image plane to the corresponding point $p_w = (x, y, z, 1)$ in the world coordinate system.

$$p_i = Pp_w$$

Using QR decomposition, P can be separated into the intrinsic and extrinsic parameters, which we are already familiar with from section 3.3.1 on page 34.

$$P = A[R|T] \tag{3.22}$$

The coordinates of the optical center can be obtained by:

$$c_i = -R^{-1}T$$

This allows us to write P as:

$$P = [R| - Rc_i] \tag{3.23}$$

In order to simply notation, we refer to R as a vector of row vectors:

$$R = \begin{bmatrix} r_1^T \\ r_2^T \\ r_3^T \end{bmatrix}$$

We initially have one such PPM for each camera; P_{ol} and P_{or}. The elements of P_{ol} are denoted as $l_{i,j}$, and the elements of P_{or} are denoted by $r_{i,j}$.

After Rectification, the configuration is described by two new PPMs, P_{nl}, and P_{nr}. The position of the of the optical centers remains unchanged from the original PPMs. The new PPMs share the same rotation matrix R_n. In order to ensure that all corresponding points share the same vertical coordinates, the two new PPMs must also share the same intrinsic parameters. Their values can be arbitrary, as long as the are equal for P_{nl}, and P_{nr}. Using (3.22) and (3.23), we can write the new PPMs as :

$P_{nl} = A_n[R_n| - Rc_1] \quad P_{nr} = A_n[R_n| - Rc_2]$

As stated above, the new coplanar image plane must be parallel to the baseline connecting c_1 and c_2. This gives the following equation for the new X axis:

$$r_{n1} = (c_1 - c_2)/||c_1 - c_2||$$

Using the knowledge that the rotation axes of R_n must be orthonormal, we can obtain the X and Z axes as follows:

$$r_{n2} = r_{ol3}xr$$

$$r_{n3} = r_{n1}xr_{n2}$$

Instead of using r_{ol3}, any other unit vector could be used as well.

A can be simply calculated by taking the average of A_{ol} and A_{or}.

The rectifying image transformations which map the original image planes to their rectified counterparts are then given by T_1, T_2:

$$T_1 = P_{nl}P_{ol}-1$$

$$T_2 = P_{nr}P_{or}-1$$

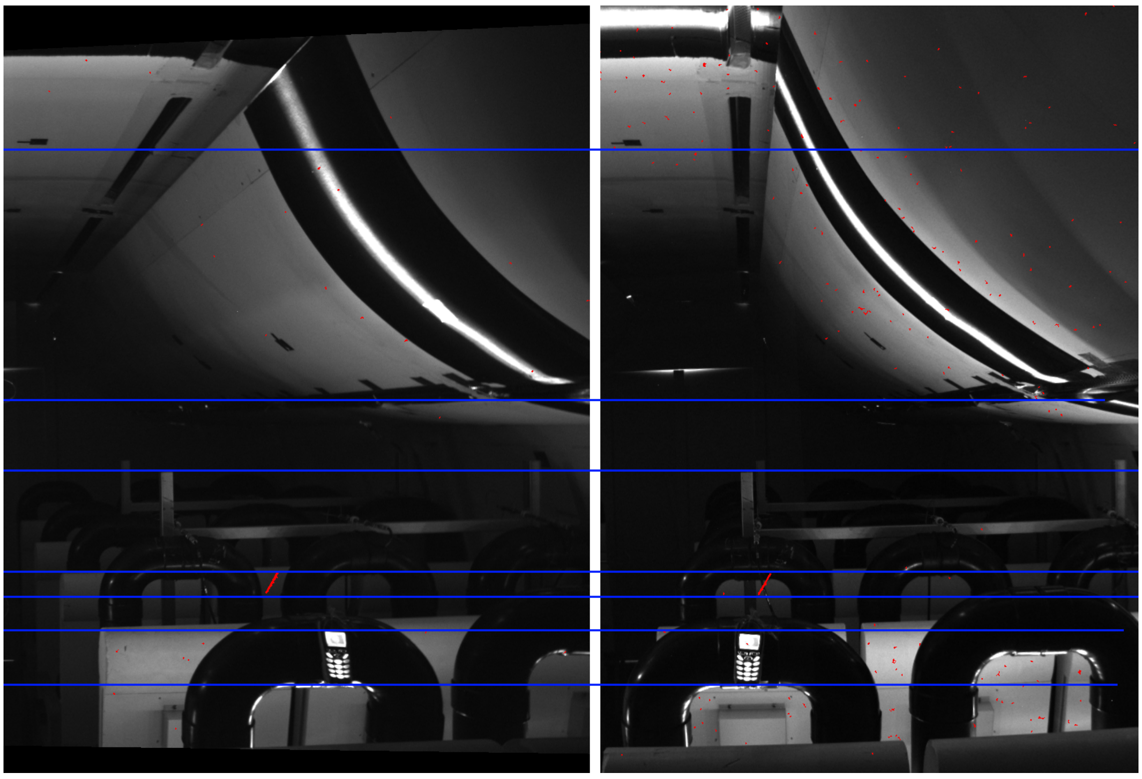


Figure 3.9: A pair of images after the application of the rectification transformation T_1, T_2. Red lines have been added to highlight how corresponding pixels share the same y coordinates.

3.6 Triangulation

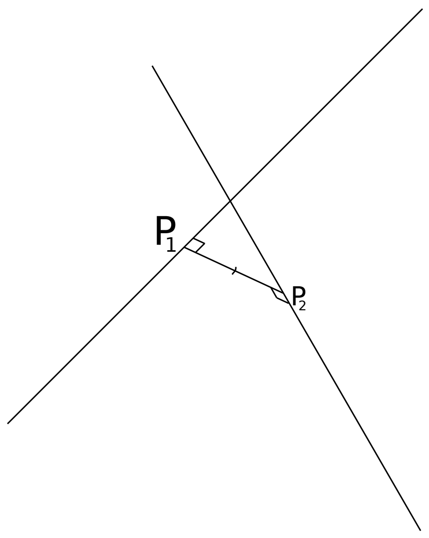


Figure 3.10: The situation of 2-view triangulation: skew lines in 3-dimensional Euclidean space

We describe here a situation which, though straightforward, is central to stereo reconstruction, and which therefore merits a detailed description.

Triangulation, as defined in [HZ04], is the act of using Euclidean geometry, specifically trigonometry, to recover a 3D location from two image points. A fully calibrated camera setup allows this kind of metric reconstruction, as explained in paragraph 3.2.3 and presented in the following few paragraphs.

Skew lines

Earlier (paragraph 3.2.2), we noted that one would ideally have the expression $P = (O_1 \wedge p_1) \vee (O_2 \wedge p_2)$ for the space point. In coordinates, that means that there must be scalars λ_1 and λ_2 which multiply the unit vectors representing the optical rays such that they meet again in P.

$$\begin{cases} P = O_1 + \lambda_1 \vec{u}_1 \\ P = O_2 + \lambda_2 \vec{u}_2 \end{cases}$$

But that is nearly never true: in reality, whenever the epipolar constraint isn't fulfilled, the optical rays do not meet and one always observes skew lines.

No calibration is perfect, and point detection, for several reasons (given in section 4.3) also doesn't result in perfectly matched points.

Derivation of triangulation

Let us work, without loss of generality, in a world coordinate system centered on the left camera, such that $O_1 = 0$ and $O_2 = \vec{T}$. The length of the translation vector $\vec{T}$ is called the baseline of the setup.

Let A and B be the (pinhole) camera coordinates of the points, obtained by projectively projecting the point P in the left resp. right coordinate system:

$$\begin{aligned} A &= [1|0]\Pi \\ B &= [R|t]\Pi \end{aligned}$$

The usual way to impose a solution is to find the point of closest approach on either optical ray and then use the midpoint, which incidentally minimizes the sum of squared distances to both rays. This approach extends to the n-view problem n optical rays and gives rise to Bundle Adjustment ([HZ04] p. 437).

One must minimize a function which is quadratic in the unknowns λ_1 and λ_2 (multipliers of the unit vectors $\vec{u}_1$, $\vec{u}_2$ describing the optical rays, named as in [DH73], pp. 398ff), which admits a closed-form solution:

The condition of mutual closest approach is met when the Euclidean distance

$$||\lambda_1\vec{u}_1 - \underline{\mathrm{R}}(\lambda_2\vec{u}_2 + \vec{T})||$$

is minimal. Optimize the squared distance function instead:

$$\arg\min_{\lambda_1,\lambda_2} f(\lambda_1,\lambda_2) = ||\lambda_1\vec{u}_1 - \underline{\mathrm{R}}(\lambda_2\vec{u}_2 + \vec{T})||^2 \tag{3.24}$$

In [DH73], one finds the following expressions (mutatis mutandis)

$$\begin{aligned} \lambda_1 &= \frac{\vec{u}_1\cdot\vec{T} - (\vec{u}_1\cdot\vec{u}_2)(\vec{u}_2\cdot\vec{T})}{1-(\vec{u}_1\cdot\vec{u}_2)^2} \\ \lambda_2 &= \frac{-\vec{u}_2\cdot\vec{T} + (\vec{u}_1\cdot\vec{u}_2)(\vec{u}_1\cdot\vec{T})}{1-(\vec{u}_1\cdot\vec{u}_2)^2} \end{aligned} \tag{3.25}$$

ostensibly for the case of parallel image planes, but simple calculation shows that the general case, supposed above, works just the same (just rotate the vector B beforehand, which does not change its norm).

With the help of triangulation, it is possible to pass from the point in space to its projections and back – again, as in [DH73]:

$$\hat{P} = \frac{1}{2}(\lambda_1 \vec{u}_1 + T + \lambda_2 \vec{u}_2) \tag{3.26}$$

From space to views and back

As we said above, 3D reconstruction by triangulation can also be applied in the case of non-matching points. The point $\hat{P}$ above, as a function of $\vec{u}_1$ and $\vec{u}_2$, is a continuous mapping from (normalized) image coordinates *onto* the reconstructed space and a continuous, injective mapping from world coordinates onto the slice (linear subspace, in fact, because of the Epipolar Constraint) of image coordinates defined by the epipolar constraint. Both mappings are smooth maps, so it can be treated within the framework of differential geometry, which is useful for calculating how errors in the estimation of points propagate.

Sections 7.1 and 7.4 show how we used the smooth relationship between the two spaces to map regions in space to the "uncertainty" encountered when measuring there.

Pixels to distances

The above procedure is not quite complete; for the units to check, one needs to transform the pixel coordinates into world coordinates. This is done via the "intrinsic parameters" $\underline{A}$ matrix from calibration, which contains the conversion.

3.6.1 Critical configurations

Having introduced the geometry of stereo images, we can move towards the determination of its implications for the geometric precision of the reconstruction.

Let us anticipate a little what lies ahead and postulate that for the interpretation of the images, space curves are going to play a central role (the definition of a curve is standard in differential geometry; for reference, ex. [BG80]).

Specifically, there are questions of interest like: how does a three-dimensional parametric curve map onto the individual views?

3.6.2 What happens to trajectories under projection

The trajectory of a particle is modeled straightforwardly as a k-times differentiable curve parametrized by a time parameter t. This meshes in with the “velocity field” view: a pathline is nothing else than such a curve, determined by the differential equation 2.1 on page 15

The image under projection of a curve need not have the same degree of differentiability. For instance, a smooth curve in a three-dimensional “scene” can be tangential to an optical ray.

By a simple calculation, one sees that a necessary and sufficient condition for the vanishing of the derivative of the curve $\vec{\mathscr{C}}$’s projection (expressed in image coordinates) is indeed

$$\begin{aligned}
& \frac{\partial}{\partial t}(h \circ \vec{\mathscr{C}})(t) = 0 \\
\Leftrightarrow \quad & \sum_i \left.\frac{\partial h}{\partial x_i}\right|_{\mathscr{C}(t)} \frac{\partial \mathscr{C}_i}{\partial t} = 0 \\
\Leftrightarrow \quad & \begin{pmatrix} \frac{1}{z} & 0 & \frac{-x}{z^2} \\ 0 & \frac{1}{z} & \frac{-y}{z^2} \end{pmatrix} \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{pmatrix} = 0 \\
\Leftrightarrow \quad & \text{both} \begin{cases} \frac{\dot{x}(t)}{\dot{z}(t)} = \frac{x(t)}{z(t)} \\ \frac{\dot{y}(t)}{\dot{z}(t)} = \frac{y(t)}{z(t)} \end{cases}
\end{aligned} \tag{3.27}$$

The image curve has a singular point when the motion goes in the direction of the image point, which is also geometrically clear.

Why is it a problem?

Generically, the situation does not occur: sloppily put, the tangent directions of $\mathscr{C}$ describe a (one-dimensional) smooth curve on the manifold $SO(3)$ of orientations in 3-space and this has zero probability of hitting a specific direction.

However, as the curves in question are smooth and the projection is smooth also, one should expect divergent behavior as the direction of motion nears the direction of an optical ray.

This might amount to a serious problem in the form of a systematic error, because of a systematic failure to detect motion in some directions. Some of the crude point detection methods initially employed in the prototype 4 on page 61 indeed had a *systematic error* (7.1 on page 143) which stems from their inability to accurately detect motion in such directions.

With three cameras, the problem is alleviated but in a binocular system, say, for a revised version of the system, one must eventually try and improve the detection method.

Motion inside an epipolar plane

Appeal to imagination: consider the idealized case of a particle moving inside an epipolar plane; suppose that the only information available is geometric in nature, by which is meant that the image curves can be extracted but no surface features can be matched.

If a particle's motion during a frame took place inside an epipolar plane, the uncertainty in reconstruction is necessarily very large.

The curves' pre-images intersect in a quadrangular piece of the epipolar plane. Over the affected frames, its velocity and direction within the plane is *entirely unclear*. Such data can only be "rectified" globally. Exactly the same can be said for the time information missing because of low frame rate (which is at the root of some of the problems encountered in the later chapters)!

In curve matching, the uncertainty becomes large in such a situation: In their paper [PP91], Porrill and Pollard carry out stereo matching by considering curves, specifically points where these are tangential to epipolar lines, and find that the "perpendicular error from the epipolar" depends linearly on the radius of curvature – thus it approaches infinity for a straight line.

3.7 Conclusion

Chapter 3 brings us one step closer to the goal. It introduces the stereo vision framework which enables us to conduct efficient depth reconstruction and to estimate the accuracy of the spatial reconstruction.

It forms an independent part within the software design, the core on which the reconstruction and interpretation steps rely (the image processing is exchangeable while there is fundamentally no alternative to stereo geometry. It is simply a necessary part of any 3D reconstruction algorithm).

We will use the camera model along with triangulation and rectification algorithms in chapter 5, where we will perform the synthesis of the current chapter and 4.

Chapter 4

Segmentation & feature extraction

Only frames with fairly long exposition were available at first, during which particles have time to describe complicated trajectories.

This makes standard methods for motion estimation, for example optical-flow-like methods ([Hor97]) difficult or impossible to use.

Nevertheless, we will discuss theoretically, and demonstrate empirically, how to employ image processing techniques to detect certain spatio-temporally well-localized events and proceed from there.

Actually following and reconstructing the trajectories is the subject of chapter 5; the current chapter will describe the pre-processing steps used for locating the candidate points for stereo-matching on the images recorded by the individual cameras.

To this effect, the following aspects are to be introduced:

- an observational model, especially its geometric and signal-theoretic aspects
- the creation of synthetic test images via photorealistic computer graphics
- the pre-segmentation of image regions potentially corresponding to traces
- the detection of 2D or rather 2+1D keypoints.

We start off with the formulation of a model of the image series, which is based on the theory of digital signals (because a camera is a device whose output is a digital signal).

Then we use the model to generate some synthetic images, which will also serve as test cases for our algorithms.

Next, we compose the basic image processing algorithms for extracting particle traces according to the model of section 4.1.

Because many existing PTV methods segment particle traces with pixel and region based methods, we discuss these first and then show how these fail to capture the essence of the observations (mainly because observed luminance varies a lot depending on the location of the particle). Luckily, we could recycle the segmentation techniques for generating masks (the particles only cover a small part of the image). These masks contribute greatly to the efficiency of the process, since one can avoid processing useless background information.

Lastly, we derive from the observational model two methods for extracting keypoints from the image series. Both methods have a similar output: both are by design, capable of determining the tangent direction of the projection of the trajectory onto the individual view.

Sub-pixel localization is achieved in both cases using Newton's method.

For the whole image processing part, care was taken to incorporate the scale-space paradigm [Lin94] – one may not neglect to choose the right scale in image processing. Some statistics relevant to scale selection are shown among the experimental results (7.2.1 on page 149).

4.1 The observational model

To better construct, compare and evaluate image processing methods, we propose a model of the frames captured by the cameras during the experiment.

The raw data on which our system was tested consists of series of graylevel images of a scene grabbed synchronously from a pair of cameras

At first, it appeared to be fruitful to treat the data as a function which is directly interpretable via finite differences in time, so as to capture the moving blobs (an actual technical term: [Lin94]) and their motion patterns.

Time series and temporal sampling

Basic knowledge of digital signal processing (e.g. the first few chapters of [Bra03]) is presupposed.

One can regard a whole time series $I_i(\vec{x},t)$ grabbed from a camera as a 2+1D function $L(\vec{x},t)$ which is integrated over a certain lapse of time and sampled at regular intervals,

$$I = ш * (L * Rect_{\Delta t}) \tag{4.1}$$

Where $Rect_{\Delta t}$ is the rectangle function $Rect_{\Delta t} = \frac{1}{\Delta t}(\Theta(t+\Delta t) - \Theta(t))$.

If the integration happens over a sufficiently small interval of course, one can use $L * R$ as a good approximation for L itself, since the limiting case for a very narrow rectangle function $Rect$ would be like convolution with a Dirac delta, i.e. with no effect at all.

Long exposure times

The effect of long exposure times is called motion blur – when the frequency used for image acquisition is so low as to stretch the above approximation (of the “instantaneous” distribution by its integrated version) too far, one gets into trouble when trying to interpret the images. An arbitrary number of things may have happened and left their traces, resulting in a very bad approximation!

This wouldn’t be so bad if the sampling frequency wasn’t linked to the duration Δt of integration. So with slow cameras, one badly sub-samples that integrated function, and it becomes a problem because important information is lost.

Focus and point spread function

The pinhole camera model does not account at all for the effects of focus(ing). While in principle the blurring which occurs as defocus can be used as a clue to retrieve depth information [FS07], we do not attempt to use it in this work, not only for want of a usable model describing the unblurred image – but since the effect of defocus is a broader PSF, it means that when the aperture is not very small, traces will look different according to location, signal to noise diminishes when they are blurred.

(The effect of a finite aperture on the image is emulated in the synthetic image series (4.1.5 on page 69). It did not play a large role in actual experiments, though.)

Content of a single frame

We describe shortly the model for a single frame, warning that one should finally regard the frames as part of a holistic, also temporally coherent whole, not as a disconnected bag of tidbits, so as not to lose important structure while interpreting.

Above, we wrote $L(\vec{x},t)$ as a function of a continuous parameter $\vec{x}$, since we model the image information as being spatially coherent, i.e. with correlation between (neighbouring) intensities.

Frames $\{F_{i,f}\}$ captured by the camera number i at frame number $f \in \{0,\ldots,f_{max}\}$ are images obtained by the camera's sensors integrating from $\frac{f}{framerate}[s]$ to $\frac{f}{framerate} - gap[s]$ along the time axis. The gap during which the sensor is "not exposed" should be as short as possible; it is always disregarded in the remainder of the text.

In our setup, the camera positions are fixed with respect to an immobile scene. The background's overall brightness may vary, because of oscillations in lighting intensity (indeed, fluorescent lamps might serve as light sources), which flicker at a frequency usually out of tune with the imaging system. This should not throw off our system, and indeed in the experiments it didn't, even without more sophisticated background subtraction.

$F_{i,f}$ is assumed to be composed of background $BG_{i,f}$, overlaid always *additively* with noise $N_{i,f}$ (an random variable for each pixel), and an image of particle traces $T_{i,f}$, consisting of the visible images of particles.

All other spurious information, such as residual background, is assumed, for the time being, to be subsumed under one of these labels.

Noise can often be modeled as being independently distributed, but there are some situations where its structure may play a detrimental role. We will also pretend that the useful information is also *added* onto the background, while in reality the visible particle traces are semi-opaque. This works especially well because intensities cannot be interpreted, and only the geometry is desired.

4.1.1 A model for particle traces

3D trajectories

We can start off with a formulation of the trajectory of a particle (say, of its center of mass to make it well-defined) through the volume of fluid. For this trajectory, we demand C^k differentiability with $k \geq 2$, so one can calculate velocity and acceleration at each point, cf. paragraph 3.6.2 on page 57. This will be used later, from chapter 5 onwards.

The goal of the image analysis modules is to allow a satisfactory reconstruction of these trajectories by building them from the bottom up.

Practical trace description

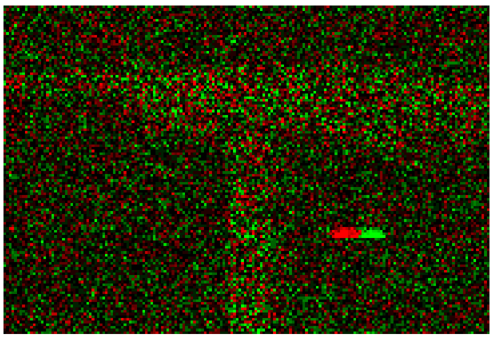


Figure 4.1: Region from a difference frame of *V4* series, showing a trace and some residual background structure. For a color legend, refer to 8.7 on page 197

$$\mathscr{D}(t_a+t) = h \circ \vec{\mathscr{C}}(t_a+t) = \mathscr{D}'(s) = \vec{\mathscr{D}}(t_a) + \frac{\partial \mathscr{D}'}{\partial s}(0) \cdot s + \frac{1}{2}\frac{\partial^2 \mathscr{C}'}{\partial s^2}(0) \cdot s^2 + O(s^3) \quad (4.2)$$

In most points, the projected curve can be approximated like 4.2 ($\mathscr{C}$ is the space curve, h the de-homogenization, $\mathscr{D}$ the projected curve and $\mathscr{D}'$ its unit-speed version (parametrized by arc length, since t is unobservable in practice).

Note that this whole description also has an inherent "scale"; small bumps occurring randomly in the trajectory are not very interesting in the broad picture and one can assume fairly smooth flowlines.

Image of a particle trajectory

Visibility To say that an object is visible means that its presence changes the image in a way that statistically significantly differs from noise and other features.

The visible trail of a lit, non-occluded particle will be referred to as a *particle trace*. In the handling of many subtasks (4.4 on page 85, 5.3.4 on page 108), we refer to a certain model for particle traces, which is given in the next paragraphs.

At this point, why not simply threshold and skeletonize? One can certainly try pixel and region based techniques like adaptive thresholding, pixel neighbourhood morphology with hysteresis (to be found in the next section) but separating T from noise reliably and for different images seems impossible.

Occlusion Not all of the ambient space is visible under experimental conditions; there are places where nothing can be measured because there are occluding objects in the scene. The real image planes of the camera are likewise not infinite.

The occlusion function $occ_i : S \to \{1;0\}$ describes whether a location is visible from camera i. It evaluates to 0 outside of the region covered by the views.

In a scene where the only thing that moves is the fluid, it does not depend on time. When a particle is occluded from view, or vanishes, or equivalently appears (the data looks the same forward & backward), we do not wish to make a wrong measurement but exclude the last bit of trajectory from further processing, because the velocity cannot be ascertained (the exact time of vanishing is not known).

Occlusion of particles by each other is a different question, addressed below.

PSF in practice

Every imaging system in existence is subject to finite blur because no physical system can transfer arbitrarily high frequencies faithfully (cf. linear image formation model in chapter 2 of [Köt07]). The impulse response of the imaging system (modeled as a LTI system) is known as its *point spread function* (PSF). Strictly speaking, there are different definitions of a PSF, most excluding the variation with focus, so that one can really represent it by a single LTI filter.

We could try and simplify further by subsuming any influence which causes the image of the particle's centre of mass not to be a point under the label "PSF", point out that no details smaller than the "apparent radius" ρ of a particle's "instantaneous" image play a role, except as noise.

The situation can be approximated with a PSF in the sense of [SK05] (also in [Ste08]) if one is willing to forgo the variation with depth.

Our "PSF" definitely depends on the depth map, though luckily not so starkly as to require handling: the detection of key features turns out not to be very scale specific.

We propose here one possible formulation of the model generating the $T_{i,f}$ ("foreground") image: l enumerates the distinct particles; particles are opaque to each other, as is empirically verified on images where particle traces cross.

We decide that opacity of particles with respect to the background image can be ignored.

$$T_{i,f} = max_l \int_{t_f}^{t_{f+1}-gap} k_i(h \circ P_i \circ \mathscr{C}_{\updownarrow}(t))dt \cdot PSF \tag{4.3}$$

P_i is the perspective projection onto the i-th view, k_i a reflectance factor $k_i = 0$ where $occ_i = 0$.

The reflectance factor k_i is related to the BRDF (bi-directional reflectance distribution function, [Kaj86, Hor97]), which is integrated over the whole visible surface of the sphere (because details are not discernible in our analysis of the situation).

One can read from the second-order term of the arc-length parametrized curve (4.2 on page 65) κ, the local curvature, which may play a role in the localization afforded by certain detection methods.

4.1.2 Remark on background frame

A "background template" might be obtained by observing the blank scene for a while, and then calculating e.g. a pixel-wise median image.

There are two kinds of effects which make the "background" intensity in a frame vary non-linearly with the "background template". First, sensor properties like saturation and a nonlinear transfer characteristic (see properties of sensors in section 2.5). Second, it is conceivable that lighting varies in such a way; reflection from surfaces usually behaves linearly, but there is generally more than one light source.

Synopsis

$$F_{i,f}(\vec{x}) = \lambda \cdot BG_{i,f}(\vec{x}) + N_{i,f}(\vec{x}) + T_{i,f}(\vec{x})$$

4.1.3 Two-frame difference images

Since our above model prescribes that useful information will be more-or-less overlaid over background information, it must be very well preserved on difference images while the background is greatly reduced. Indeed, the consecutive-frame difference images, which appear in this chapter, stem from the delusion of trying to capture the motion patterns by differentiating – not everything can be seen on them: no motion can be recovered when a particle moves along an optical ray.

Further exploiting the time series property, all further processing in the system prototype starts with difference images (probably a mistake, in which we ignored the integration in the sampling process).

Let us call $n_{i,f} = N_{i,f+1} - N_{i,f}$ the noise difference image. Noise variance ($N_{i,f}$ and $N_{i,f+1}$ being uncorrelated) will double; assuming i.i.d. Gaussian distributions for the pixels of $N_{i,f}$, the resulting distribution is Gaussian again, with zero mean.

Whether and how to combat residual background (after subtracting background frame) depends on whether we expect it to be confused with weak particle traces (see 4.3 on page 76).

4.1.4 Single difference images

If the background can be removed adequately, as discussed in the "background removal" paragraph, – after the subtraction, the resulting image is very close to a pure $T+N$, according to our model. This is, for several reasons more appealing than working on the difference images. The images which have been stripped of background also constitute a sensible time series, i.e. a 2+1D function, though a much less muddled representation than the 2-frame difference images.

4.1.5 Creation of synthetic images for validation

For the validation of algorithms performing image processing and especially interpretation tasks, one likes to refer to a *ground truth*.

Obtaining “ground truth” (or rather “air truth”?) in velocimetry measurements is notoriously difficult and demands ingenuity. Physical probes could be used, but they are not unproblematic (see paragraph 2.1.2). And there is the question of detection of particles. The only supplementary information available to determine the quality of detection on real images was comparison with manual annotations. To assess the accuracy of measured velocities, we employ photorealistic computer graphics to render a simile of real use cases. Such photorealistic images calculated from a scene description allow creation of tests in a fully controlled and parametrized way.

We argue that a way of creating artificial images which show known, and realistic, scenes, is a valuable tool for experimentation and validation. In summary, one can say that computer graphics proved a valuable tool for experimentation, and one which substantially shortened development cycles.

Realization

We use the versatile POVRAY software [oVPL], not free software for the moment but available for academic use.

It combines ray tracing and global illumination [Kaj86] techniques and accepts surface-based scene descriptions written in a Turing-complete scene description language.

It is not too hard to use a higher-level language and produce scene descriptions via “generative programming”. A direct incorporation of the generative model (background, lighting, reflecting particles) enables us to compare the model to the real acquired data (in terms of differing performance).

Modeling Ray tracing should be largely sufficient because global illumination particles are so small that diffuse lighting between them can be ignored.

Rendered scene images can be found in the appendix.

Camera geometry Ray tracing is a simplification of light transport based on geometrical optic (chapter 3) and the pinhole camera model 3.1 on page 28 is often the prime camera model used; generating images via ray tracing allows, but does not force, us to bypass the calibration process and to test some error sources separately.

Focal blur Focal blur can be simulated with a stochastic algorithm, which randomly perturbs light rays, which is also implemented in the povray package ([oVPL] in the software list). One can specify the desired aperture, focal length, number of rays and error bounds (low quality simulations of focal blur with this method looked grainy – a quality of noise not found in the real images).

Lighting Testing performance under different lighting situations is an easy task with simulated images. It is possible to simulate diffuse (ambient) lighting as well as the light slit from the original setup, which we did for the synthetic test images.

Noise models Artificial noise of arbitrary statistical properties can be put onto the synthetic images. Our test image series contain i.i.d. Gaussian noise.

Synthetic motion patterns It is perfectly possible to connect the scene generator to a fluid dynamics simulation programme (and use the scene geometry for both, etc.), or to enter standard fluid dynamics test data.

Our test series just contain a few particles, and no effort was made to simulate realistic motions, but there is no reason not to use extended test data for future experiments.

4.2 Segmentation of particles and mask generation

4.2.1 Background removal

Background can be eliminated by several methods. One approach is to capture a series of pictures before the experiments begin and calculate either an average or on a median background image. The average of a pixel can be obtained in linear time $O(n)$ w.r.t. to the number of images. In contrast, the median can be calculated in $O(n \log n)$ time, if an efficient sorting algorithm like heapsort is used. The average has the disadvantage of being more sensitive to noise than the median, but benefits from having constant memory requirements. The median's memory usage grows linearly with the number of images.

In addition to these two methods, it is also possible to simply subtract consecutive images. This is the fastest method, by far. It can be done using measurement data containing particle traces. As particles do not remain stationary, the subtraction will not eliminate them, unless other particles cross their trace in the consecutive frame.

4.2.2 Pixel exact particle segmentation

The first intuition, when faced with the problem of detecting and tracking particles, is to segment the particle as accurately as possible. To this end, several segmentation approaches have been employed and evaluated. First, a structuring element was used, before the segmentation method was switched to a Mean Shift based algorithm. We also evaluated gradient magnitude based segmentation.

Structuring element based segmentation

Our first attempt to segmentation was to use a structuring element on a thresholded difference image. The difference images contain the additive noise of the input images, which makes it crucial to find noise resistant methods of segmentation. The idea behind using a structuring element is that pixels belonging to particle traces generally have a higher connectivity than pixels belonging to noise. The following element $E_s$was used:

$$E_s = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

After using the element in an erosion operation, only pixel with four connectivity remain. Since the element operates on pixel basis, the detection of traces depends on how well they are aligned with the pixel grid. The required number of required calculations is constant with regard to the size of the image.

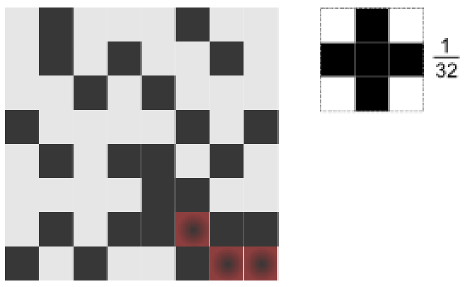


Figure 4.2: The structuring element applied to an image. Matches are coloured red.

Figure 4.2 shows the binary structuring element on the right, and its application on (random) data on the left. The used element tests for four-connectivity in order to detect particle traces. Assuming white noise, the odds for finding a false positive are $\frac{1}{2^5} = \frac{1}{32}$. Actual noise was found not to be white, partly due to imperfect background removal, partly due to camera intrinsic noise, as outlined in section 2.5 on page 22. Applying the element to measurement data showed that particle traces were often segmented incompletely. In some cases noise disrupts the continuity, leading to a single trace being segmented as several.

While the element is noise resistant, as shown above, it does not always segment the entire trace. Depending on orientation and intensity of the trace, only part of it may be segmented. Very narrow traces, with small horizontal or vertical velocity components, cannot be segmented at all. The first problem can be tackled by the application of hysteresis. Starting from the segmented trace, the intensity gradient is being followed until a minimum has been reached. All pixels on the gradient are added to the trace segment.

Hysteresis requires two thresholds. One threshold is needed to determine from which starting points the gradient is to be followed. The other is needed to set a minimum beyond which the gradient is not followed.

We calculated the hysteresis on the original non-thresholded difference images, starting from all points where the structuring elements detected a particle trace. The detection rate of this approach depends to a large degree on the selection of the initial threshold. This, along with the poor performance lead us to discard this method.

Gradient Magnitude

Pixels belonging to traces tend to exhibit a strong gradient in one direction, and a weak gradient in the perpendicular direction. We attempted to use this feature for segmentation by gradient magnitude. For every pixel in the input image, the gradient magnitude was calculated. For an image $f(x,y)$, it is defined as:

$$|\triangle f(x,y))| = \sqrt{(\frac{\partial f(x,y)}{\partial x})^2 + (\frac{\partial f(x,y)}{\partial y})^2}$$

Having generated the magnitudes, we applied a global threshold filter in order to generate the mask. The threshold value was chosen by calculating the average gradient magnitude over the image and adding one standard deviation. Masks using this method were quick to compute, while being very sensitive to noise. The threshold had to be adjusted for each experiment, and acceptable noise reduction came at the cost of major signal loss.

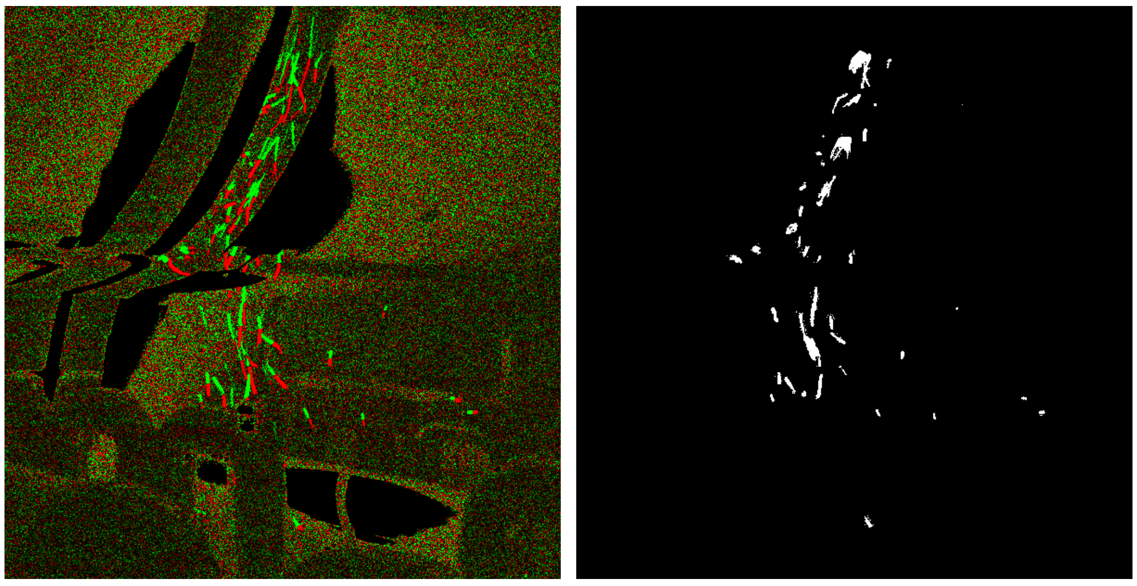
4.2.3 Mean Shift based segmentation

The third approach is based on the Mean Shift segmentation method first introduced by [CM02]. The Mean Shift is a non parametric technique for Kernel Density Estimation. It operates in arbitrary feature spaces. In this case, a joint feature space consisting of intensity and spatial information has been used by us. Using additional features, such as cornerness, would likely improve the segmentation result, but increase the computation time and were not tested for that reason.

Once constructed, the feature space is searched for local density maxima. This is done by calculating the local kernel density for an differentiable kernel. The gradient of the kernel density estimate is then used as estimate for the gradient density.

Calculating the Mean Shift vector for every pixel is prohibitively expensive. In order to increase performance, a pyramid representation, as described in [AAB$^+$84] of the images is used. For each increasing level of the pyramid, the input image is convoluted with a Gaussian kernel, before being downsampled. The meanshift segmentation occurs at the ẗopöf the pyramid. In order to move down in the image pyramid, towards the original image, the current level is upsampled and convoluted with the same Gaussian kernel. This is a lossy operation, of course, so a second pyramid is constructed, which approximates the lost information at each level. This is done by storing the difference between the image at each level of the pyramid, and the upsampled image from the next level, convoluted with the kernel.

After the application of the Mean Shift, the real valued feature space is digitalized through a low global threshold in order to create a mask for key point extraction.



(a) Original difference image (b) Mask image

Figure 4.3: Performance of the Mean Shift based mask generator with kernel radii of 8. Not all traces are distinguisable as separate segments, which has no bearing on the keypoint extraction. Some traces are partially segmented, making an dilation or blurring adviseable before the mask is used. The detection rate does not change significantly with different kernel values.

Summary

Erosion with the structuring element is a fast way of determining traces, but its performance is dependent on the orientation and velocity of the particle traces. The Meanshift method needs to construct an image pyramid in order to be useable on large images, the construction of which can lead to weak traces disappearing. Noise sensitivity can be controlled through the kernel sizes. The structuring element requires hystersis in order to segment entire traces. The thresholds for hystersis have a significant effect on the detection rate, and are difficult to determine automatically. While segmentation using the gradient magnitude is fast, it is difficult to distinguish between signal and noise using the gradient at each pixel.

4.3 Point feature detection

In this section, we discuss how to identify the projections of particle locations, namely *inter-frame* locations, from the image series. These 2D points are to be used as input for triangulation (of section 3.6 on page 54).

We explain why these were chosen and singled out as the fundamental starting point for reconstruction, among other possible choices.

Useful 2D point features consist not only in points marked on the image. Points enriched by supplementary information are called *keypoints*.

4.3.1 Why inter-frame locations

The choice of inter-frame locations is a natural one in long-exposure velocimetry because these are much more stable than any intra-frame choice of feature.

We argue that intensity maxima, for instance, cannot be used here, even though many of the PTV algorithms we encountered use them as a starting point. One reason is their instability, mentioned above, and explained in the following section; another reason is that in a 3D PTV application, matching and tracking are important and mere intensity maxima are not very helpful in the first task, and in our special application perhaps too *diffuse* for the second task.

4.3.2 Relevance to velocimetry

Given two series of exactly synchronous images whose exposure times are well known, it would be extremely advantageous in a velocimetry context if that knowledge could be put to use by linking some observations to precise timestamps. As we will see, this is also the starting-point for the next step of the reconstruction, in chapter 5.

4.3.3 Guideline for the design of detectors

[Can86] remains the eminent guideline for the design of image feature detecting operators.

1. Good detection
2. Good localization
3. Single response

If there were are criterion demanding specificity, we would think it justifiable to lighten it; the single response requirement, however, is central: scattered responses mean bad accuracy.

4.3.4 Of critical points and flowlines

This section portrays a technique which temporarily played the role of keypoint detection in our prototype, before it was replaced by the detector based on cornerness.

By employing differential geometry, one makes use of a technique which, far from being useful only for the limited task mentioned above and below, allows for continuous methods to work, and whose scope could possibly be expanded to encompass some of the steps of curve reconstruction.

Viewing an image as a Morse function has become a staple of image analysis [MK05, Köt07] and even offers an elegant representation of an image via Morse theory [RV06], which we do not exploit here even though it is extremely elegant, since decomposing the image into regions is not interesting in this special application.

The advantages of such a representation are manifest: it is intrinsic, whence invariant to affine transformations and rescaling of values (as can be easily shown). It traces an image along elements it really contains and not arbitrarily.

Continuous and differentiable image views

The beginning 21st century has seen a paradigm shift in image analysis from purely pixel-based methods, which were popular in the early days, towards methods which harness the powerful mathematical framework which deals with continuous functions or distributions [Bra03].

Though thorough care is necessary to ensure that differential operators are well-defined ([Flo97], which offers a very deep perspective), theories can be formulated which allow the analyst to abstract away the digitization (pixel grid, value quantization), except when its properties play a signal-theoretic role (as in Fourier transformation or for estimating the characteristics and effects of noise – which depend on the grid.)

For the purpose of analysis, a (band-limited, see [Ste08]) image will be construed as a smooth function. Indeed, the very notion of band-limitedness is dependent upon treating an image not as a set of pixels but effectively as a two-dimensional, continuous signal which has been *sampled* on a uniform grid.

This view of images as differentiable functions can be pushed very far. In this chapter, we make extensive use of spline image views [Mei09]. These allow the programmer to access an image at any subpixel position by means of a spline interpolation of arbitrary order. The gradient of a single-channel image is readily defined and can be evaluated at any position, at low computational cost ([UAE93a], [UAE93b]), once the view is built. Second derivatives are also crucial, since they tell a lot about the local structure (in terms of curvature), and can easily be calculated from the spline image view. It has become an invaluable instrument for the implementation of image analysis algorithms.

Critical points

A critical point of a differentiable function is a value of the parameters for which the function's differential vanishes.

A generic critical point [MSW63] can be described locally by the second order of the Taylor series, which is an expression involving the tensor of second derivatives of the function. For calculational purposes, the tensor is given by the second derivatives matrix (the *Hessian*) and diagonalizing it reveals the main curvatures (one for each dimension). If none of its Eigenvalues vanish, the critical point is not called "degenerate".

About genericity (for definitions, see [Lu76]), it has to be said that in images we often have functions with plateaux, which, strictly spoken, are special.

An analysis which is just as relevant is found in [KF01], which asserts that the scale space over a 2D image is rarely a Morse function, but that points with degenerate Hessians do show up generically.

The dia-scale behavior of spatial critical points

For an explanation of the action of scale space on spatial critical points, we follow [KF01]:

As one moves through the Gaussian scale space over a 2-dimensional image, scale points are found to coalesce; in the scale space over a generic 2D image, spatial critical points (i.e. critical points) do not only annihilate, but some are also born, while eventually all are reduced to just one maximum.

The points at which these events take place form a closed, measure-zero subset of scale space; consequentially, so do the scales at which this occurs and within an ε of the original scale no such event will occur. However just *at* the special scale, many critical points are born which can be of interest.

Application to particle locations

What is characteristic of our inter-frame particle locations? We would have liked to find a distinctive feature in terms of critical points of a functions. One advantage is manifest, the other two are more speculative:

- it is an intrinsic property of the image (or at least of its idealized, continuous version) – and invariant to affine transformations and rescaling of values
- maybe other properties like flowlines can be exploited
- perhaps the approach can be adapted to work directly on 3+1-dimensional time series or even higher-dimensional ones which seamlessly incorporate the parameters which control stereo calibration

To pick up a thread from the introduction (4.3.1 on page 76), where we argued against the use of inter-frame features like intensity maxima: it is perfectly true that intensity maxima corresponding to a single particle trace coalesce at some scale, it is not reasonable to expect that this coalescence will come at the exclusion of nearby particle traces, because unlike Weickert's [Wei98] anisotropic diffusion, the process leading to Gaussian scale space does not "respond" to image information, in a sense; neither is it clear, in the presence of occlusion and varying reflectance, that corresponding particles will match up in the sense of epipolar geometry.

The model formulated in section 4.1 led us to believe that normally the sought-for location lies on a zero level set of the difference image; section 5.2 of the next chapter shows a related property.

Isthmus creation

First of all, we pre-smooth the (difference) image function. That is because at a very short scale, sensor noise usually dominates useful information, as was confirmed experimentally. Squaring the smoothed image is tantamount to convolving its Fourier spectrum with itself; that process

- doubles the Nyquist frequency
- transforms *anti*symmetry into symmetry

This is merely a trick: squaring the original function makes all points on the zero-line into minima in at least one direction, albeit whether the zero line is straight or curved, the resulting minimum is always a degenerate one (except for isolated ones, which aren't interesting).

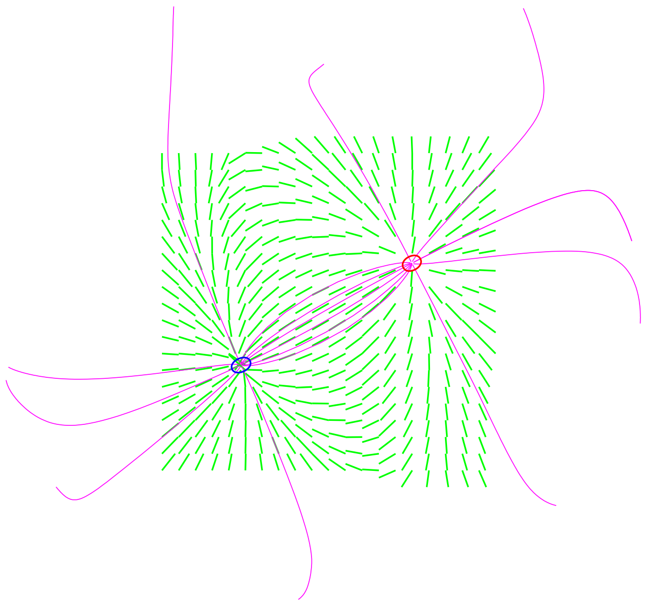


Figure 4.4: Critical points, gradient direction and some gradient flowlines of the original image at $\sigma = 5$. Legend: maxima are blue, minima are red and saddles are green.

The second trick consists in performing a minimal smoothing with a Gaussian of width ε.

Within an ε of the original-scale squared image, if there's a "concentration" of density (i.e. higher greyvalues), the Gaussian process will, intuitively, cause the higher density to diffuse (the smaller ε, the smaller the effective "locality") and lift the former degenerate minimum slightly, such that in one point, a maximum is created in the direction (to first order) of the zero line.

Saddles not corresponding to features will end up being revealed at a later step.

Gradient flowlines

Integral curves of the gradient field asymptotically end at critical points; from a saddle point, one can stably follow the directions of steepest ascent to end up at maxima or minima. Information gathered on the flowlines might serve to prune saddle points which do not correspond to the desired feature

In figure 4.7 on page 83, flowlines are colored according to the median intensity encountered on them (yellow=good). But one must be wary of the parametrization: in the experiment, we used a simple Forward Euler scheme, whose step size depends on the gradient intensity – this way, the statistics are skewed.

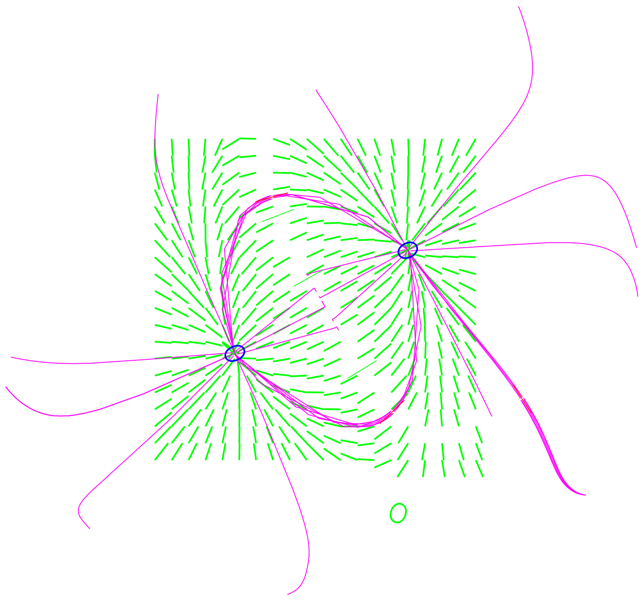


Figure 4.5: Critical points, gradient direction and some gradient flowlines of the squared image, without post-smoothing.

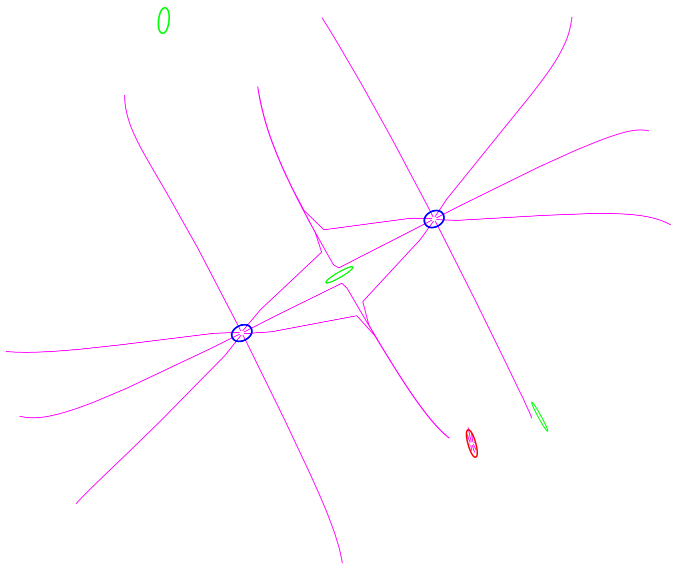


Figure 4.6: Critical points and gradient flowlines of the squared and smoothed image. A saddle is now detected between the maxima.

This idea, while presenting a certain potential for development, was shelved in favor of the approach described in chapter 5, which seemed more flexible: we will construct arbitrary curves first, then assign a score according to an exchangeable criterion.

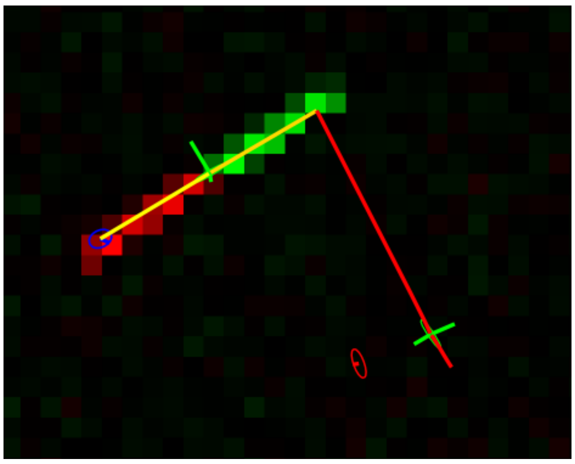


Figure 4.7: Flowlines followed from saddles, superposed on original difference image; same scale as above images. The color stems from an experimental homogeneity criterion, see below.

4.3.5 Importance of background intensity

With the synthetic images especially, there was a problem with oscillating overall image intensities, because of an erroneous normalization after the noise was added. This was remedied by subtracting the median value from the whole image and thus move the most frequently occurring value to zero. In the above test image (figure 4.8)), the median pixel value was $m = -2.535$ and the mean $\mu = -2.53$, the highest $max = 9.665$ and the lowest $min = -14.428$. After subtracting the me(di)an, $|max - m|$ are within 3% of each other (seems reasonable, since the single "particle" has the same orientation and about the same position relative to the light source below it).

Issues

Even if it is, in principle, present, the critical point sometimes (even if done with subpixel accuracy) escapes detection, which runs afoul of the philosophy of generate-and-filter. Furthermore, distinguishing bad candidates from good ones is very hard, because the good ones are weak by construction! (the surface is only relatively very slightly curved there). Following flowlines may be costly (simple Euler iteration, as used above, has bad stability; one needs adaptive-step Runge-Kutta methods [Mei09]).

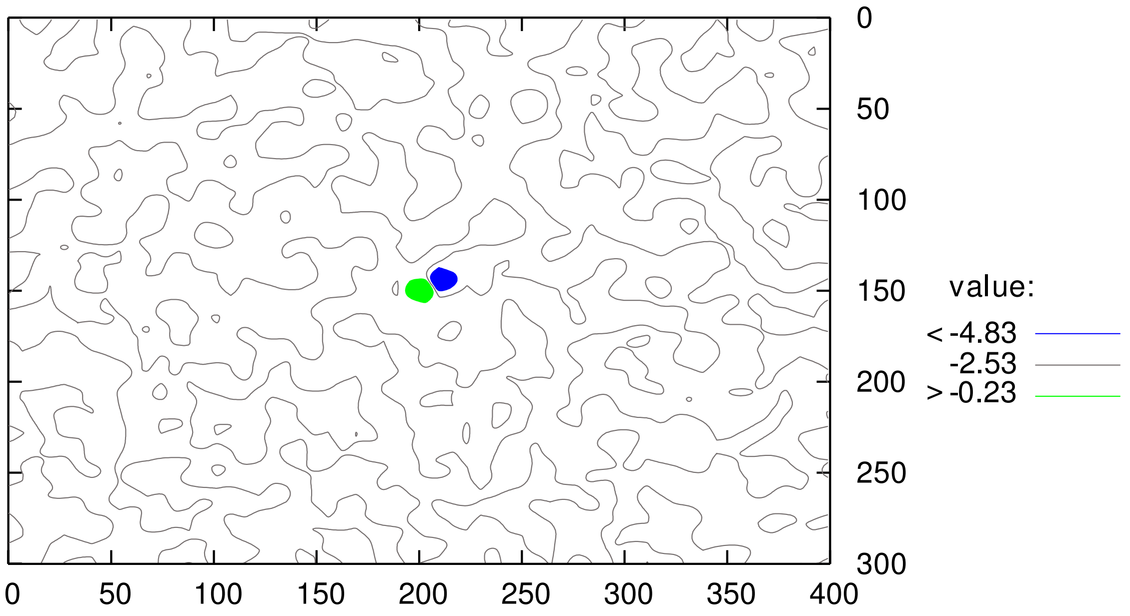


Figure 4.8: An image from *synth*1, scale $\sigma = 4$, zero level set, $<25\%$ of minimum, $>25\%$ of maximum after subtracting mean value.

Disturbances in illumination influence localization, which leads to frequent misplacement of feature points; worse, there is no right scale for the Gaussian filter: either one obtains a deluge of saddle points even in a very small region, or when surroundings are a little crowded, influence from neighbouring values spreads and a systematic error is introduced, which would have to be corrected later (cf. section 5.6 on page 119, or by descending in scale-space after the initial detection, see [KF01]).

The influence of κ (the curvature, defined in paragraph 4.1.1), and kindred parameters, in conjunction with scale, on localization, is again not easy to ascertain because the hunt for the critical point is a very local affair while Gaussian scale space is global in character.

Put on hold in favor of less capricious detector

The detection proposed above was very provisional (it worked on 2D difference images in the first place, which means that motions tangential to an optical ray are undetectable, but it could have been upgraded to a full 2+1D detector); instead of developing it further, for the time being, it is replaced by another detection method, described in the next chapter, which performs significantly better (in experiments, we see localization errors shrink to small fractions of pixels relative to manual localization).

4.4 Extracting keypoints by cornerness

Finding an acceptable way of reliably detecting keypoints, especially encompassing all varieties of particle motion, is a delicate matter. Fortunately, it turns out there is a common feature to all situations where a particle trace finds a continuation on the next frame.

In the last section, we considered two-frame difference images (and showed that they do not preserve all relevant information, especially in the case of motion along an optical ray).

In this section, consequently, we detect inter-frame locations not on difference images, but on *individual* consecutive frames from which the background has been subtracted.

4.4.1 Appearance of traces

Continuations of particle traces can occur in two ways: a motion in the direction of an optical ray results in a small blob-like trace on both frames and a motion with a tangential component must have a "line termination"-like feature (in the sense of [KWT88]) at the point which lies at the interface of both frames.

One could expect both situations to present a fairly high isophote curvature profile ([Flo97], p. 168ff), but such considerations were not specific enough and failed as a detector, since they were often met on the background, and neither did they lend themselves to a very accurate localization.

4.4.2 Region properties

One can distinguish image regions by their local differential properties.

It must be said that one is not interested in delimiting regions, as these are mere artefacts in this case (we modeled the trace as a "thick line") and the exact delimitation is without meaning – in fact, in our understanding, there are semantically no regions, because the "line terminations" and the "bright spots" are only blurred versions of the ideal geometrical image of the trajectory.

We settled on an approach which treats two consecutive frames separately, applying Harris' and Stephens' corner detector ([HS88], explained below) at an appropriate scale. In the next step, the results are combined (by multiplication), then local maxima are sought, so that a feature maximum is registered only where both images present "jointly maximal" positive cornerness.

That is to say, the peaks of cornerness suffice and segmentation is not required.

Cornerness

Harris' and Stephens' combined corner and edge detector builds on the structure tensor (in [Köt]), whose components are Gaussian averaged Gaussian derivatives.

$$M_{ij} = g_\sigma * (grad f \otimes grad f)_{ij} \tag{4.4}$$

It has many applications in image processing, and is also useful here. There are two scales of smoothing. The outer averaging is necessary because else M would be reduced to the tensor product of the gradient with itself, which can contain only intrinsically one-dimensional information.

A large determinant of the structure tensor indicates the presence of two-dimensional information like corners, as opposed to edges – the matrix being close to singular when the gradient varies little in one dimension.

Harris' and Stephens' detector is defined as follows ($k = 0.04$ being the most frequently used value):

$$R = det(M) - k \cdot tr(M) \tag{4.5}$$

Edges end up having negative response, and corners positive. Completely flat regions have zero response.

It is clear that one must choose appropriate scales (the target features are best localized at a rather narrow scale, but one wants to guarantee sufficient overlap for fast motions parallel to the image plane).

Relevance of the feature

The feature, as described here, is quite robust to difficult situations (as the results in 7.3 on page 153 show). Nevertheless, we feel there is potential for improvement.

The product of cornerness along the time direction, in the proposed role as a particle trace detector, has several advantages over other features:

- because of its smoothness, it manages to indicate corresponding ends of traces
- it also detects motion along an optical ray, i.e. a radial symmetric feature also gets a positive response, if k is chosen small enough.

There are some shortcomings in the current utilization of the feature which make it less specific than it could be. One should strive to find a better way than to multiply the responses

- the response can be quite weak, with low contrast
- “crossings” of traces also get marked, because the negative responses, when multiplied, also result in a maximum! The presence of an strong edge-like gradient also leads to a positive response when multiplying the contributions.

Specifically, it seems necessary to use a mask or find a criterion to filter keypoints.

Localization

Sub-pixel localization is achieved by interpolating the resulting image with splines (see 4.3.4 on page 78 and the references listed there) and searching minima with Newton’s method.

Recovering the direction

The “direction” of the feature can again be determined locally by examining image derivatives (for ex., reuse the Gaussian derivatives from the structure tensor so everything can be built from the same LTI filters).

We take derivatives on difference images directly as an estimate of the direction of motion (bearing in mind that a vanishing derivative means a motion in a special direction, which is the direction on an optical ray). The experiments found in section 7.3, agree with this view.

4.5 Conclusion

We have demonstrated empirically, and justified theoretically, how to employ image processing techniques to detect certain key events. These events are highly meaningful to the application – they represent locations of particles at well-defined instants – and serve as input for the following reconstruction steps.

The introduction of keypoints was convenient to get a handle on the extraction problem. In many ways, the bottom-up approach seemed most likely to succeed: it integrates well as a step of a reconstruction pipeline. We will see more of the advantages in the following chapters.

The drawbacks, where the use of keypoints instead of regions can be dangerous, are twofold. Firstly, features which are missed in the first step are impossible to recover afterwards. Secondly, it does not afford a straightforward extension to a non-synchronized experimental setup (which we will come back to in the very last chapter).

Overall, we come to the conclusion that the advantages outweigh the disadvantages because the use of keypoints is a significant simplification over methods based on spatially not well-localized features like regions.

4.5.1 Comparison of the detectors

We have presented in this chapter two methods for generating candidate keypoints for stereo matching. Both methods are not purely ad-hoc, but derive from the model described earlier in the chapter: both are based on local differential properties of images, but they differ in the kind of information used; while the first one uses second-order differences, the second is based on the Structure Tensor, which describes local image properties in a different way.

As for the first detection method (based on saddle points) proposed, it is not ruled out – but it would have to be modified: we are well aware of two related systematic problems with the use of consecutive-frame difference images:

1. While two-frame differences contain nice point features defined by local antisymmetry, these are subject to a shift which depends on orientation (in 3D) and brightness
2. Motions along optical rays, as discussed in section 3.6.1, cannot be captured at all since the differences may vanish there.

The second detection method presented is currently used in the software prototype, because it performs well in practice.

Chapter 5

Curve fitting and top-down interpretation

In this chapter, we will present our joint approach to the two subtasks of stereo matching and of tracking proper, which are combined to a way of extracting particle trajectories.

The approach of extracting inter-frame key locations and combining matching and tracking is, as far as we know, novel and not to be found in any prior publications.

Other sub-pixel particle-based techniques like [NDT04] use mere intensity maxima; many publications also invoke neighbour searches and extrapolation of trajectories for the purpose of tracking.

These courses of action become unavailable when exposure times are long enough to generate significantly, and randomly, curved traces (see notes on turbulence, 2.1.1 on page 13).

The chapter starts with an application of the first tentative keypoint finding technique from chapter 4 directly to reconstructed space, to show the possibilities and prepare for work in 3D and 3+1D.

It then moves on to present two kinds of curves which are used in many applications, ranging from 3D graphics to numeric methods for simulating physical processes.

The Bézier curves of section 5.3.3 might look like a diversion (in that they are not used in the final version of the prototype), but they led us to consider B-splines (section 5.3.4) which remained to be the useful representation of choice, and there are theoretical reasons behind the move to B-splines too (the chance to avoid high-degree polynomials, for one).

After defining the relevant curve spaces, we exhibit a graph-based representation (section 5.4), which allows for efficient handling of ambiguous paths constructed from keypoints.

The relevant data structures used for implementing this into software will be explained in the following chapter (under section 6.3).

Lastly, some remarks on energy functionals on spaces of curves (5.6 on page 119) which shall eventually help to determine to what degree trajectories are spurious.

5.1 Motivation for joint matching and tracking

Our top-level goal remains to identify and closely reconstruct particle tracks on synchronized, calibrated stereo image series, in order to gain insight into the motion patterns of a fluid.

From the basics of stereo geometry (chapter 3), one could easily infer the impossibility of matching point features with only two cameras, because of the inherent ambiguity in matching.

It is not like in scenes with structured and textured surface; the obtainable 2D keypoints are by no means specific enough to e.g. calculate feature distances and use these for matching.

Given a point, say p_L, on one image, the locus of points on the other image which fulfill the Epipolar constraint is a line (as has been discussed in chapter 3).

The problem is only exacerbated by imperfections in calibration which force one to extend the search to a two-dimensional window.

This means that there are many points on the other view whose features "fit the description", and there is no sure way of telling which one corresponds to p_L, or if *any* one of them corresponds to p_L.

Some PTV systems (mentioned under "Prior Work" and [BvdPK]) make use of extrapolation to determine where to search for position on the next frame. We did not find extrapolating over the duration of a frame to be helpful because significant changes in direction were possible.

Prediction would make sense in areas with mostly laminar flow, but not when there are turbulences at the relevant scale – and, since a known property of turbulence ([LL59]) is that it spans many scales in space and in time, this is indeed sometimes the case.

5.1.1 Proposed solution

Instead of dealing with single frames, we have decided to instead adopt a holistic view of the time-series data and describe the useful information contained in it in terms of entities transcending the single frame and the pixel and already closely corresponding to an interpretation of the data.

The data is thus described and evaluated in a combinatorial way, by forming 3D key-points, constructing a finite set of paths from them and keeping only the best ones, using geometrical criteria. The paths are converted into curves by finding their least-squares approximation with certain well-known and well-behaved basis functions.

5.1.2 Improved accuracy of velocity measurement

What can curve fitting do for the accuracy of measured positions and velocities?

It is well-known that higher approximation order means higher accuracy in approximation of smooth functions.

Many ideas in this chapter stem from the observation that not all possible trajectories are equally viable physically (or supported equally well by the observations).

And even if there are small, irregular motions, these are of no interest, so one should strive for a simple description.

5.1.3 The last word has not been spoken

There is much room for improvements, and this chapter is also an account of explorations which didn't find their way into the prototype in time (among others, the energy functionals of section 5.6). Nevertheless, most of the other algorithms have been implemented (including some simple, but reliable criteria for evaluating the quality of curves) and fulfill their function.

5.2 3-dimensional view

We already noticed in section 4.1 the multitude of descriptions appropriate to the problem at hand. This section serves as a demonstration of, so to speak, interchanging the steps of keypoint generation and matching. The images shown on the following pages lie in a three-dimensional space which is identical to the space viewed by the cameras: they try to represent all possible matchings between the individual views, in a sense explained below – this should guide our intuition on how the individual views combine in space.

All images were made with VTK [SML97].

Actual images The first few images (5.2 on the facing page, 5.2 on page 94, 5.2 on page 95 and 5.2 on page 96) show how the greyvalues of two individual views reproject into space and how two views might combine to a unified picture. A more general and information-preserving way to do this is certainly using the tensor product of the individual greyvalues; here, we just show product intensities.

The images underlying figures 5.2 on the facing page through 5.2 on page 96 are a synchronized pair from the *V4* series (described in the appendix). They show some background structure and a crowd of particles in the upper part.

Scale selection Compare the three images 5.2, 5.2 and 5.2. They represent three-dimensional contour plots of functions composed from the original images, as reprojected into space.

Legend: the axes are $10cm$ in length. The “Z” axis from the reference camera system, which points in a “depth” direction, is drawn in green. The rainbow colors stand for normalized greylevels.

“Z precision” In the pictures presented subsequently, one can observe the presence of very elongated regions corresponding to more round bright regions on the individual images. This is the effect of the baseline problem, which causes bad depth estimation where the optical rays are almost parallel, and which is discussed in sections 7.1 and 7.4.

One can clearly follow localization getting worse, but local structures disappear as one goes up the scale. Here, it seemed more appropriate to apply Gaussian blurring to the source images than to the 3D volume.

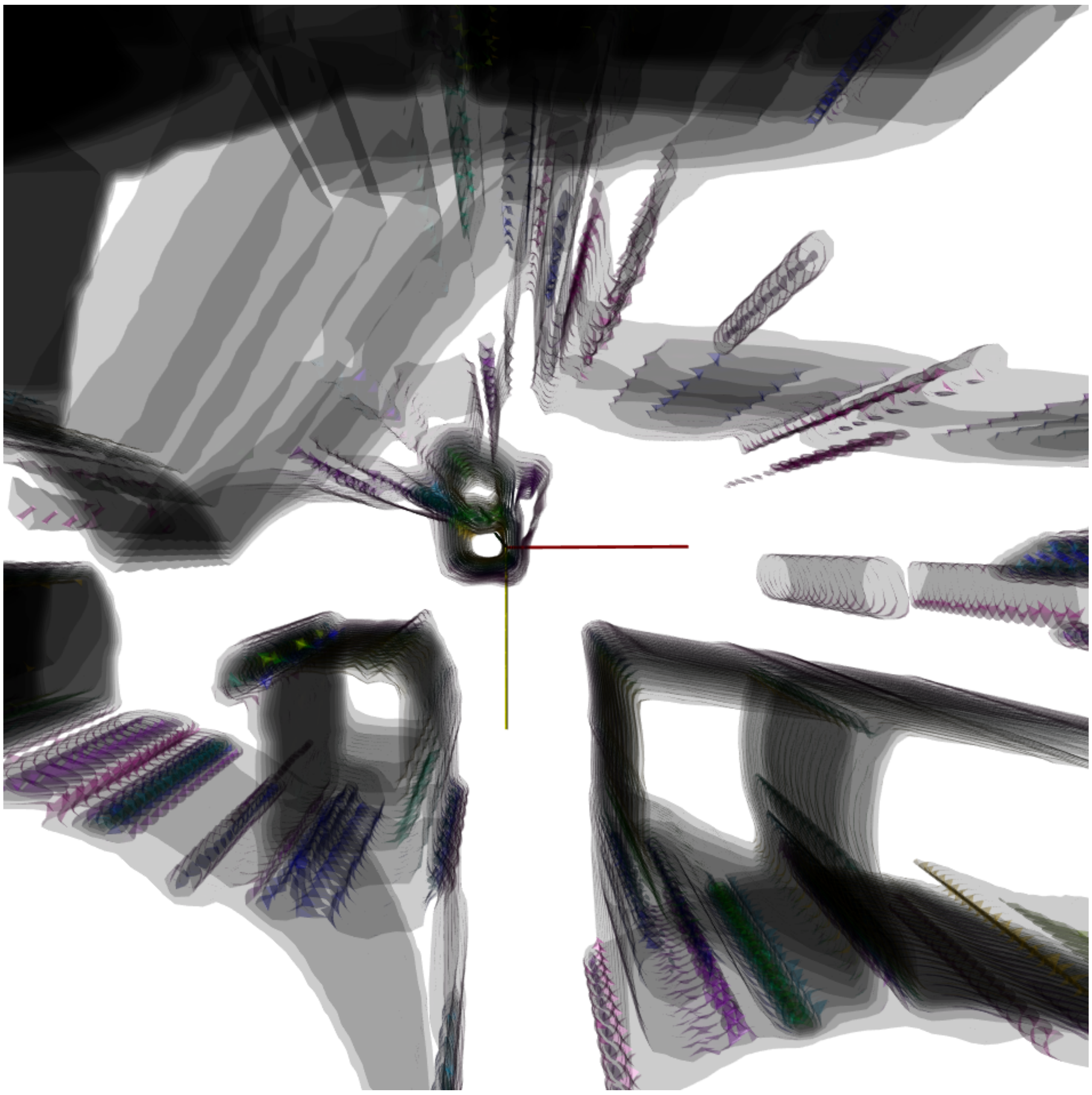


Figure 5.1: An image from *V4*: Ten level sets of the left image's contribution (clearly, they should be smooth & tube shaped, which they don't appear to be on this image – sampling artifacts during preparation of the illustration)

Search for critical points As in the 2D case, one can search for critical points. Gray-values of volume were pre-smoothed, and post-smoothed in the same way as in subsection 4.3.4, critical points then roughly searched using one step of Newton's method with finite differences (since there wasn't a 3D "spline image view" ready). The last three images (5.2 on page 100, 5.2 on page 101 and 5.2 on page 102) show the result.

Legend: color is according to Morse index: Minima in red, two kinds of saddles, maxima

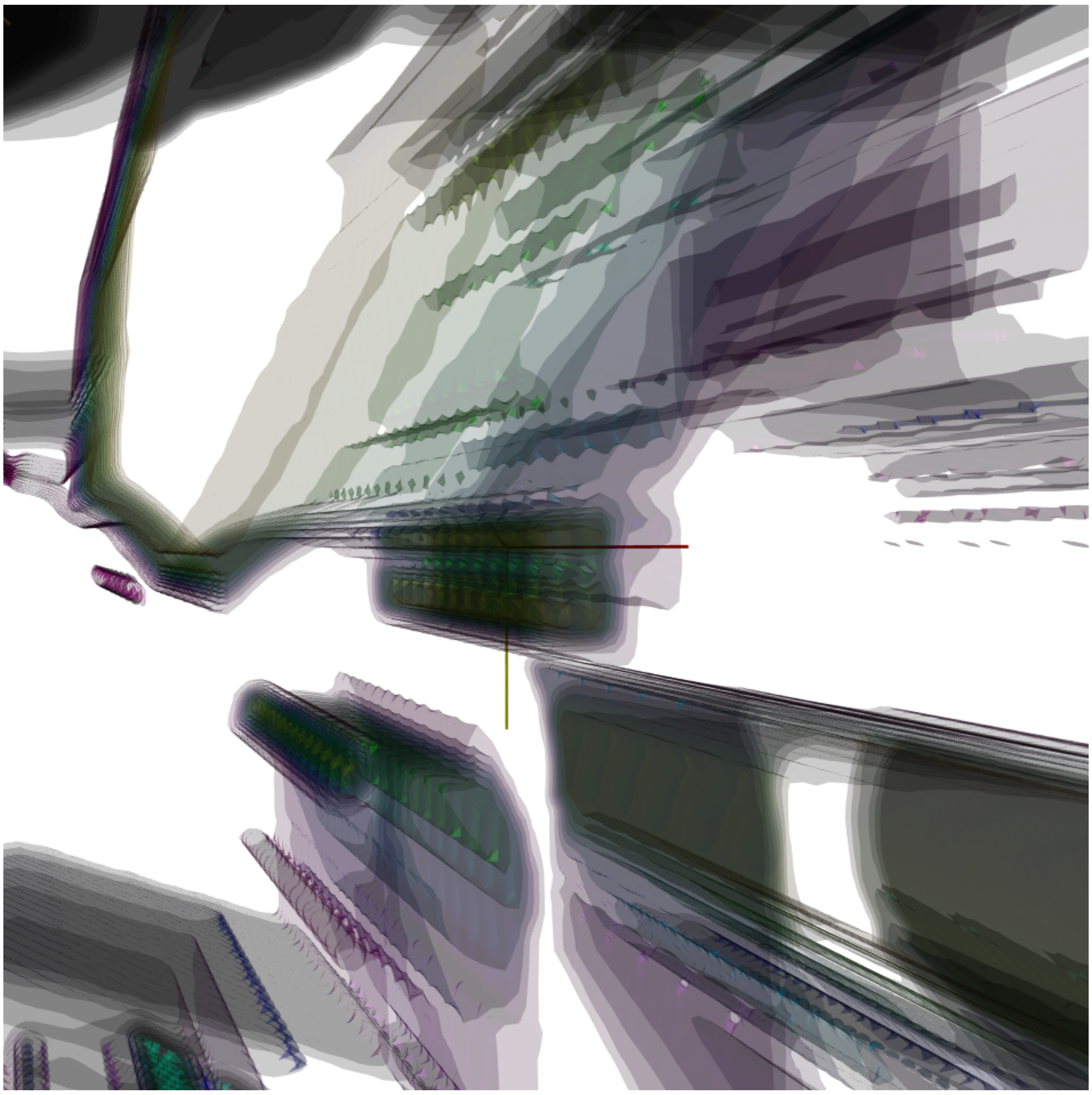


Figure 5.2: Ten level sets of right image contribution

in green. Shape indicates principal axes (the Eigenvalues have been normalized).

Again, the functions whose properties we show are products of greyvalues found upon reprojection, and serve as a crude indicator. One could replace them by something more useful.

Undistortion was not yet being done while creating these images, so (except for the synthetic image series, which do not suffer from lens distortion in the first place) matching is bound to be suboptimal because the calibration parameters are not perfect.

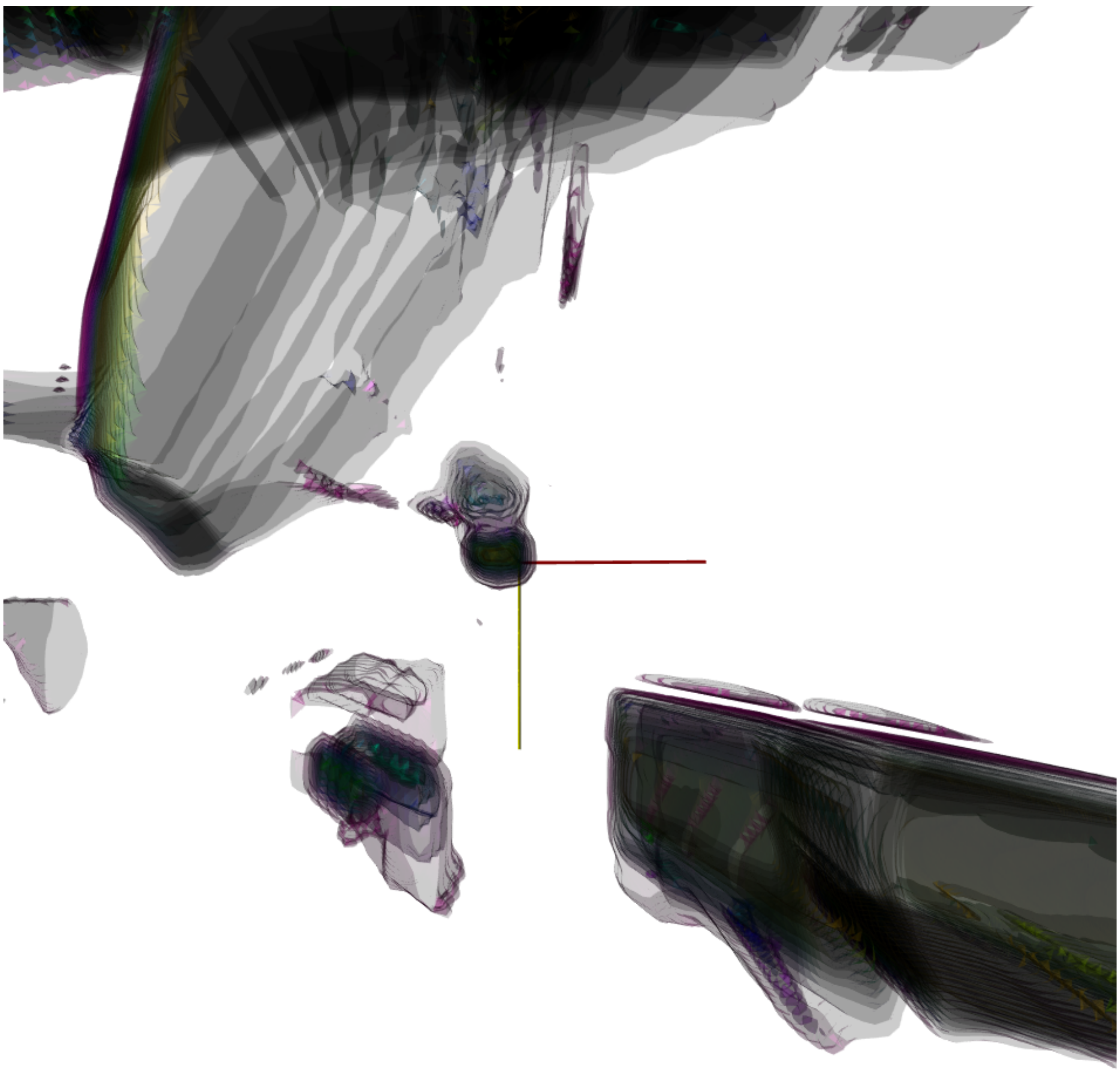


Figure 5.3: Product of the two. The box measures 64x64x64cm. The result seems to be dominated by the left image, even though both have been normalized beforehand

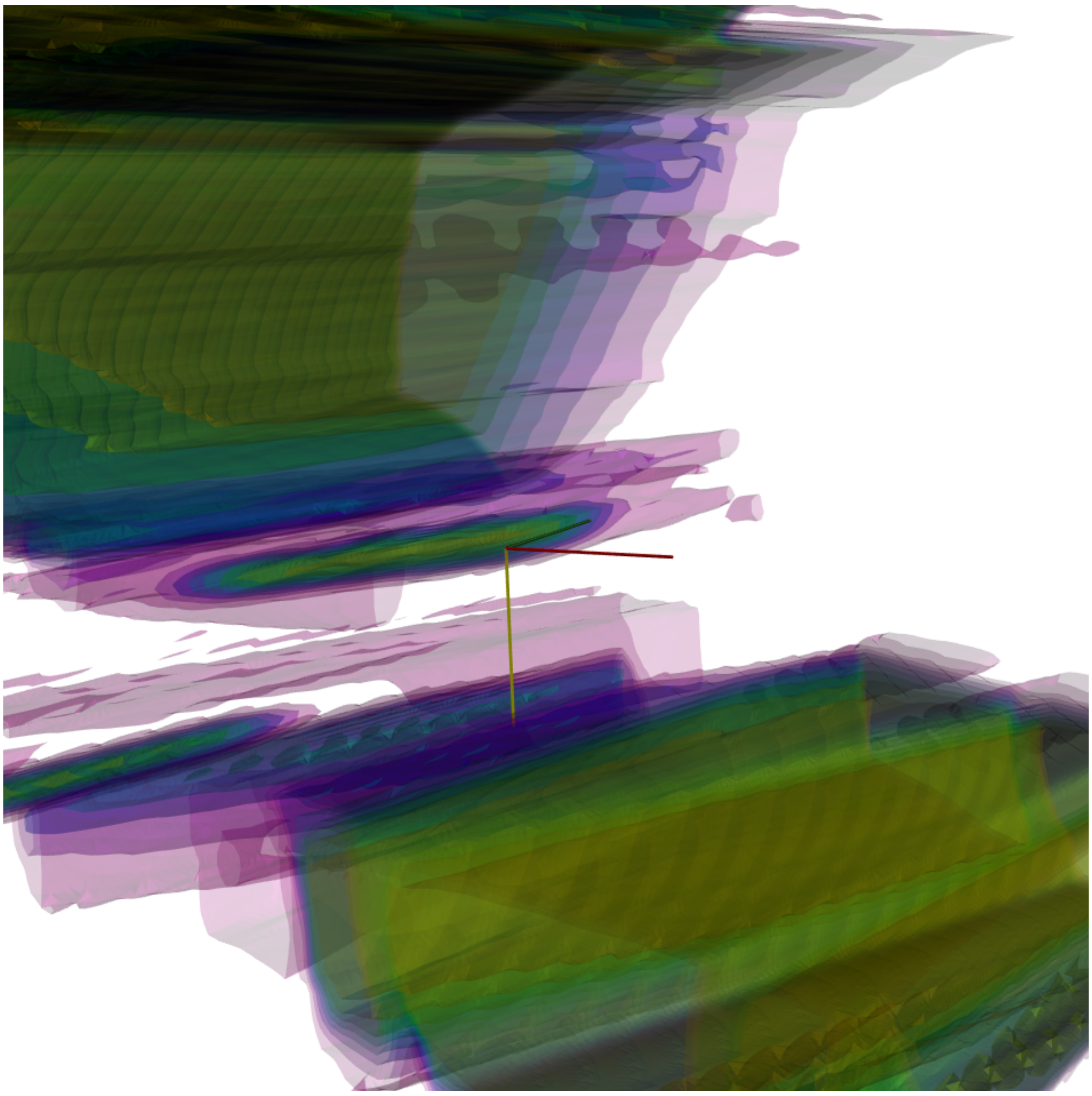


Figure 5.4: Same as previous image (side view). All the dimensions are scaled equally

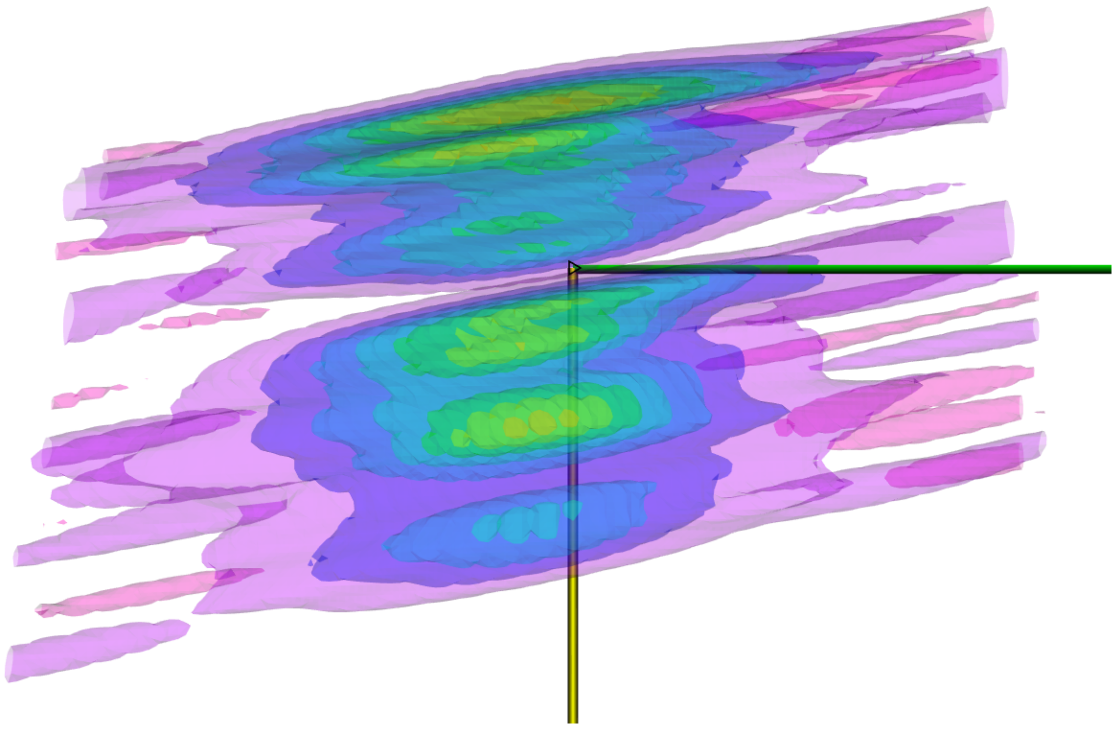


Figure 5.5: Pair of difference images from the *synth4* series, smoothed individually at $\sigma = 0.6$. This shows $|I_l(p_{xl}) \cdot I_r(p_{xr})|$

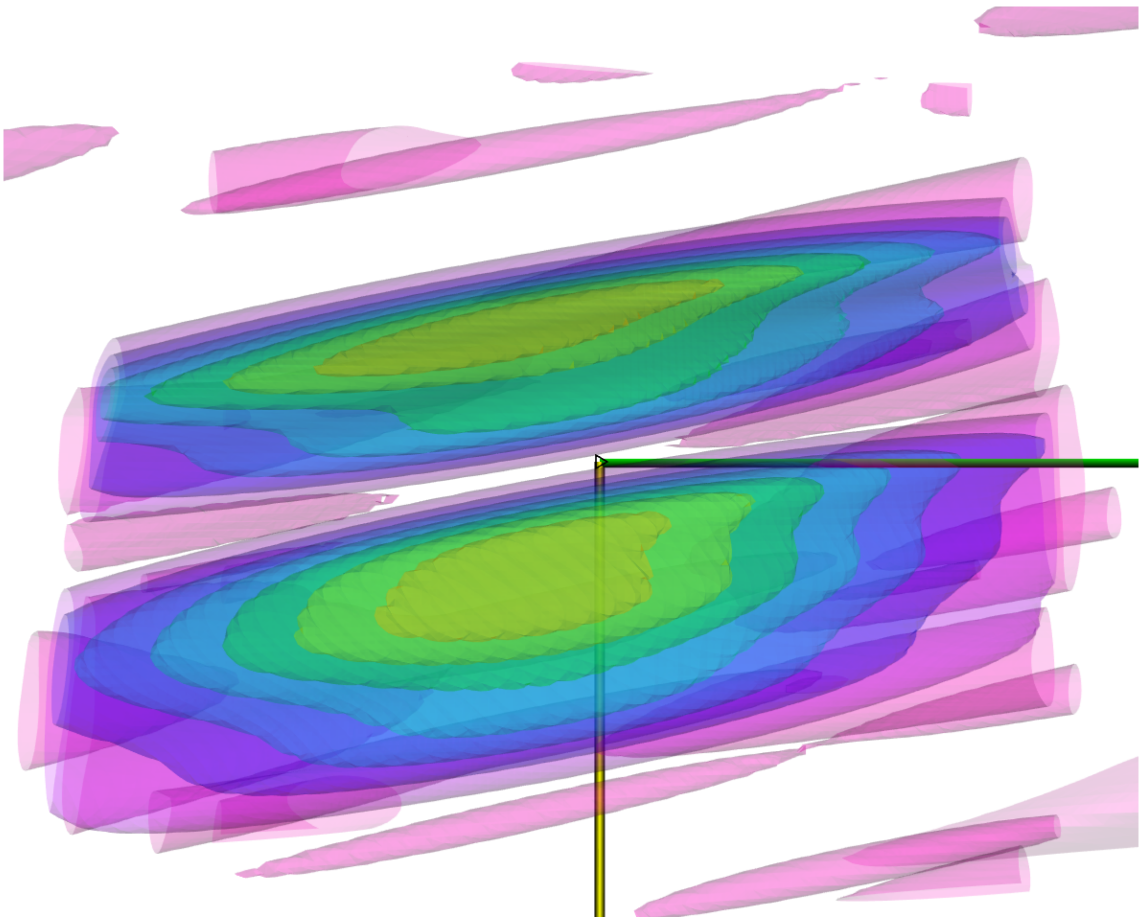


Figure 5.6: Pair of difference images from the *synth4* series, smoothed individually at $\sigma = 1.5$.

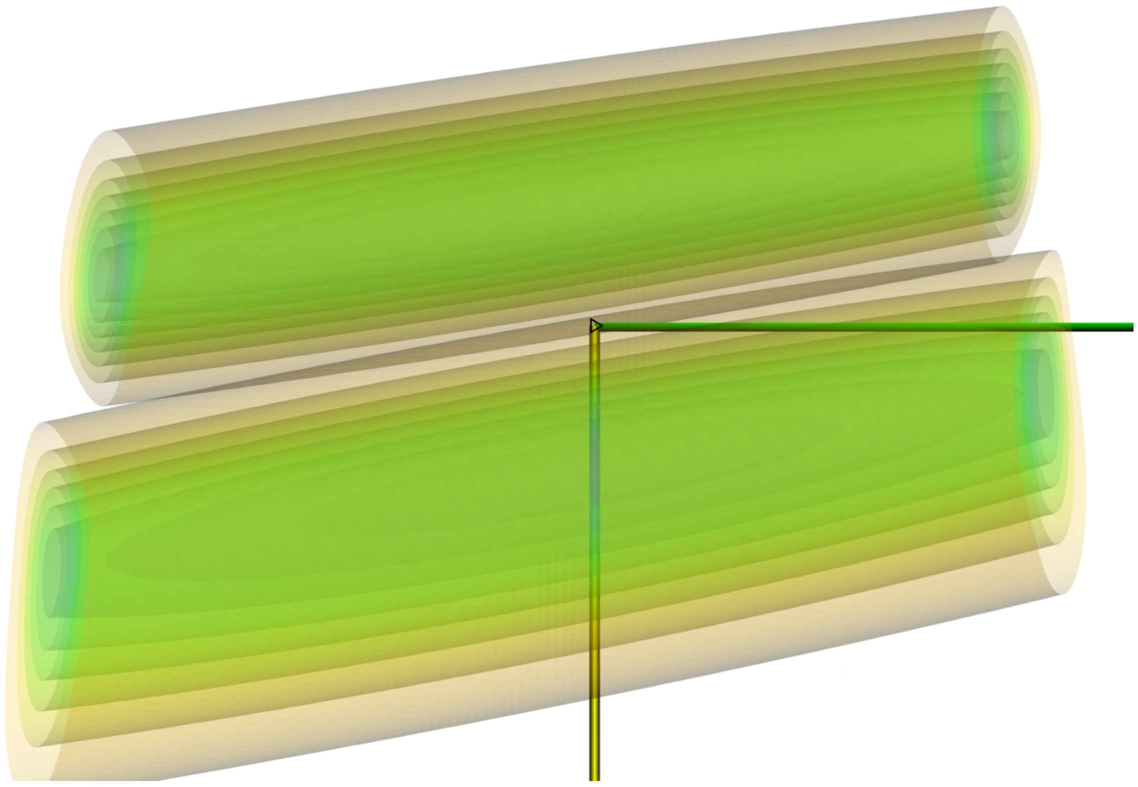


Figure 5.7: Pair of difference images from the *synth4* series, smoothed individually at $\sigma = 4.0$.

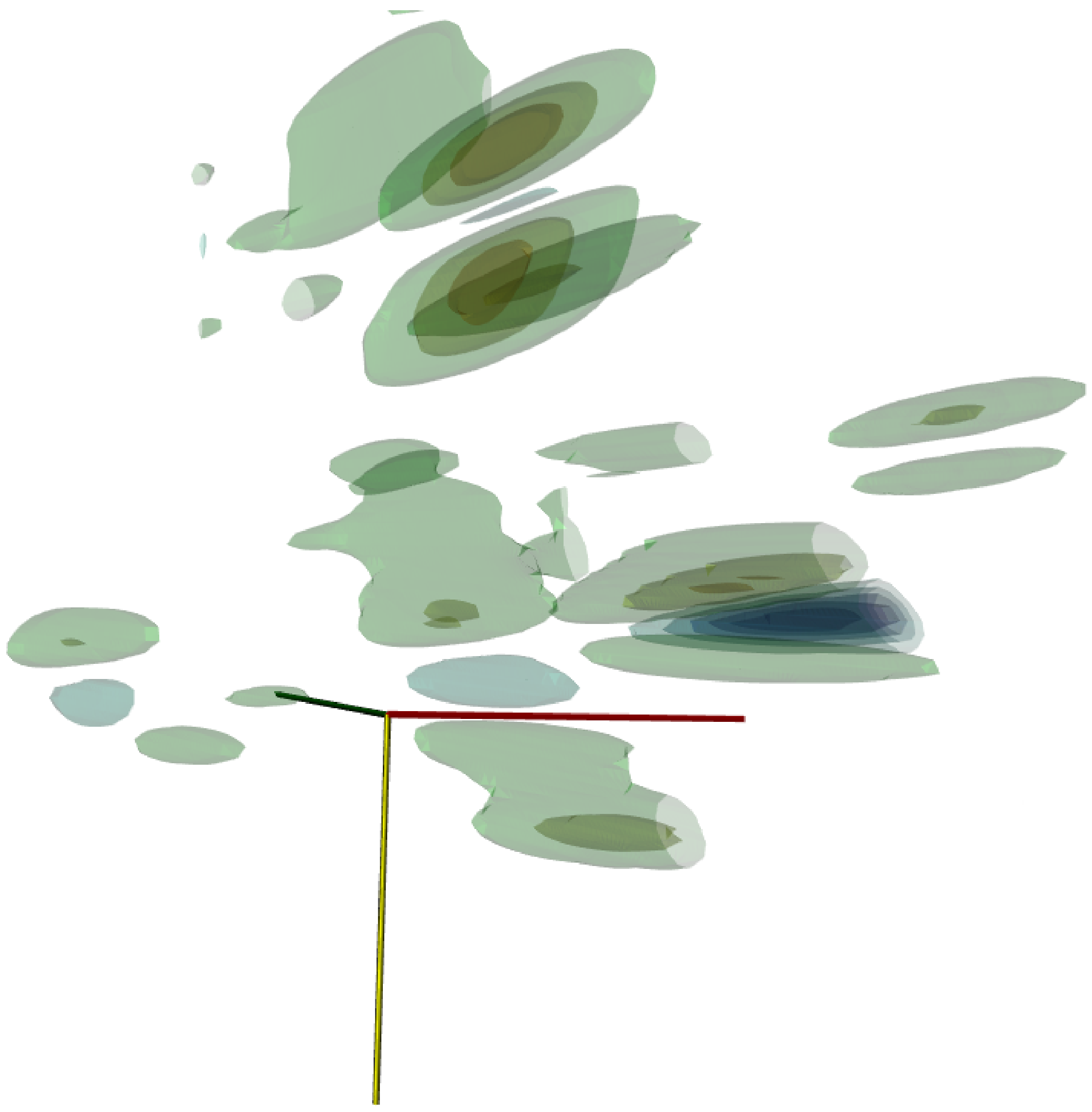


Figure 5.8: Another excerpt from the same situation, except as a difference image (and enlarged by 2). The background is no longer there; instead, one can observe regions where the product of the individual images is positive (ochre) or negative (blue); it is negative when a positive region overlaps with a negative one. This occurs only because of imperfect stereo matching (cf. 5.2 on page 97).

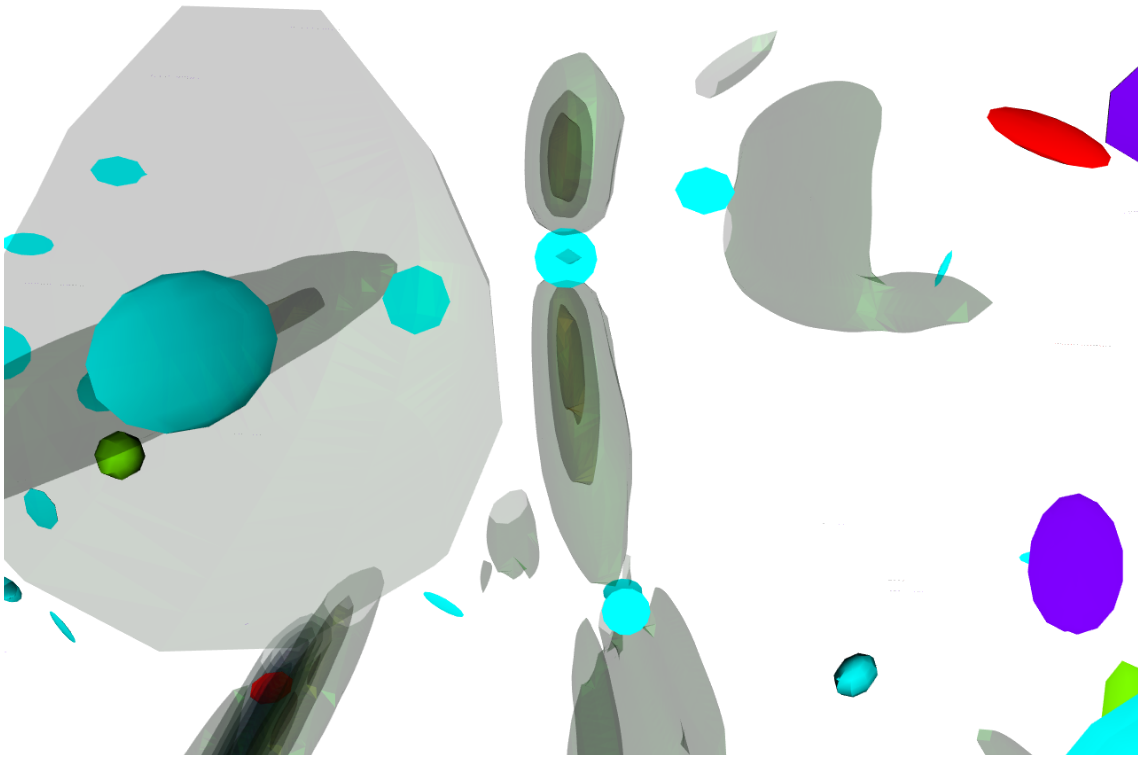


Figure 5.9: Critical points of the two-frame intensity product function have been marked by ellipsoids.

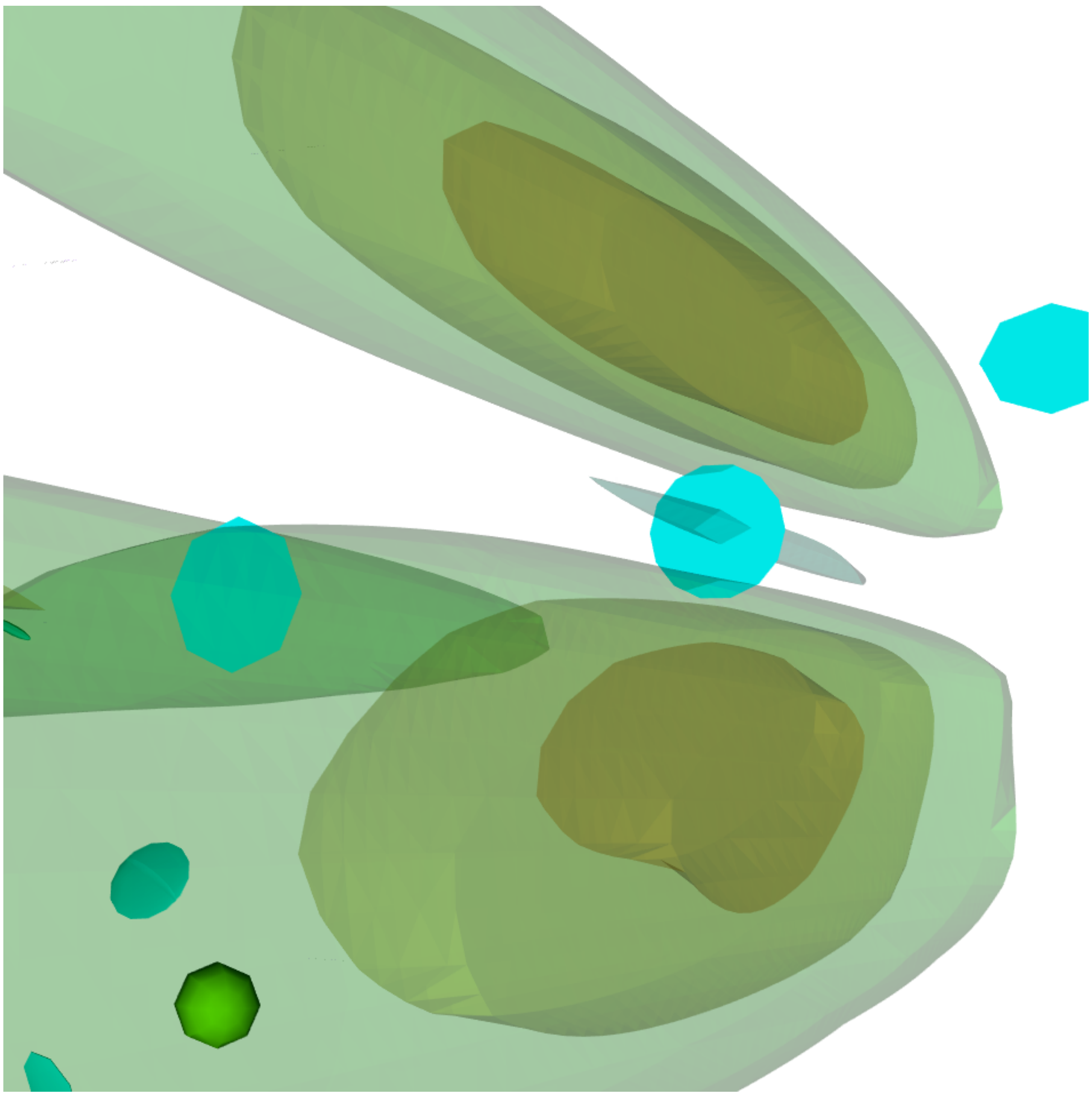


Figure 5.10: Close-up from the previous image: the search algorithm seems to have missed the maxima, but there is a saddle point visible near what could be used as a 3D keypoint.

5.3 Splines

Splines are a class of piecewise polynomial curves, widely used where data is to be treated as continuous.

The word “spline” originally designates a flexible strip of wood used by craftsmen to draw smooth lines ([PBP02], which we would also recommend as a well-written reference) guided by control points, which already nicely illustrates the idea of mathematical splines.

5.3.1 Kinds of splines

One kind of splines, which have many useful properties, are B-splines [Boo01], which we use in this work, exploiting some of their special properties.

As we will see below, B-splines can be used to interpolate or approximate; they share some properties with Bézier spline, almost encountered in the previous section, but are easier to handle thanks to some special properties. Bézier *curves* actually are special B-spline curves.

In this section, we will compare these formulations (Bézier curves and B-splines) and find that B-splines are better suited to describing particle trajectories. After an exposition of the theory, we will start the applications of B-splines to our problem.

Notably, we found it useful to employ them wherever continuous curves were needed: the pathlines or trajectories are nicely represented by B-spline curves, which we obtain by interpolating or approximating several points in space known (or supposed) to be on or near the trajectory.

Non-related uses of spline techniques also implied in this work We used some very convenient spline-based image representations in experiments as well as in the software prototype. These data structures allow for easy and efficient sub-pixel queries of values and derivatives ([UAE93a, Mei09, KM00]). We will *not* explain these methods in this chapter;

5.3.2 Position and direction

One can reconstruct a position and direction in space from two given matched image points and directions. It is of course impossible to obtain the magnitude of the velocity without recourse to further information, and this is where splines come in.

From a spline joining keypoints through adjacent frames, one can easily obtain velocity estimates which are fairly accurate when the motion is well described by a smooth curve. Indeed, the tangent direction of the spline can later be reprojected and matched with the original 2D tangent directions, which (to anticipate a little) was implemented as our first data-fit criterion to filter splines with.

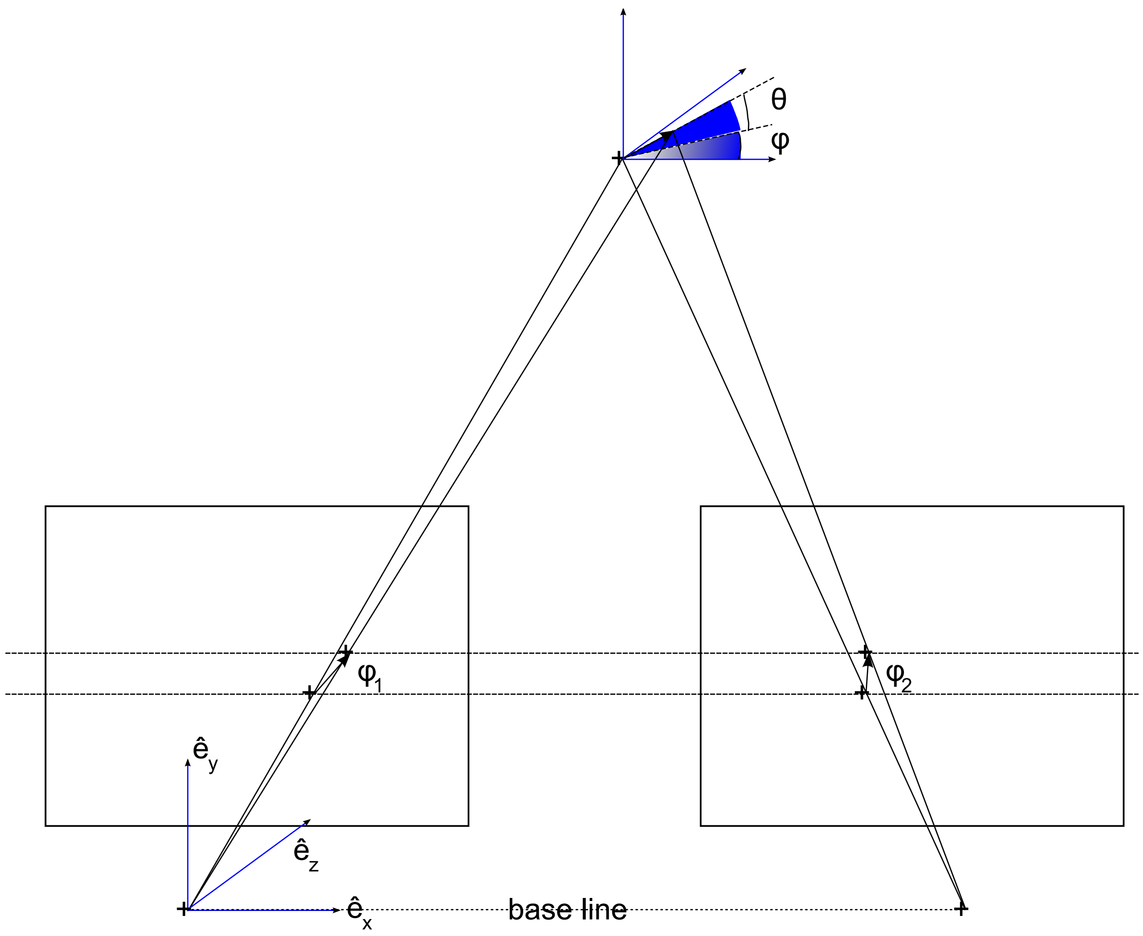


Figure 5.11: One can reconstruct a position and direction in space from two given matched image points and directions.

5.3.3 Bézier curves

Our first approach to velocity estimation via is a direct approach and starts from an estimation of tangent direction in 3D (figure 5.11 on the preceding page) to obtain a curve, which to optimize in accordance to a data fit term. The first choice of curve was a so-called cubic Bézier curve, a natural choice since its parameters have a *direct* relationship to position and tangent of the curve at certain moments.

Definitions and properties

Bézier curves are polynomial curves which can be built from a polygon of n control points. They always have degree $n-1$ by construction.

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{5.1}$$

The curve, when evaluated in the interval $[0,1]$, is contained in the convex hull of the control points.

Cubic Bézier curves One mostly uses cubic Bézier curves and pieces together larger curves from these, so as to avoid numerical problems (it is well-known that high-degree polynomial interpolation is not always a good idea and may cause Runge's: uncontrolled overshoots)

Let us write down the base functions (Bernstein polynomials) for a Bézier curve of degree 3: they are $B_{0,3}(t) = (1-t)^3$, $B_{1,3}(t) = 3t(1-t)^2$, $B_{2,3}(t) = 3t^2(1-t)$, $B_{3,3}(t) = t^3$. Altogether, the curve is $\vec{C}(t) = \sum_i \vec{P}_i B_{i,3}(t)$.

The control points P_1 and P_2: $\frac{\partial}{\partial t} C(t)\Big|_{t=0} = 3(P_1 - P_0)$ are easily interpreted geometrically as the tangent directions at the tips.

Optimization

Figure 5.12 on the following page shows the evolution of a Bézier curve The criterion used in the example on figure 5.12 was $\int I(\vec{C}(t))dt$ on one smoothed image only. The image shows a particle trace; the endpoints were initialized approximately on the "line terminations" and the tangent directions so as to achive a straight line in uniform parameterization.

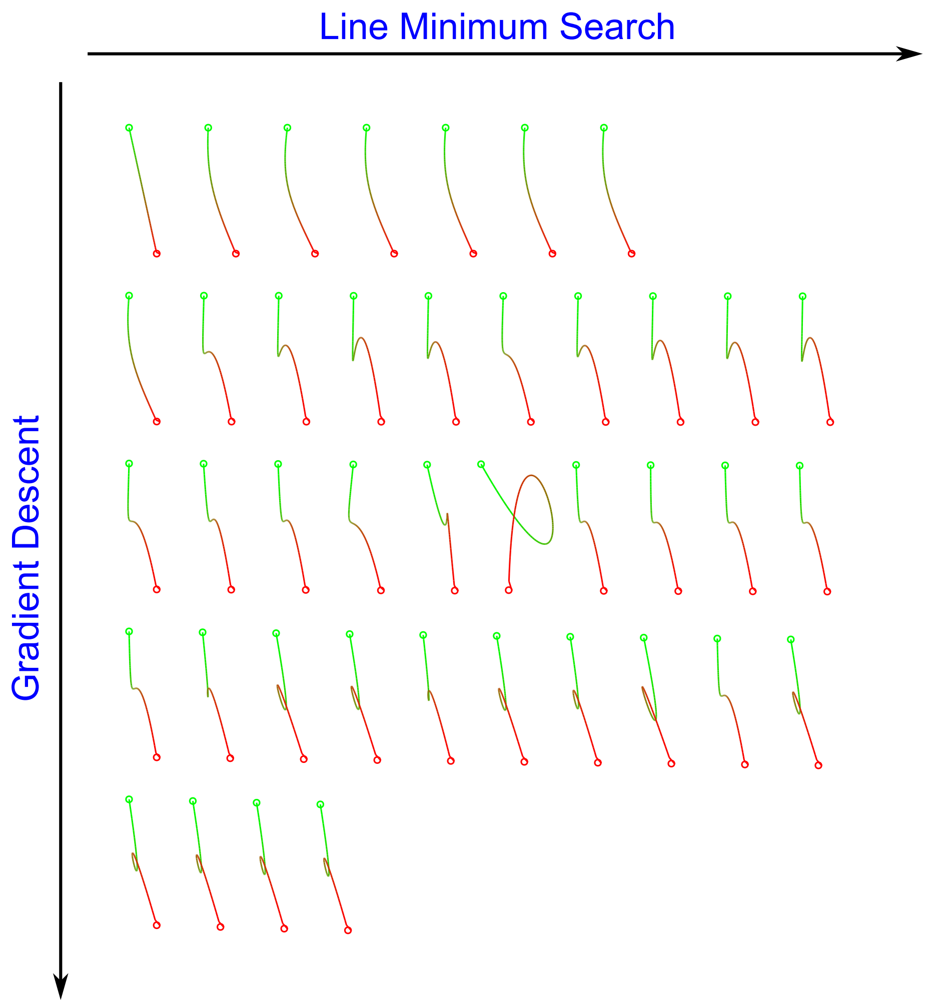


Figure 5.12: The evolution of a curve fragment under "optimization". The table shows a conjugate gradient process: columns correspond to the search for a minimum energy along a "line" in parameter space; parameters here are: tangent directions.

In this examples, the endpoints were held fixed, only the tangent directions and magnitudes free.

We then proceed to optimize that term by evolving the free parameters in the Bézier curve with the method of *conjugate gradients* (described for ex. in the article by Shewchuk [She], which also provides “canned” algorithms). As one can see on figure 5.12, this showed instability, but note that this is just a simple example and it does not necessarily mean an objection to such methods. One can clearly observe that the parameterization plays a detrimental role when capturing purely geometrical properties of the curve. See also sections 5.6 on page 119 and 5.3.4 on page 112 for possible remedies.

5.3.4 Elements of B-splines

B-splines ([Boo01]) form an important part of our method, having been chosen as the trajectory description. Proofs of claims made about B-splines can be found in the literature ([PBP02], [Boo01]).

Curve spaces

B-splines are of the form $C(t) = \sum_i c_i N_i^k(t)$ where the N_i^k are basis functions which are defined as a function of the knot vector and the chosen *degree* or *order* $k \in \mathbb{N} \setminus \{0\}$. B-splines are not isolated objects: all splines which can be defined on the same basis functions constitute a vector space (there are also operations like knot insertion and deletion, which allow to pass from one space to another).

B-spline *curves* are of the form $\vec{C}(t) = \sum_i \vec{P}_i N_i^k(t)$, where the P_i are control points (noted as position vectors).

There are many benefits to using curve spaces which allow high-order approximation to smooth functions with few parameters: one major benefit is the ensuing simplification and regularization. One could view that choice as an instance of Occam's Razor: it excludes many unreasonable curves and avoids supposing a difference between curves which cannot be discerned anyway given the resolution of the observations.

Definition of the basis functions

Let's start with some points which appear in the definition of B-splines, which are helpful for understanding the motivation in the following paragraphs:

To define a space of B-splines, one needs a vector of real-valued *knots* [1], $t_i, i \in \{1 \dots k\}$.

Further, a number of control points + times determines where the curve will go.

The basis functions are piecewise polynomials of low degree. A recursive definition, which stems from the fact that one convolves the lower-order basis functions to obtain the higher ones, can be given as follows:

$$N_{i,k,\vec{t}}(t) = \begin{cases} 1 & t_i \leq t < t_{i+1} \\ 0 & \end{cases} \tag{5.2}$$

[1] sic, always called *knots*, not *nodes*

$$N_{i,k,\vec{t}}(t) = \omega_{i,k,\vec{t}}(t)B_{i,k-1,\vec{t}}(t) + (1-\omega_{i+1,k,\vec{t}}(t))B_{i+1,k-1,\vec{t}}(t) \tag{5.3}$$

with $\omega_{i,k,\vec{t}}(t) = \frac{t-t_i}{t_{i+k-1}-t_i}$

Properties First of all, it is important to note that the algorithm which will determine the coefficients can only work with points in space*time* and if a specific time is not associated with a point, the time would have to be estimated.

The spline basis functions form a partition of unity; that is, at any moment t within the "inner" knots they sum to 1. Outside of the first and last points (if applies) it cannot be used.

The support of a basis element lies between n+1 knots. So they have the further advantage of a certain locality of effects, which is in principle also interesting for speeding up computations.

From the above it is already clear that the capacity for extrapolation when using B-splines is zero. The value of high-order polynomials for the extrapolation of trends from data is at best dubious; splines do not even try.

When all knots are equally spaced (some applications to signal processing use bi-infinite knot vectors), all the basis functions are alike, but for translation.

Multiple knots If we wish to ensure, on a finite-dimensional knot vector, that the first and last points actually appear as values of the spline curve at t_0 and t_{end}, these must be made into multiple knots, of a multiplicity equal to the degree. This (sometimes very desirable) effect can be explained via a limiting process, as shown on 5.3.4 on the next page.

This is particularly easy to see with degree 1 B-splines, which give rise to a piecewise linear interpolation. The ascending leg of the first basis function gets pushed ever closer to the summit.

A B-spline curve, like a Bézier curve, is always contained in the convex hull of the control points (this follows from the partition of unity property). It is even contained in smaller convex hulls of the points whose basis functions' supports contribute.

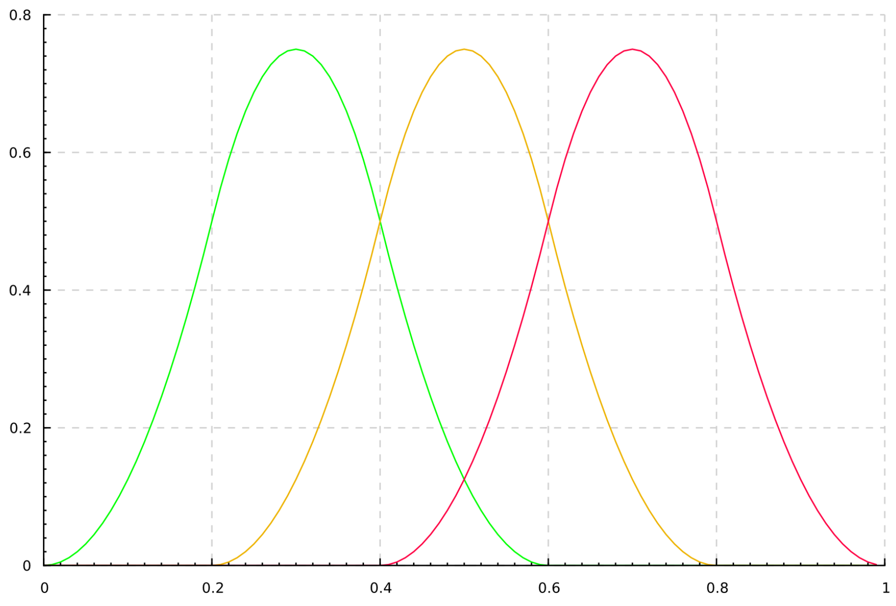


Figure 5.13: Some B-spline basis functions of order 2, on equidistant knots.

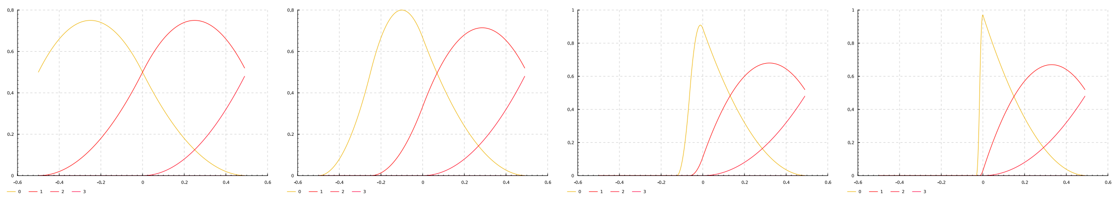


Figure 5.14: The effect upon the basis functions of shifting together B-spline knots

Approximation vs. interpolation

Interpolation is the act of fitting a continuous representation (curve, hypersurface) through a cloud of data points. On the other hand, for an approximation the requirement that the curve pass through the data points is dropped in favor of an approximate version, e.g. a global distance criterion to be minimized subject to other constraints. See [WPL04] for an interesting take on the subject.

Usually, approximation is preferable over interpolation, since any measurement is subject to uncertainty. But see below for special considerations.

How to fit a B-spline curve to data Fitting a B-spline curve to *n* points (observed data), given a vector of "time points" at which the observations presumedly took place, can be done by solving a system of linear equations; they can be used in an interpolating or approximating manner.

(Because it is really spatio-temporal points which are used for spline fitting, one can therefore view the B-spline as traversing a four-dimensional space.)

The following discussion subsumes the two cases, the difference in procedure being minute.

As a preparation, we explain the procedure for fitting a one-dimensional spline to scalar values (vector of observations $\vec{b}$): Determining weights c_j such that the (mean, sum) squared error of the resulting curve to the observed values is minimized gives rise to a linear least squares problem.

$$\arg\min_{c} \left\| \underline{\mathrm{A}}\vec{c} - \vec{b} \right\|^2 \tag{5.4}$$

The solution of this system is well-known; it can be calculated with the Normal Equations.

$$\vec{c} = (\underline{\mathrm{A}}^\top \underline{\mathrm{A}})^{-1} \underline{\mathrm{A}}^T \vec{b} \tag{5.5}$$

While one can derive these equations, one can also motivate the solution of the linear least squares problem with properties the linear operator A

When the data points are to be approximated by a smaller set of basis functions, the system is generally overdetermined and strictly possesses no solutions.

So in effect, one transforms the data vector into a lower-dimensional vector so as to find the best solution within the column span of A. All details are to be found in [Lan96, GL96].

In case the data are samples of an underlying smooth function, the interpolating B-spline has approximation order $d+1$.

Subdivision

Subdividing B-splines for increased flexibility is not hard. There exist sophisticated algorithms for such operations [PLH02]; we experimented with sampling the existing spline and approximating algorithm again. In fact, subdividing destroyed some advantages of the splines in tentative energy optimization experiments (not shown here).

Derivatives

Taking derivatives of parametric curves gives the sought-for velocities. The derivative of a N-th order spline is a $N-1$-th order spline. After [PBP02]:

$$
\begin{aligned}
C(t) &= \sum_i c_i N_i^k(t) \\
C'(t) &= \sum_i d_i N_i^{(k-1)}(t) \qquad (5.6) \\
d_i &= \frac{k}{t_{i+1}-t_i}(c_i - c_{i-1}) \qquad (5.7)
\end{aligned}
$$

Arc-length parametrization

Some criteria (see section 5.6) profit from sensible, parametrization-independent sampling on the image through a curve.

5.3.5 An illustration of the smoothing power

The following images 5.3.5 and 5.3.5 on the facing page illustrate the capacity for smoothing inherent in the choice of the knot vector. The legend for the images: the parameter t is shown with a rainbow color coding. Axes are as usual (as in section 5.2 on page 92: their length is $10cm$ and the Z “depth” coordinate is shown in green).

The 2D keypoint locations were on purpose truncated to integer values. The particle shown, which is from a synthetic image series, originally describes a circle inside a $Z = const$ plane with constant speed.

in the table below, the greatest deviation from that $Z = const$ plane is shown: this is the deviation in the direction normal to the osculating plane.

degree	#knots	t	deviation
1	6	4.0	0.024
2	4	4.0	0.023
2	3	3.75	0.015
2	2	5.0	0.011

The error committed in truncating the subpixel locations is consistent with the regions of uncertainty to be shown in the next chapter (section 7.4 on page 160).

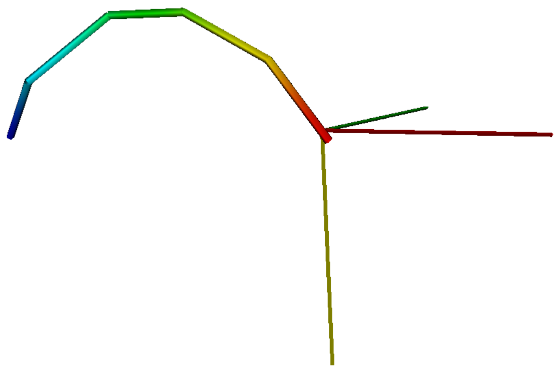


Figure 5.15: Seven (manually selected) 3D matched keypoints from the *synth*4 series, as a polygon curve.

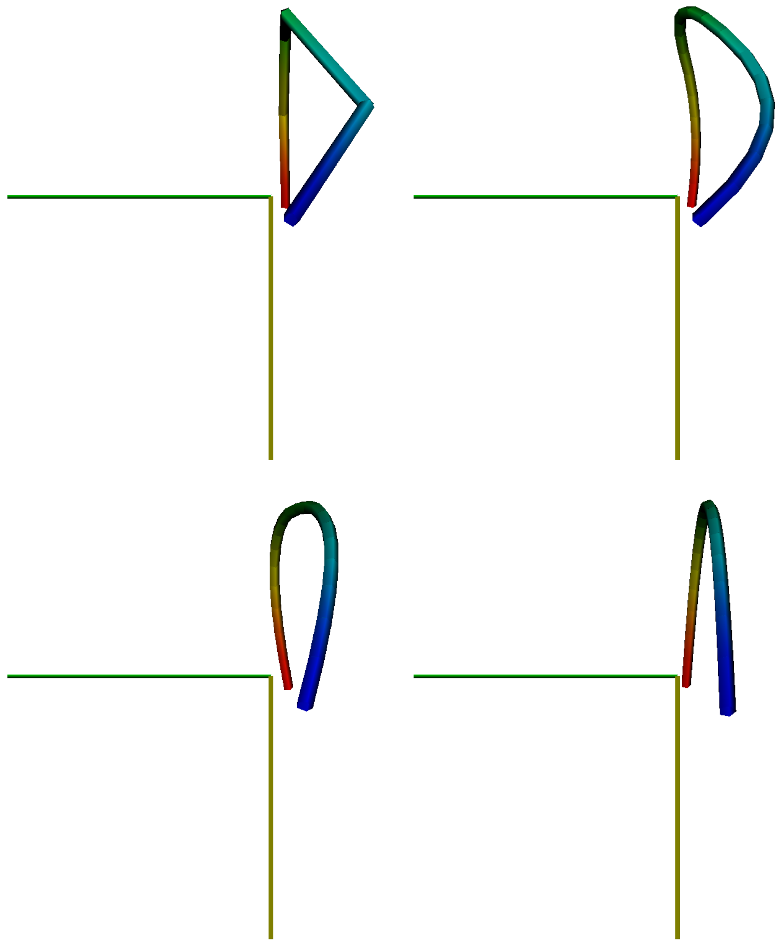


Figure 5.16: $\begin{array}{|c|c|}\hline 1 & 2\\ \hline 3 & 4\\ \hline\end{array}$: The same curve; quadratic B-splines with 4, 3 and 2 knots for comparison. The curves impose various amounts of smoothing on the data.

5.4 Top-down candidate path selection

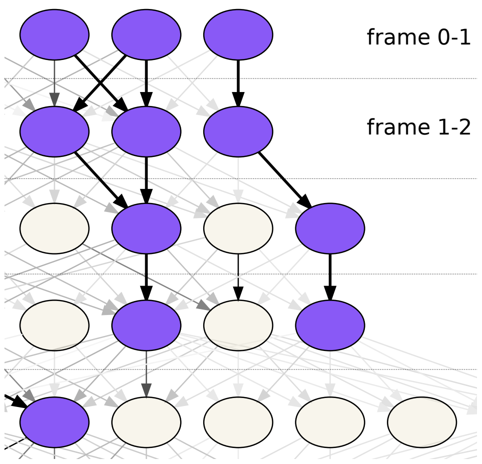


Figure 5.17: The leitmotif for path selection: a directed graph of possible continuations.

As we took a mostly combinatorial way of overcoming the correspondence problem and extracting possible particle motion from stereo views. The next logical step after extracting keypoints was to connect them, in a bid to use a kind of reverse curve matching of our own devising for tracking.

Instead of directly approximating (see above) most of the observations (which are not presented a priori in a "discrete" way – as distinguished from the method of [Lee00], which applies to point clouds), we prefer to work backwards by evaluating a data-fit term on a curve obtained from a few points obtained in an earlier step.

5.5 The Bayesian paradigm

Extracted/assumed pathlines (definition 2.1 on page 15) are corroborated against evidence found in the images.

Originally, a probabilistic interpretation was aimed for (see [JN01] for a reference on Bayesian decision networks, which codify the interrelations of influence between unknowns), as this encourages the modeler to make all their assumptions explicit.

The simplest application of the famous Bayesian formula to the problem of deciding which curves to accept as valid interpretations could be

$$p(\beta|g) = p(g|\beta)\frac{p(\beta)}{p(g)} \tag{5.8}$$

The intended interpretation of the different variables in equation 5.8 on the facing page is: β for a choice of curve parameters, g for the evidence (the frames at hand).

The formula expresses the probability of a certain choice of curve parameters in terms of three probability distributions, which are better accessible:

- $p(g|\beta)$, which could be estimated by means of the generative model
- $p(\beta)$, which could possibly be determined on physical grounds
- $p(g)$, which can be ignored (the evidence being what it is) to obtain a likelihood, which one should try to optimize.

Probabilities were hard to nail down and would have called for a representation of probability densities over large spaces of curves, which would soon again have had to be simplified by taking point estimates ... The current state is that paths are assigned a score, which should be read as a monotonous function of likelihood.

The formulation of equation 5.8 does not contain disambiguating interdependences yet.

5.5.1 Candidate points

The first rule is that nothing is inferred which is not visible; one searches for a set of "maximal" paths inside the graph, or maybe allows subpaths to be included also. These paths must then be assigned a score.

Only trajectories passing through the keypoints detected in the first step can contribute. These points, when they correspond to the projection of real particle locations, are associated with a well-determined time stamp, the cornerstone of velocimetry. Ends of traces, where a particle vanishes and all time information is lost, are of no use in velocimetry.

A serious stumbling block is the presence of errors in calibration and undistortion, which can ruin the detection of stereo pairs (3D keypoints) if one isn't careful.

5.5.2 Candidate paths

The 3D keypoints serve as vertices, joined by edges. There is no harm in including every possible edge between nodes of two consecutive frames, in principle, but in the actual system, this explosion in the number of paths must be contained as effectively as possible.

5.5.3 Graph view

The candidate 3D keypoints points and candidate paths joining them constitute an acyclic directed graph of possible continuations. Figure 5.17 on page 114 shows an excerpt such a graph. The edge strength, indicated by the line width, was based only on spatial distance.

One can reasonably consider it as a directed hypergraph. Generic algorithms for use on hypergraphs have yet to become mainstream, but the joining of candidate keypoints over several frames with interpolating or approximating spline curves has strong reasons going for it.

If, as we proposed in the sections on splines, one uses only the 3D keypoint locations as data for spline fitting, there is no way to obtain anything but a straight line from two locations. The goal of curve fitting is then only attainable by possibly subdividing and optimizing, which turned out to be fraught with difficulties in our experiments.

This is one more argument in favor of the approach chosen – if one considers paths of moderate length, curve optimization is not needed.

Combinatorics of subpaths

There are good reasons for breaking down long chains of locations putatively belonging to one particle into shorter ones (with overlaps allowed, when appropriate). Temporal locality plays a role, though it is already ensured by the use of B-splines (section 5.3.4) instead of polynomial curves: it is not helpful to describe the whole trajectory in one piece.

Often, the maximal chain which can reasonably be built has poorer support with the evidence than a subchain (e.g. when a particle vanishes); it is therefore necessary to evaluate all subchains up to a certain length $f \in \mathbb{N}$. There are $\binom{f}{2} - f$ of them, that is $O(f^2)$.

With a k-nearest-neighbour heuristic, one can limit the total number of paths which must be checked to $O(k^f \cdot f^2)$

Taking intra-frame observations into account

Up to this point, only individual locations have been detected, at discrete moments in time, and only very little information has been taken into account.

More evidence can be gathered from the images, such as the data-fit terms from section 5.6. There are several possibilities for managing this, but it would be most satisfying theoretically to work from equation 5.8 on page 114 and harness the probabilistic framework.

5.5.4 Data-fit terms for evaluating candidate paths

Our basic point features (putative locations of particles at the interface of two frames) can be linked to form “time-discretized skeleta” of surmised trajectories. The paths described thereby are mere chains of keypoints and not much can be said about them directly.

Whilst some geometrical constraints can already be imposed on physical grounds (particles do not jump across the room), this representation alone does not provide sufficient leverage for assigning credibility, because it does not contain sufficient information.

Ultimately, the Bayesian framework should give the answer. One can imagine using a generative model (such as the one introduced in chapter 4) to evaluate $p(g|\pi)$ (this time with π representing the entire set of paths) and use an algorithm like Expectation-Maximization [JN01], with π as hidden variables, to find a locally optimal solution.

Disambiguating constraints

Considering the physical properties of pathlines (definition and discussion are in section 2.1) one can establish a constraint which makes sure that measured particle trajectories never cross, since that is physically unlikely – a collision would probably destroy soap membranes, if the particles are bubbles. (their projection images, of course, may cross – that is not relevant.)

Remaining in the probabilistic paradigm, one could get away with ignoring some interdependencies.

5.5.5 Conclusion

Many questions remain for further research.

While the prototype so far filters candidate paths according to criteria, it will certainly be a significant theoretical improvement if one can effectively harness the Bayesian formulation in the future.

How to proceed after constructing the (hyper)graph of continuations depends also on efficiency considerations – there are design decisions to be made. Up to this point, we experimented with several alternatives.

Does one keep only the best-fitting path for each node (and apply a threshold based on the same data-fit criterion on the remaining paths) or does one allow contributions from different paths, possibly from conflicting paths then to simplify the method?

The sections on experimental results, where a screenshot can be seen, show the second variant in action.

The first variant certainly leads to a smaller data volume at the output, and is probably the most sensible one to adopt. It necessitates a combinatorial optimization; a greedy algorithm, which starts from the best continuations and successively flags edges conflicting with the paths already chosen, also worked well in our experiments.

5.6 Energy functionals

This section serves to introduce the possibility of evaluating curves according to internal properties on the one hand – particle trajectories having very sharp bends are less likely, physically, than straight trajectories or at least ones presenting a smoother curvature profile – and image information on the other hand.

5.6.1 Optimizing curves

When one wants to pose an optimization problem in a function space (e.g. a space of curves), so as to obtain a solution with a certain property (here: a data fit term for a curve), the functional is usually called an Energy (the name stems from physics), and the optimization is an iteration which converges to functions which are stationary points of the Energy.

Usually, such spaces are infinite-dimensional and one needs the Calculus of Variations to deal with the optimization (sometimes one can obtain good approximation schemes by discretizing afterwards, see the ample literature).

Models like b-spline curves (on an usually fixed knot vector) are really helpful here, because the infinite-dimensional optimization problem is immediately replaced by a finite-dimensional one, where one has only a few parameters for optimization. [BHU00]

5.6.2 Energy models

Next, we list some usual, and unusual energies, and examine what we can learn from them.

Our approach to shape fitting by an energy term is inspired by *Active Contour Models* or *Snakes* [KWT88], Which has since inspired many other authors to propose models based either on various physical analogies, or derive some on information-theoretical grounds.

A small anthology of active contour models

With $\vec{v} : [0,1] \to \mathbb{R}^2$ a differentiable curve, (for all relevant definitions, refer to [BG80]) and derivatives marked by subscripts, following the standard in Snake literature.

Often, one encounters active contours which are closed curves applied to image segmentation; balloons and level set methods. Our "contours", on the contrary, do not bound a region, which would make the name a misnomer in our case (or which shows the difference).

It is interesting to note that [KWT88] already discuss the use of their Snakes for stereo matching. The few other publications about using active contour models in an epipolar geometry context we became aware of are [CC97, CB90].

The traditional snake of [KWT88] is $\int_0^1 E_{int}(\vec{v}(s)) + E_{image}(\vec{v}(s)) + E_{con}(\vec{v}(s))ds$. The "internal" energy is $E_{int} = \frac{1}{2}(\alpha(s)|\vec{v}_s(s)|^2 + \beta(s)|\vec{v}_{ss}(s)|^2)$ The "image" energy is usually derived from edge terms like $E_{image} = -|\nabla I(x,y)|^2$ or a smoothed variant. Also interesting: the "termination" (of line segments) seeking energy $E_{term} = \frac{\partial\theta}{\partial\vec{n}^{\perp}}$ with θ the gradient angle of the smoothed image. The ingredients are weighted according to their importance in the specific problem.

[CKS97] first modify the problem $E_{int} = |\vec{v}_q(q)|$ and $E_{ext} = g(|\nabla I(\vec{v}(q))|)^2$ (with $g : \mathbb{R}^{+*} \rightarrow \mathbb{R}$ a strictly decreasing, asymptotically vanishing function), deliberately scrapping the curvature part, as it is in some sense implicit in the formulation, then they remark that the expression is still not intrinsic. Finally, they arrive at $\int_0^1 g(|\nabla I(\vec{v}(q))|)|\vec{v}_q(q)|dq$, which resists reparametrizations, can be interpreted as a geodesic length and optimized by Euclidean Curve Shortening Flow ([CKS97, ABF99]) [CV01] also has this.

Then there is [XP98]: their GVF propagates information into homogeneous regions and allows faster convergence (at the cost of intensive PDE iterations beforehand) and also makes the contour move into concave regions, and [YMX06], which also propagates edge information, but handles only closed curves.

5.6.3 Adopting an energy

Before we stray too far afield, let us return to the problem at hand, which is to assign a score to each curve encountered.

As in the geodesic active contour models, we do not need much of an internal energy because parametrization is

- mostly irrelevant at this stage (of shape description)
- maybe actively harmful, as we saw in the Bézier curve experiments.

But it proved very important to penalize deviations from initial length, lest the result looks like pop-art.

5.6.4 Conclusion

Optimization of open-ended curves is fraught: does one require that the control points stay put? In that case, with b-splines, there is nothing left to optimize. Or does one impose other artificial constraints like restricting the control points to certain regions ... so we decided to forgo actually evolving the curves and settled for calculating energies on the original curves instead.

Another justification for not needing actual optimization is that localization (precision) is good enough so that even in the case of bad accuracy, one can ensure that the reprojection still hits the right information (we ensure this by effecting a slight affine transformation of the images, such that epipolar constraint is better fulfilled).

One can still, however, make use of an energy: to judge a curve without trying to turn it into a local optimum first. There required interfaces are in the prototype.

5.7 Hyperbolic and parabolic regions of 2+1D images

Earlier, we introduced criteria to filter trajectory candidates according to supplementary image information. Currently, there exists a filter based on tangent directions of the re-projected curve, but other criteria are being investigated and implemented.

If one views an image as a continuous function, one can partition the domain of an image into hyperbolic and parabolic regions, and the surfaces separating them.

These regions are formed according to how the image, considered as a surface, is curved.

This partitioning has been used for region segmentation (see [Flo97]). They can be detected by analyzing the derivatives of the gradient.

Note how the Hessian's properties vary *smoothly*.

On the images (figure 5.18ff.), the surfaces of vanishing determinant are shown; they surround different kinds of regions and are always neatly apart in generic sufficiently-smooth images.

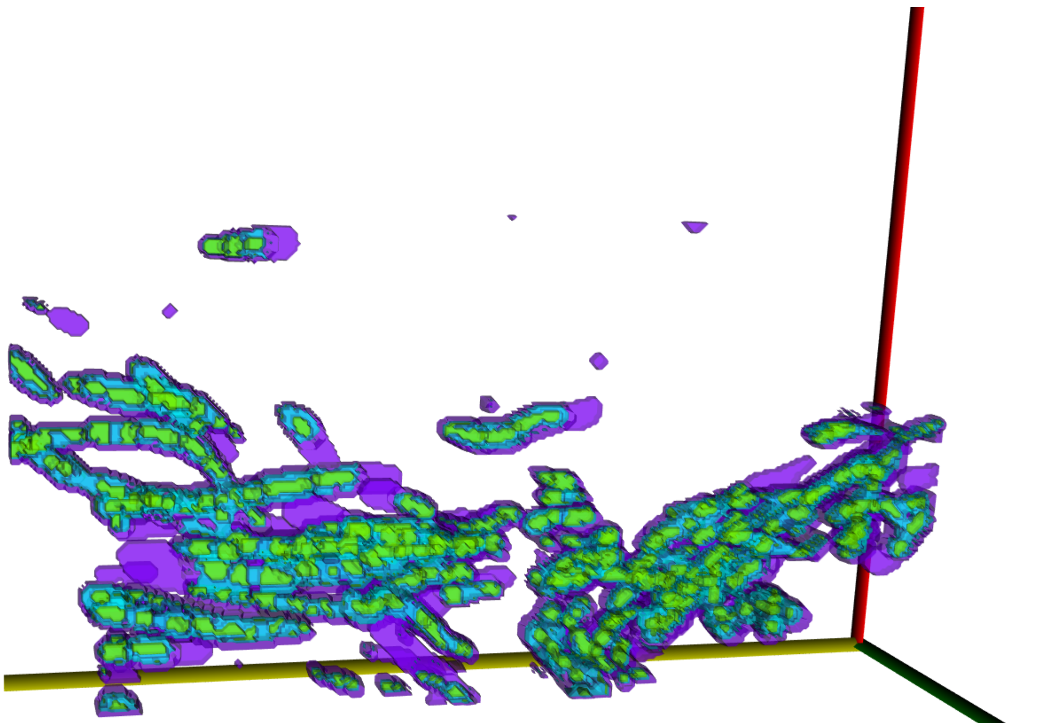


Figure 5.18: *V4*, left views only, as 2+1D volume. On this "spatio-temporal" volume image, elliptic "maximum-like" regions are inside the green contour, those with one direction of increasing gradient are represented in cyan and those with two are shown in violet.

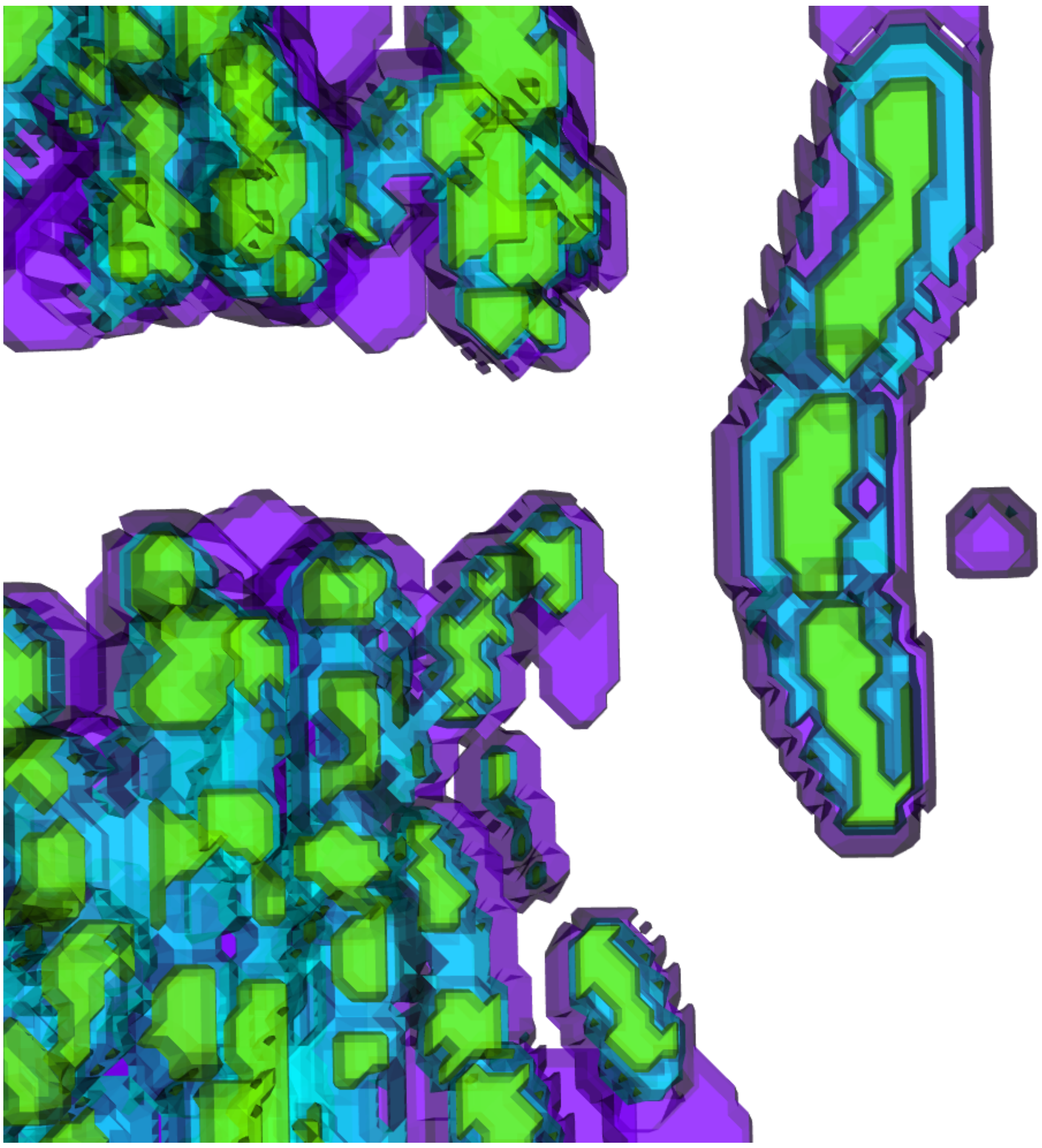


Figure 5.19: A closeup from figure 5.18 on the preceding page. Contemplate especially the isolated particle trace in the east

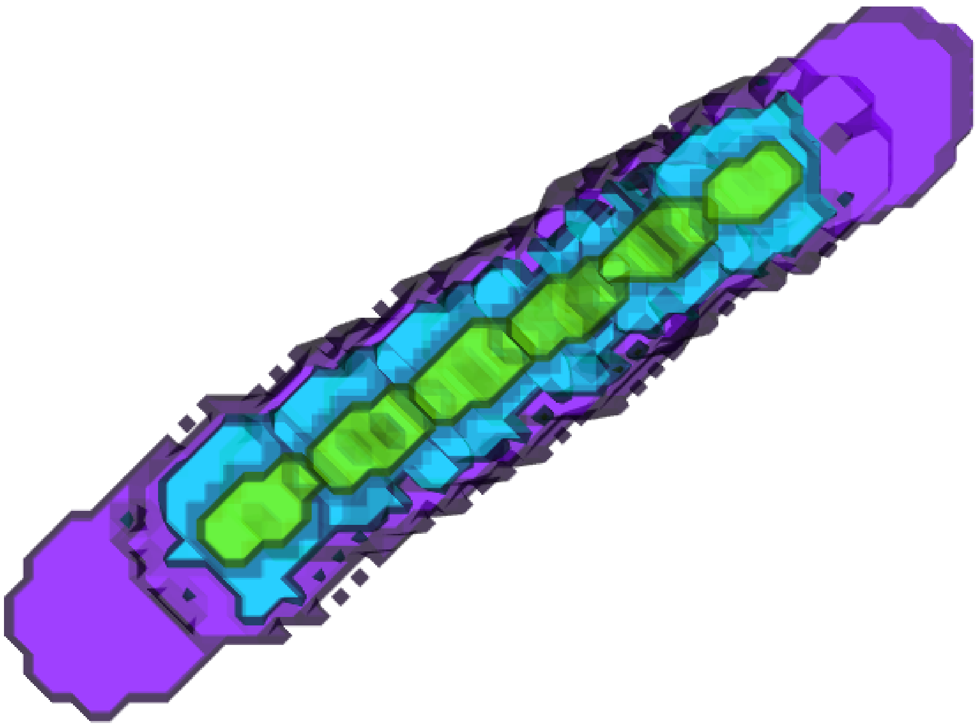


Figure 5.20: The same, from *synth1*, for comparison. (viewed from “below” on time axis).

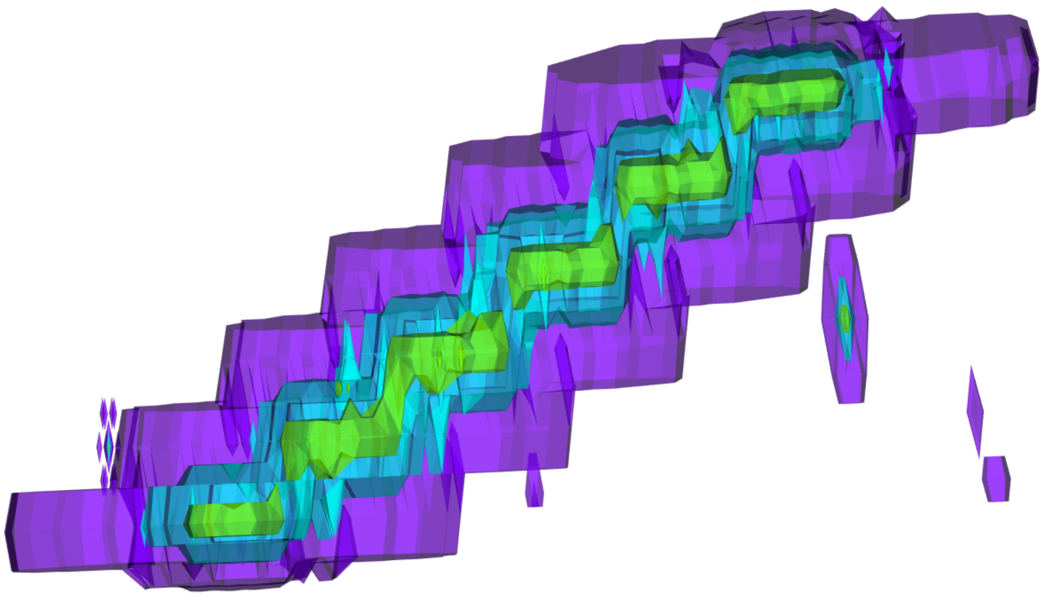


Figure 5.21: Same as 5.20 on the preceding page, side view

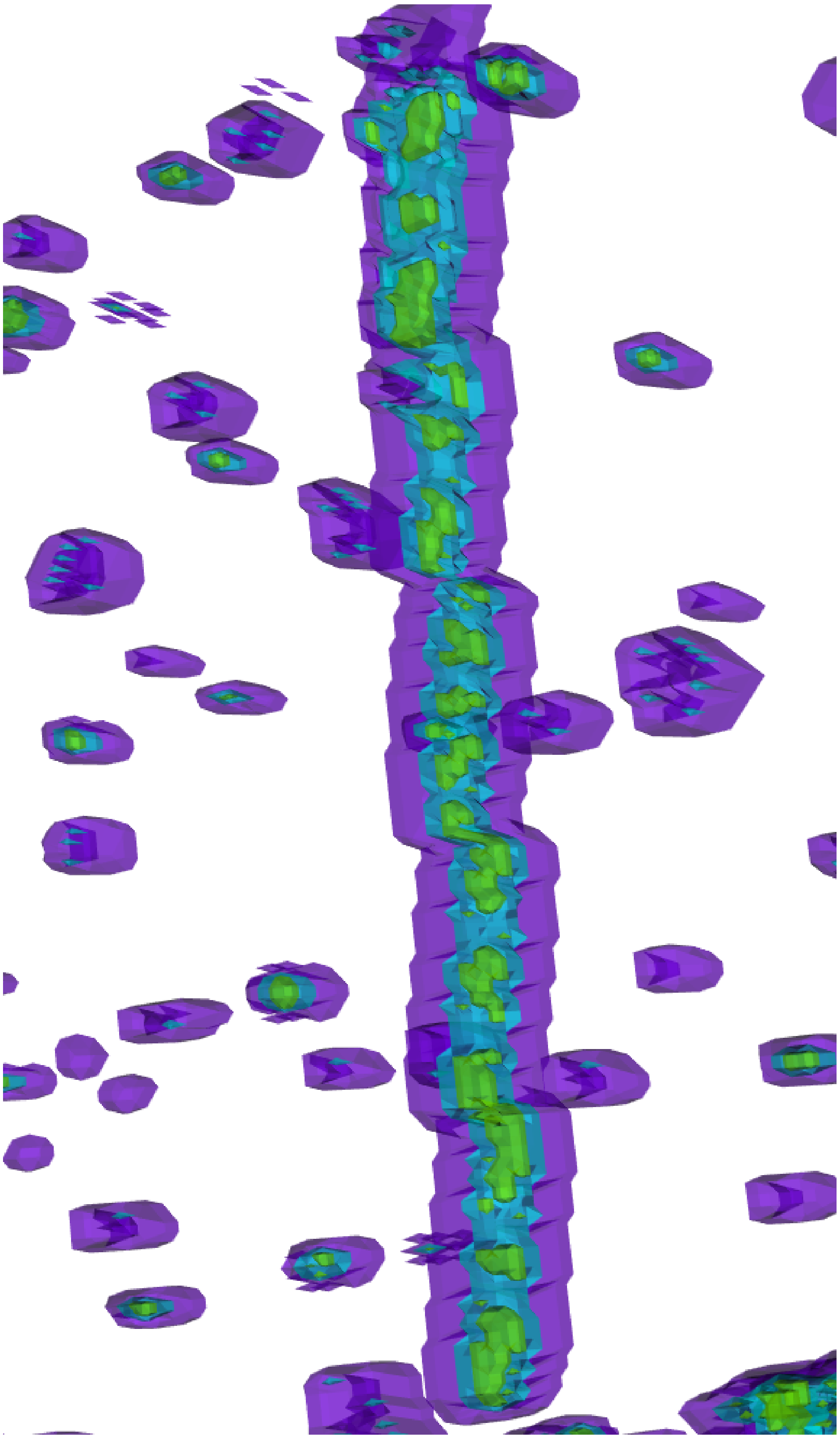


Figure 5.22: An elongated trace from a rapidly moving particle.

5.8 Conclusion

In this chapter, we investigated how to combine the individual image series, using the calibration data and techniques from chapter 3, to a three-dimensional view.

We revisited two classes of parametric curves which can be specified with a finite number of parameters, and motivated our choice of B-spline curves over Bézier curves.

We also introduced a multitude of different ways of determining which curves are most likely to correspond to actual particle trajectories and therefore should be counted as measurements, and made first steps toward integrating all of these considerations into an overarching, probabilistic model.

All relevant image information has thus been reduced to the geometry of four-dimensional curves, which is almost what we demanded of the extraction process: leveraging the parametrization of b-splines yields the desired velocity estimates, since these curves can be treated as particle trajectories and sampled accordingly.

Chapter 6

Software architecture

In this chapter we will discuss the prototype, which we implemented to test our system, and which will be handed over to Airbus. We will discuss the requirements which the prototype had to fulfill, and which design decisions we made to meet them. This is followed by an overview over its components, with a special focus on the key component, in which the particle detection and tracking is implemented. A documentation of how to operate the prototype can be found in the appendix.

6.1 Design decisions

When considering what language to use for the development of the prototype, we considered C++ and Matlab as the main options. In the initial requirement specification, it was intended that the first prototype should be developed in Matlab, before the porting the entire system to C++.

Development in Matlab would have had the advantage of easy prototyping and the existence of image processing toolkits, along with the possibility of linking specifically compiled C code. It was also thought that we might be able to reuse parts of the existing system for 2D velocimetry.

However, the existing system proved completely unsuitable for the task of depth reconstruction. Image processing libraries are available for other languages as well, and as an eventual implementation in another language than Matlab was wanted anyway, we decided against the use of Matlab.

Instead, we opted to develop the prototype in C++. Several factors contributed to this choice. The first was given by the intended use of the application. Our industry partner wanted a system which allowed rapid testing of ventilation inlet configurations.

C++ gives the developer the ability to write highly optimized code, and complete control over the programme flow. Another key factor in the decision towards C++ was the number of computer vision libraries available for that language, which simplified development considerably by providing optimized functions for many standard image processing tasks.

6.2 Implementation

6.2.1 Requirements

As this thesis grew out of a cooperation with an industry partner, who was interested in improving his measurement capabilities, the development of a functioning prototype was a main objective. There are several requirements, which the prototype had to meet:

- *Usability*: A user should be able to operate the program with no knowledge of the underlying PTV system. We took effort to design the system in such a way, so that the user only has to provide paths to input and output data. There are no parameters for the actual detection algorithm that have to be set by the user.
- *Visualization*: The user must be given a way of quickly viewing the generated particle trajectories from within the program. To meet this requirement, we implemented two visualization widgets, which both can display particle trajectories.
- *Output for postprocessing*: Results must be accessible for external programs in a well defined format. The trajectories are made accessible through a XML format.

6.2.2 Overview over the components

Figure 6.2 on page 134 shows the structure of the prototype with all its modules. Each module will be discussed separately, along with the data exchange between the modules. The framework for the prototype was provided by the QT library [BS08], which we used to design the Graphical User Interface (GUI).

Calibration and undistortion

The prototype assumes that the intrinsic and extrinsic parameters are already known. We used a matlab toolbox ([Bou08]) for obtaining the single and stereo camera calibration data, along with estimates for the distortion coefficients. These are stored in a XML file, which can be serialized to matrices using boost::serialization ([dt]).

As both mask generation and keypoint extraction can operate on distorted images, it is not necessary to undistort entire images. We have therefore implemented undistortion form of a utility function, which can undistort single pixels.

World to camera calibration

We were supplied with a model of the test cabin by Airbus. In order to make it usable, we needed to implement a way of determining the $4x4$ transformation matrix between the cabin coordinate system and the world coordinate system used by the camera. Section 3.3.5 on page 43 discusses the algorithm we used to obtain the transformation.

The user is presented with a two panel interface, into which image pairs from the camera are loaded. These image pairs have to contain markers with known cabin coordinates. The user has to select corresponding pairs of markers, which are triangulated, as described in section 3.6 on page 54. Once three point tuples have been selected, the transformation matrix is estimated and written to a XML file, in addition to being emitted as a signal.

Rectification

At first, we rectified entire images as a preliminary step. As with the removal of lens distortion, this is highly inefficient. The only time rectification is useful, is when stereo matching is being conducted. Neither mask generation or key point extraction require rectified images. It is therefore enough to determine the rectification transforms once and only apply them as necessary to points.

Detection

The detection widget contains the bulk of the system. It is parametrized through the GUI and passes those parameters on the threaded Detect-Thread class, which supervises the execution of the detection algorithm. A flowchart of this class is shown in figure 6.1 on the facing page. Detection consists of mask generation, key point extraction, and path generation. Each is implemented as a separate class. The thread will be discussed in detail in section 6.3 on page 136

We used the Mean Shift implementation provided by OpenCV ([Cor]), operating on an image pyramid. A variation of the OpenCV Mean Shift algorithm, which allowed for three features (in addition to space localization) was implemented but not used due to lack of easily computable features for mask generation, and because the intensity proved sufficient.

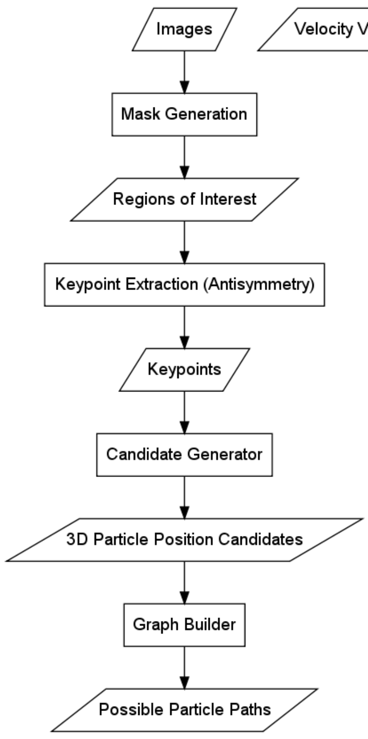


Figure 6.1: Flowchart of the detection thread.

Visualization

We implemented two visualization widgets, both using the same set of helper classes to load the input data. One widget was developed directly in OpenGL, while the other makes use of the Visualization Toolkit library ([SML97]) . The OpenGL widget was designed with the goal to optimize rendering performance. To achieve this aim, rendering commands for particle traces, cabin geometry, and axes glyph, are cached. The traces are color coded in accordance to their velocity. In order to display their orientation, we designed a custom 3D arrow class, using cylinder primitives.

The other widget makes extensive use of the VTK library to provide additional features over the OpenGL widget. One is the ability to interactively select traces via casting a ray to intersect the displayed volume, and display information on the selected trace. Another difference to the OpenGL widget is that traces are rendered as cardinal splines, instead of lines.

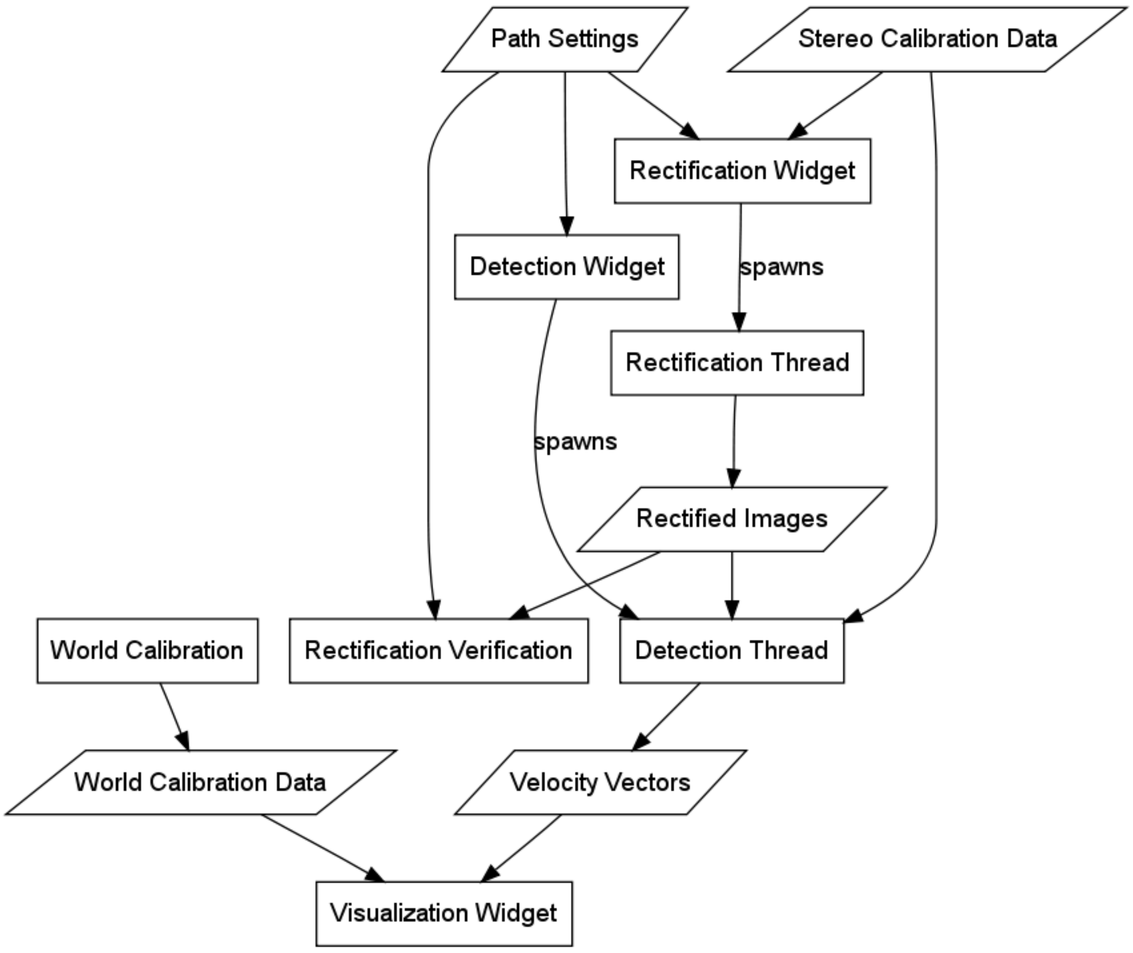


Figure 6.2: Flowchart of the Prototype's initial design. All images were rectified before further processing steps were taken. Processes are displayed as rectangles, whereas data is represented as parallelograms

Data exchange

Each prototype widget offers signals and slots, which other classes can connect to, following the known Observer pattern [GHJV94]. Part of the communication between Widget classes and their member object follows the observer pattern, as well, while the rest occurs through calls of public methods by the parent class. Designing all communication according to the Observer pattern was not desirable, as the signal/slot framework is provided by QT, and gaining a dependency on that library was not worth the added flexibility gained by the Observer pattern. We attempted to use boost::signals instead of the QT implementation, but found that compatibility was poor. Communications between GUI widgets is handled by a central class, which establishes the required connections.

6.2.3 Modular design

The system has been designed with the aim to make it as modular as possible, so that implementations of algorithms can be easily exchanged and compared. To this aim, well defined data structures for the communication between the modules are required.

For the most data, XML, was chosen as storage format. The choice was made because XML standardized and human readable. Using boost::serialization [dt], it is possible to serialize arbitrary data structures to binary or ASCII files, which can serve as input to other modules, and can be easily debugged by developers. Serialized files offer limited backwards compatibility, as long as non-required fields are to or removed from the data structure.

While the XML format is convenient for data extracted from the images, it is not optimal for the image data itself. Serializing the image to and from XML would be computationally expensive and bring no gain in terms of readability of the data. Instead, filepaths of the image are exchanged by the modules, which load them as needed.

The reconstruction process is divided into modules, based on self contained steps which were identified. The Rectification module accepts calibration data from the hypothetical calibration module and produces rectified Tiff images as output.

The graphical user interface (GUI) mirrors the modular approach of the computer vision system (better term). User relevant modules are placed in their own tabs, and can operate independent ly. This is achieved by placing each module into its own thread.

Input data relevant to several modules, such as Data paths, can be entered at any and is passed on using qt signals. The same principle applies to the distribution of results. Each module provides signals which modules can register to receive. Modules emit internal signals, which the GUI containers receives and uses for status updates.

6.3 Anatomy of the detection process

Most matching and reconstruction steps are called from the “detection thread”.

They are packaged in such a way as to be independent of both the user interface and the visualization of data, communicating with other components over specific interfaces and data exchange formats.

In this section, we will detail some of the data structures internal to the implementation of the reconstruction process.

6.3.1 Camera geometry

All camera parameters are contained in a C++ class “CameraData”. Since in the object oriented paradigm one can package the relevant functions together with the data they depend on, said class has members corresponding to all the triangulation, reprojection, undistortion, rectification operations, which can be used as needed: to pass from rectified to non-rectified coordinates, to undistort these, to pass from image to scene coordinates and inversely to reproject scene coordinates onto the images.

6.3.2 3D candidate point generation

For the generation of 3D keypoints, which serve as a “basis” for the final trajectory reconstruction, it is not of paramount importance that the choice of 2D keypoints be very specific; our approach relies on generating a multitude of candidates and testing them by trying to reconstruct curves from them and filtering afterwards.

One can always combine 2D keypoints to a 3D keypoint (as per the mapping of 3.6 on page 56). Rectification still comes in here because it makes determining a window for match search easier: if one works with rectified views, a window of Δpy scan lines on the rectified image can be used.

Then one proceeds to triangulating every one of these pairs (which is always possible, 3.6). While strictly spoken one should also scale the window depending on the position on the image y axis (the impact of a small error depending on it – one risks committing a systematic error, but the generate-and-test philosophy provides a guard against it).

6.3.3 Implementation of matching

The matching algorithm employs an Y-sorted data structure with pointers. Our platform being C++ based, we profit from the STL [SL94] data type **set**. This generic data type allows efficient (logarithmic) retrieval of ordered data, though mostly sequential traversal of the data structure is used. **set** also uses the comparison operator to ensure that it contains only elements deemed different from another (as its name suggests).

In a single pass, all suitable candidates inside the matching window are collected.

Error in calibration

Miscalibration is easily identified as a major nuisance in matching – in practice, it always leads to a discrepancy between the views and means that the Epipolar constraint is not fulfilled, especially when one is far from the footpoint (*).

Local warping was introduced as a hot fix. By locally approximating the action of recalibration with a translation (only a translation, because a general affine transformation has more DOF than can be fixed in this way. After all, we only have the epipolar constraint available to determine), better matching through neighbouring frames became possible. It is important to note that by doing so, one does not reduce the error present in the measurement, but only improves matching in difficult situations.

6.3.4 Candidate path generation

A central part of the reconstruction algorithm, and the second most time-consuming after initial image processing, is the generation and evaluation of candidate paths.

Nearest neighbour as a heuristic

The prototype relies on the ANN library [Mou98], which implements an algorithm described in [AMN$^+$94] to perform approximate nearest neighbour searches (which is much cheaper time-wise than actual nearest neighbour searches in high-dimensional spaces – this is not an issue at all in our application but the library is very handy even for simple searches).

It contains a radius search function, which is especially useful, since it is difficult to determine in advance how many neighbours are going to be needed, since most candidates will be spurious.This way, one can exclude chains of keypoints which are too far apart from each other.

In view of the combinatorics (5.4 on page 114), which may lead to an explosion of possibilites, it pays to prune early in spite of the generate-and-test ansatz, as long as no significant contributions are lost.

With low frame rates, extrapolation is not necessarily possible (4.1 on page 63) – else one would search at the predicted place. Instead, we restrict the search for the next keypoint to a disk around the first keypoint. ANN contains also another useful feature, which is a k-nearest neighbour search (instead of a radius search), which one can also use to predictably restrict the branching factor of the graph of continuations (previous chapter,5.4 on page 114).

Curves

Curve fitting and evaluation is done with a home built B-spline approximation class. This way, it was possible to implement all required extensions, such as the calculation of derivatives directly from the B-spline basis, as needed.

Data structures

The program of part 5.4 (the top-down candidate path selection) has been rudimentarily implemented as a generic algorithm on a graph data structure.

Different criteria can be plugged in. The criteria are pure functions which map candidate chains of keypoints to a “score”, and serve to select the best paths accordingly. They may have recourse to supplementary image information, which is made possible through a ring buffer to avoid loading the whole image stream at once.

Pluggable criteria

There are three criteria for curve ranking being evaluated at this time.

The first one is a purely geometrical one, which rules out implausible trajectories by filtering trajectories by maximum curvature, the rationale being that very sharp bends in the 3D curve are almost always caused by wrong paths rather than resulting from actual measurements.

The second one is the one considered initially (in section 5.3.2), where the angles between the actual reprojected curve (on all views) and the tangent direction estimated from the image gradient

6.3.5 Output

The description calculated in the detection steps consists mainly of trajectories represented as B-splines and a score for each trajectory.

Trajectories themselves can be sampled, by evaluating the B-spline and its derivative at arbitrarily spaced points of its range of definition. The "detection thread" itself closes after outputting such a description (containing times, locations, velocities and labels (number of the particle, score of the whole path) to an XML data stream, which is also saved, together with other results, on the permanent storage medium, with a time stamp.

6.3.6 Image processing libraries

The prototype relies primarily on VIGRA [KM00] as a library of image data types and processing operations; by virtue of supporting generic programming, it was found to be superior to the original candidate OpenCV, whose use soon became unpractical and unmanageable and had to be replaced step by step.

The latter library is highly optimized but suffers from the many disadvantages of its supporting language, C. Switching to the C++ library bestows a type system, significant higher-level abstractions like generic programming, object-oriented programming and functional programming, and obviates the extensive use of opaque syntactical macros even for memory management.

6.3.7 Performance

Performance is to a large extent dependent on the image preprocessing steps. These are trivially parallelizable, because the data dependencies between them are very flat.

For the moment, each frame needs to be Fourier transformed and backtransformed in its entirety ($O(n \log n)$, which is not a problem, but each pixel is revisited later on, even though that is not necessary) in order to determine the mask by Gaussian gradient magnitude; the Gaussian gradient information is preserved and re-used for the 2D keypoint detection and the tangent direction estimation on the image plane.

We judge that it shouldn't be necessary to handle the whole image after the initial masking, and that one could use the Mean Shift algorithm on the original, unsmoothed image and ignore the non-masked regions completely.

Scalability

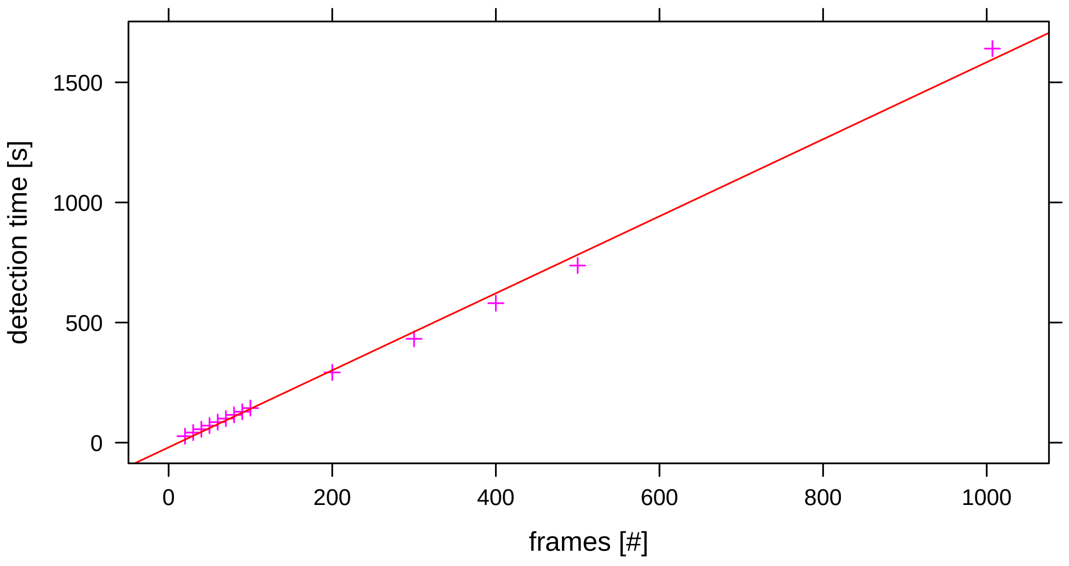


Figure 6.3: Plot of run time against number of frames – average detected keypoints per frame: 50

6.4 Conclusion

The prototype was designed to not only be a demonstration of the system, but to serve as an useable toolkit in the measurement of fluid flows. While its main features are the detection and tracking of particles, along with the visualitation of reconstructed particle trajectories, it can also be used to calculate the position of the cameras in the world coordinate system, and to visually evaluate the quality of the stereo calibration.

The most computationally expensive part of the software, the key point extraction step, is highly parallelizable. Generated trajectories are sampled and made available in form of xml files.

Chapter 7

Experimental results

Having presented the system in the previous chapters, we will now discuss the experimental results we obtained during its evaluation.

We start with an overview of various sources of uncertainty, which affect our measurements. These range from uncertainity induced by geometric limitations of the stereo camera setup, to imperfect estimations of calibration parameters.

The next result concerns the selection of the image scale for trace detection. We will obtain it by investigating the signal to noise ratio of traces in several images at different scales.

We will then consider the performance of the keypoint detector, using images from an experiment meant to reflect extremely unfavourable conditions.

This is followed by a section which evaluates the uncertainty of the depth measurement affecting a stereo camera setup.

We conclude with measurement results from one of our experiments, showing the distribution of particle velocity and the time over which they were tracked by our system.

7.1 Summary of uncertainty

Accounting for sources of errors and uncertainty, especially in an application in which a physical phenomenon is supposed to be measured, is important. One must determine the accuracy and precision of measurements.

There are several sources of error; some are related to the inner scale (linked to the finite resolution of sensor arrays), whilst others arise through imperfection in the estimation of the parameters governing the stereo vision system; a third class of errors are introduced during the interpretation steps.

7.1.1 Limitations of the setup

The proposed setup (as shown in section 1.2) is not without its intrinsic limitations and imperfections.

First, we summarize the sources of error inherent in the general setting, as became apparent during the planning and conception of the system.

Geometric limitations

A two-camera setup for stereo observations always has certain properties which imply certain limitations.

Baseline and angles The distance between the cameras' optical centers, i.e. the baseline, is crucial for accuracy in stereo measurements. Simply put, the larger the baseline the better the accuracy. But the attainable accuracy is of course also a function of location, and of the angle between the cameras; the baseline only limits the maximum attainable "stereo resolution".

One can see that a setup with a large baseline, but an acute angle is hardly any better than the mirrored setup – it is only after factoring in rectification that one can directly compare baselines.

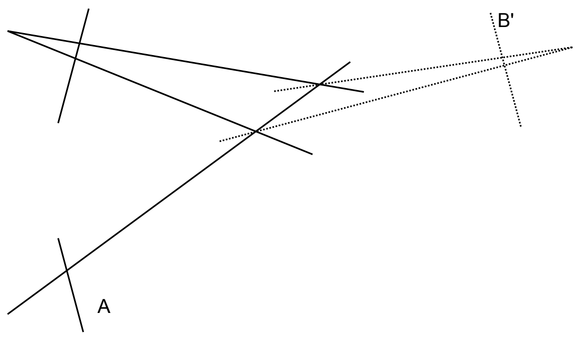


Figure 7.1: A large baseline but a strange angle. One would agree that the baseline, even when scaled with the scene, is not the only parameter influencing accuracy.

Figure 7.2 on the facing page illustrates one of the chief sources of uncertainty, which is always present in a stereo vision setup. The situation drawn here is a simplification

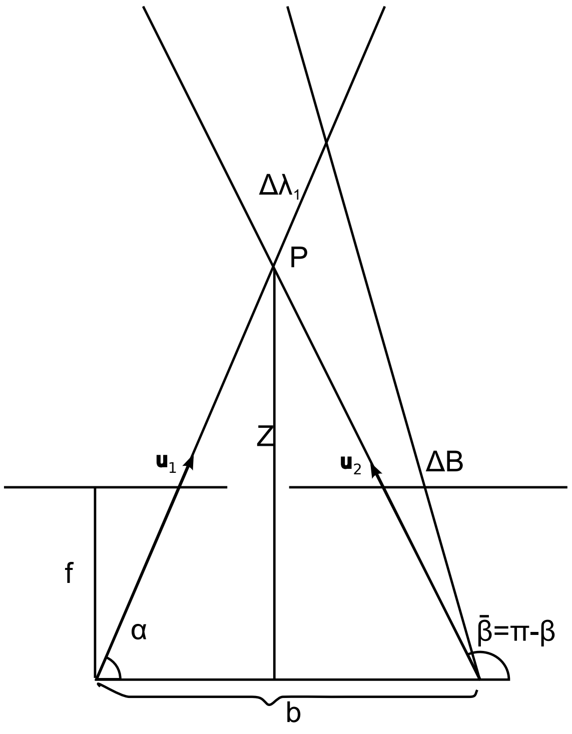


Figure 7.2: Influence of the baseline on accuracy, in a rectified image pair

from the actual setup in being rectified; both image planes are the same and the intrinsic parameters are equal in both cameras.

This can be attained by rotating and rescaling as needed, but in doing so, one must also account for the propagation, according to $T1$ and $T2$. Inside an epipolar plane: calculation of the error of λ_1 from an error ΔB in the right x-coordinate:

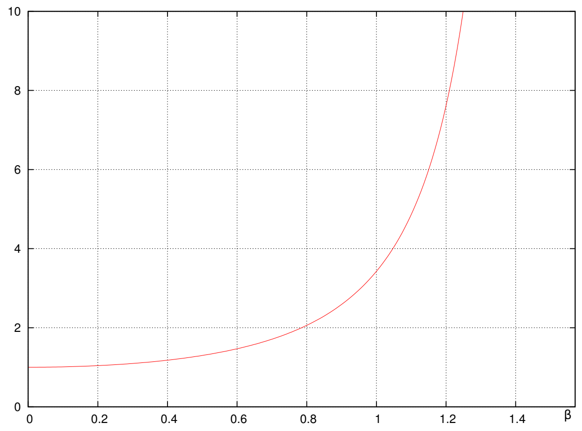


Figure 7.3: $\frac{\partial \lambda_1}{\partial \beta}$ for $\alpha = \frac{\pi}{2}$ (the angles are equal when the optical rays are parallel).

In section 7.4, there is a detailed series of plots which show how the attainable resolution is limited by the camera setup in conjunction with the pixel size, as a function of location relative to the cameras.

Sensor noise

Wherever one attempts to measure physical quantities, there is always sensor noise. It interferes with initial detection step (chapter 4) and distorts measurements.

Then, the use of a camera with a finite aperture [FS07] causes regions of ambient space to be more or less out of focus – with a varying PSF (see section 4.1 for the model of image generation).

7.1.2 Imperfection in parameter estimation.

Especially when starting from the ideal geometric model and applying it to real scenes and setups, one encounters the problem that perfect calibration is impossible,

1. due to the limited applicability of the models and
2. because one must calibrate from limited data.

There is bound to be mis-estimation of the distortion caused by the lens system, because real lens systems are not well captured by the usual models (maybe because they do not model anything, just fit a low-degree polynomial).

There is mis-estimation of the footpoint, of the focal length and pixel scaling. The influence of the intrinsic and extrinsic calibration parameters on the relation between views and space is continuous and differentiable (that is how the parameters are estimated in the first place).

However all of these parameters have a direct influence not only on the precision but on the accuracy of measurements, because bad calibration inevitably leads to erroneous matches.

Noise ratios

It is good to have an indicator for the relative importance of useful information and noise.

Signal-to-noise ratio (SNR), which was already in common use before image processing became a field of its own, is defined either as the ratio of signal power or variance (both constituting an average) $\frac{\sigma_{img}^2}{\sigma_{noise}^2}$, (the term below representing pure noise only), often in decibels because of the large range of values as $10\log_{10}(\frac{\sigma_{img}^2}{\sigma_{noise}^2})$.

Signal-to-noise ratio can probably not be interpreted when the whole image (including the deleted background) is considered, because the useful signal may be not only sparse but actually spatially few and far between, the "signal energy" and the SNR are zero for large images with few interesting points!

It would have to be calculated locally.

Peak Signal-to-noise ratio (PSNR), also sometimes called SNR or Contrast-to-noise ratio (CNR) (see [Köt07]), is

$$PSNR = \frac{A_{img}}{\sqrt{\sigma_{noise}^2}} \tag{7.1}$$

.

PSNR can be more readily applied here, not least because detection depends on the maximum strength of the signal. Uncorrelated noise is, by the Wiener-Khintchine theorem, approximately white noise; so a signal which is too spread-out has a higher chance of drowning in noise.

Overall, the PSNR seems like a good indicator for at what scale structure can be detected; for our experimental findings in this direction, refer to section 7.2.1.

Errors in detection

The following definitions are from signal detection theory (with binary classification), and also follow [Köt07].

Feature detection can err in different ways. We distinguish:

- the precision $prec = \frac{\#TP}{\#TP+\#FP}$
- the accuracy $accur = \frac{\#TP+\#TN}{\#TP+\#FP}$

- the sensitivity $sensitivity = \frac{\#TP}{\#TP+\#FN}$
- the specificity $specificity = \frac{\#TN}{\#TN+\#FP}$

But: the system is designed so that impact is not large. Missing features in the first place would be the worst case of error, beside bad accuracy and multiple reporting of features.

So we may use detectors of great *sensitivity* at the cost of *specificity*.

7.1.3 Keypoint localization

The following short discussion is pertinent if one regards only localization on the image, not on the space of possible locations *and orientations*, i.e. not the 2D tangent estimation. It is also very qualitative.

Error during feature point extraction should not normally exceed greatly the apparent radius ρ of a particle, provided the observations are not troubled by a very structured background.

The “cornerness” detector shows very good localization in the experiment (in pictures: section 7.3) and this is not surprising: as there are no sudden changes in brightness (there is little gap between two exposures), the resulting local structures must look rather symmetrical.

For “line-termination” like features, it seems clear that if the feature is specific (which it should, because the largest amount of positive curvature is seen there), even the chosen scale does not have a big influence on the localization (also confirmed by experiments); for motions along optical rays or nearly so, the distance on the image is very small anyway

Of course, Gaussian Scale Space is a global affair because it is engendered by a Parabolic Differential Equation, and remote influences can definitely lead to an unpredictable (but not uncontrolled) deterioration of localization.

Empirically, we can say however that this is seen first in the tangent direction estimation, which is much “worse” than the spatial localization.

7.2 Statistics

7.2.1 Scale selection

In accordance with the scale-space paradigm ([Lin94], [KF01]), which stipulates that image information ([Jäg95]) is to be viewed with respect to the inner scale of the image, we turn to linear (Gaussian) scale space and execute an experiment for determining whether there is a characteristic scale at which the particle traces (see4.1 on page 63) are especially well discernible.

Even if not actualizing an explicit scale-space representation, one should always try and detect image features at the appropriate scale.

We plot the PSNR (peak signal-to-noise ratio, section 7.1.2 on page 147) through scale-space; for determining it, we manually selected all particle traces visible on a real image. They were relatively isolated from each other, which makes a comparison between their behaviours possible.

We calculated the PSNR against reference regions of interests (patches of residual background) presenting statistics similar to the background of the selected particle ROIs, and also tried the same against the standard deviation of the whole background as a noise estimate (as per the consideration why SNR, as opposed to PSNR, is impossible to interpret here).

The following particles were tracked on an image from the *V2* series. Names and descriptions, so one can follow the development on the plots: A is a very bright trace, B, C, D and F are average, E is somewhat weaker, G and H are extremely weak and K is very elongated though not bright (a fast-moving particle).

The “whole image” region inherits its peak properties from the salient (by their brightness) features.

G and H’s PSNR are, at their peak scale, barely higher than the (corresponding) background regions’ PSNRs; weak contrast should be avoided, because it makes selection of regions of interest harder.

One could argue that brightness, the local property considered, is not the most useful feature for distinguishing regions corresponding to traces from regions corresponding to background; while it is true that spatial coherence is disregarded here, peak brightness is not useless; when the ratio of peak brightness to average noise magnitude falls below a certain threshold, the information can almost be regarded as submerged at that scale.

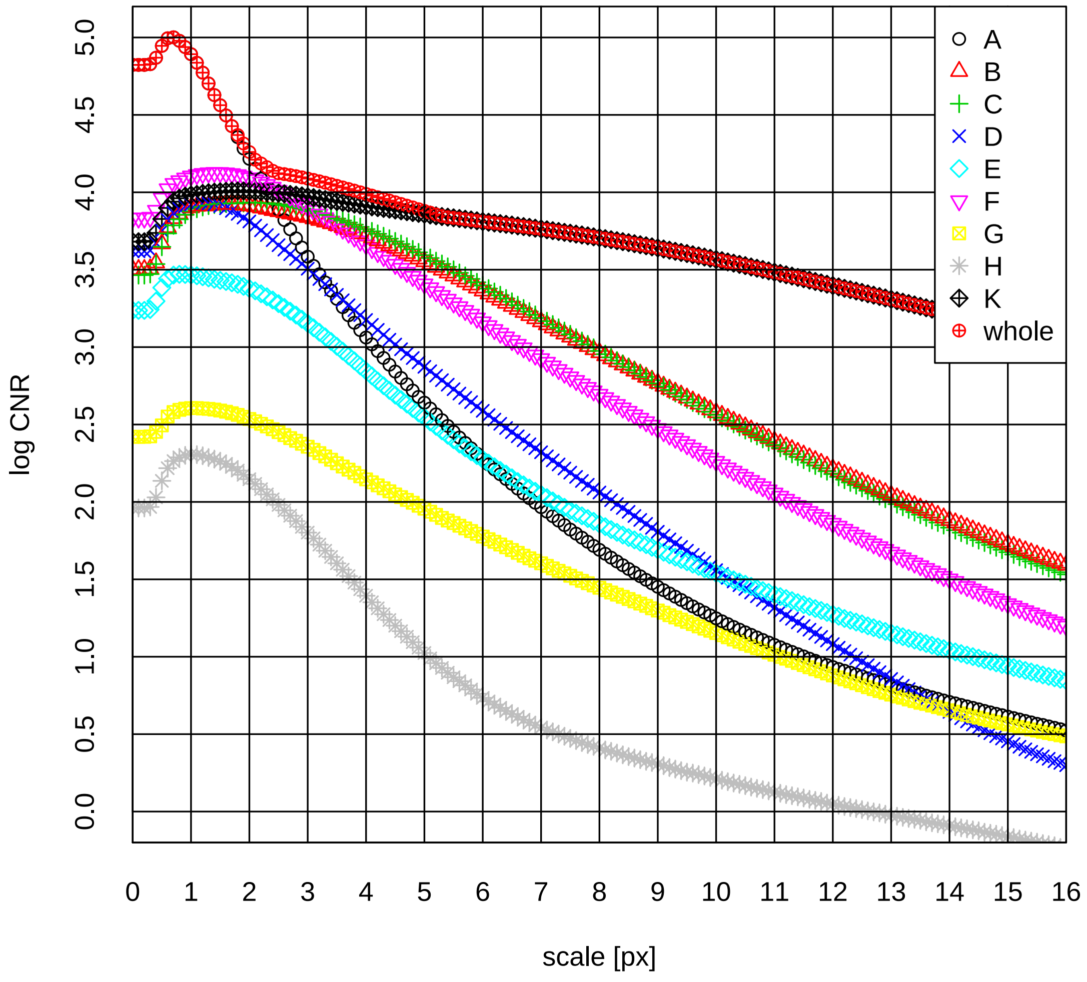


Figure 7.4: PSNR (against the whole image). Most peak at scale 1.0 to 3.0; only the very bright and compact A peaks immediately.

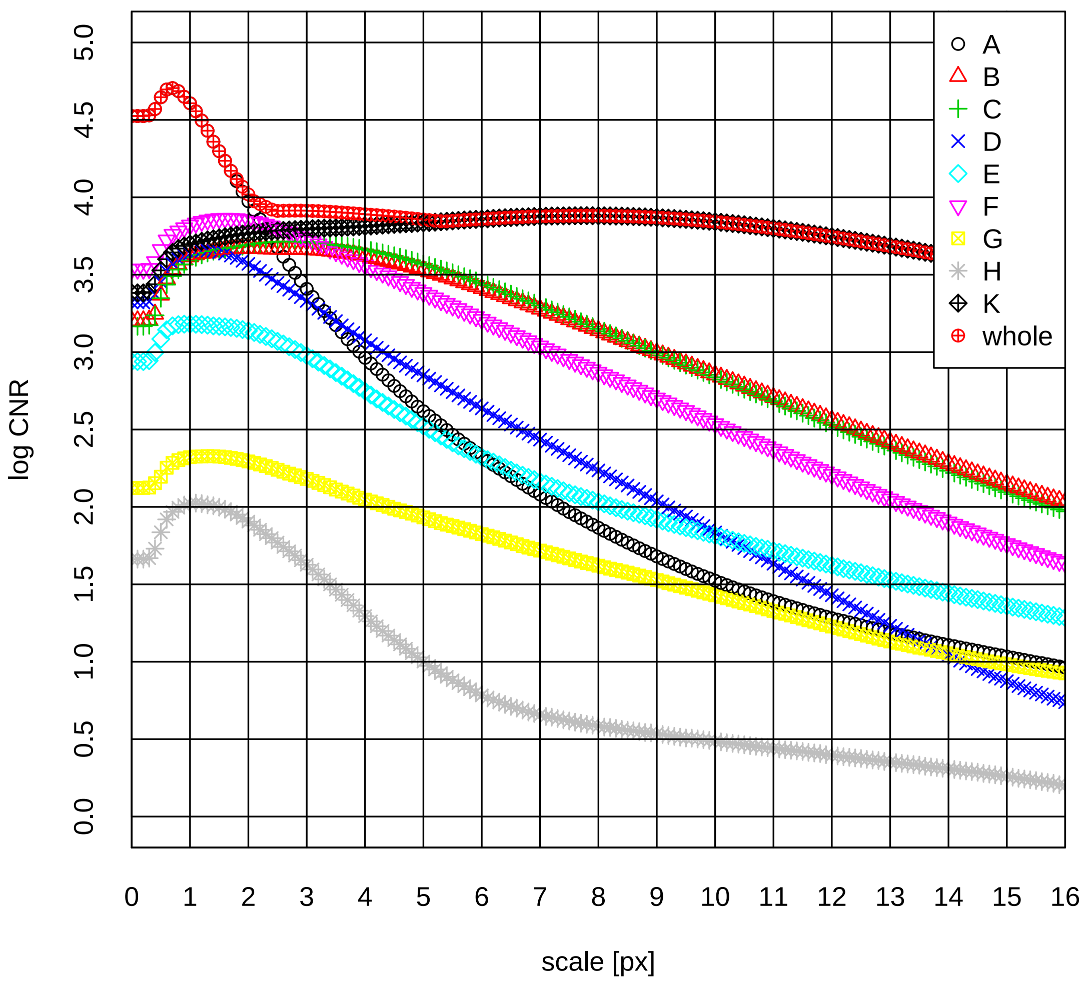


Figure 7.5: PSNR (against a background with strong noise) – the elongated trace K peaks at a very broad scale.

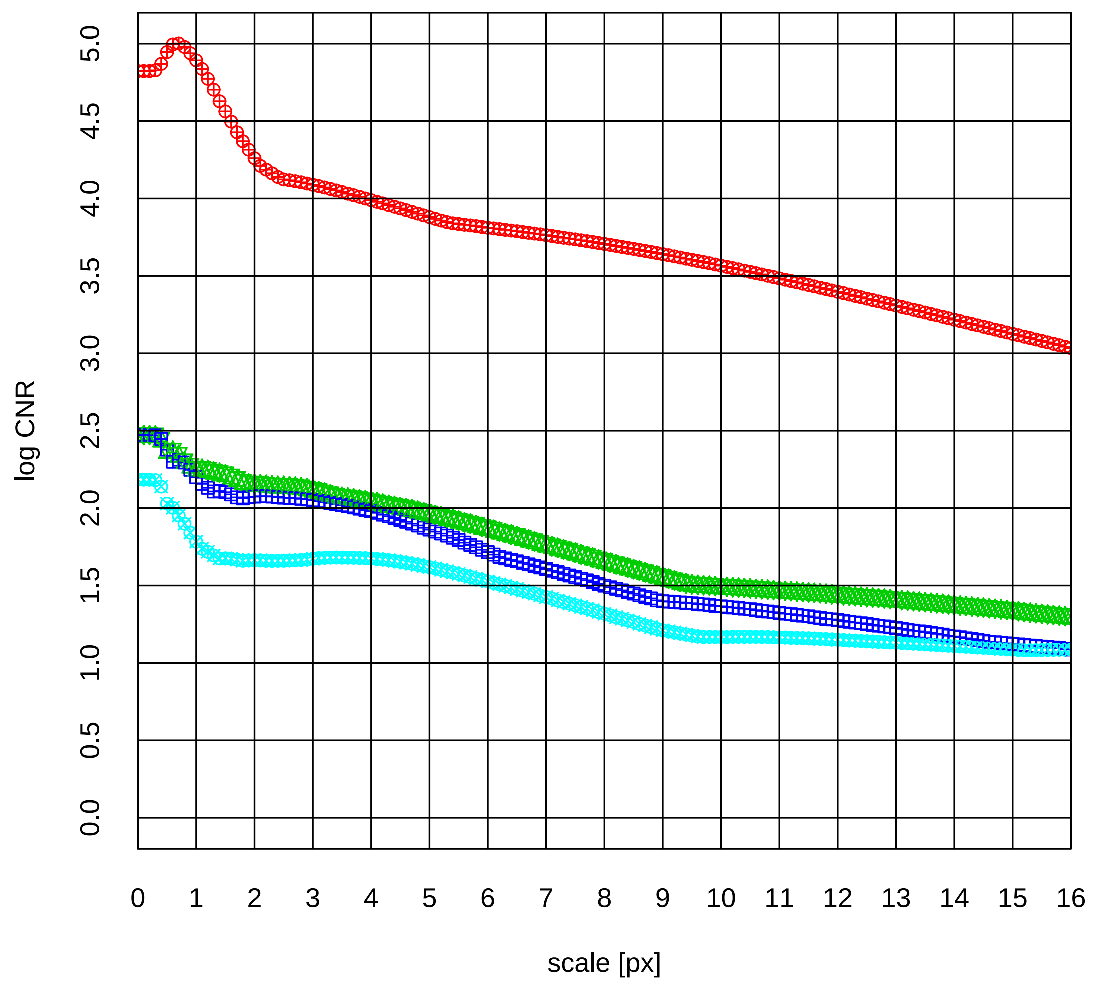


Figure 7.6: Compare: PSNR of background regions, and of the whole image, against the whole image. The top curve represents the whole image; compare also 7.4 on page 150, where it also figures.

7.3 Performance of the keypoint detector

Experiments with the "cornerness" keypoint detector are encouraging. Figure 7.7 demonstrates that it can perform well and mark out locations accurately even in atypical situations, where local differential properties go topsy-turvy.

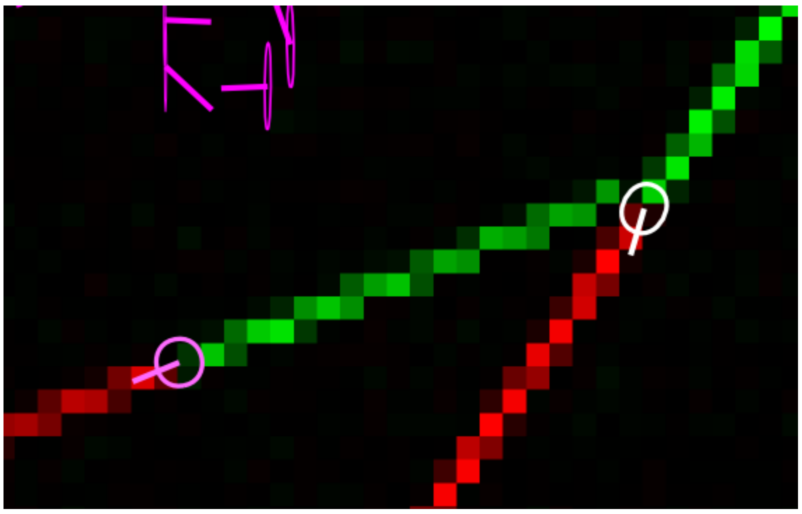


Figure 7.7: Detection in a difficult situation; inner and outer scale for structure tensor: both $1.5px$; post-smoothing also $\sigma = 1.5$.

The following images (figures 7.8 on the next page, 7.9 and 7.10 on page 155) all show the result of the keypoint detector from section 4.4, applied to successive background-subtracted frames, with Mean Shift masks applied.

Without the masks, there is a clear pattern in the feature maxima which are marked on the image: they do *not* cluster near legitimate features, instead there is a lesser number of them around "correctly" detected features and within the space covered by particle traces than in pure background regions. Such behavior is extremely commendable in a feature detector (as per Canny's [Can86] second criterion), and we assert that it can be explained when noise is supposed to be additive, in terms of the surface curvature.

The gradient direction at an appropriate scale is also drawn. It sometimes goes wrong (see lower part of figure 7.12), which could lead to a systematic error. There is room for improvement here.

Next, some images to show how our keypoint detector fares in difficult situations (lots of particles occluding each other and running alongside each other).

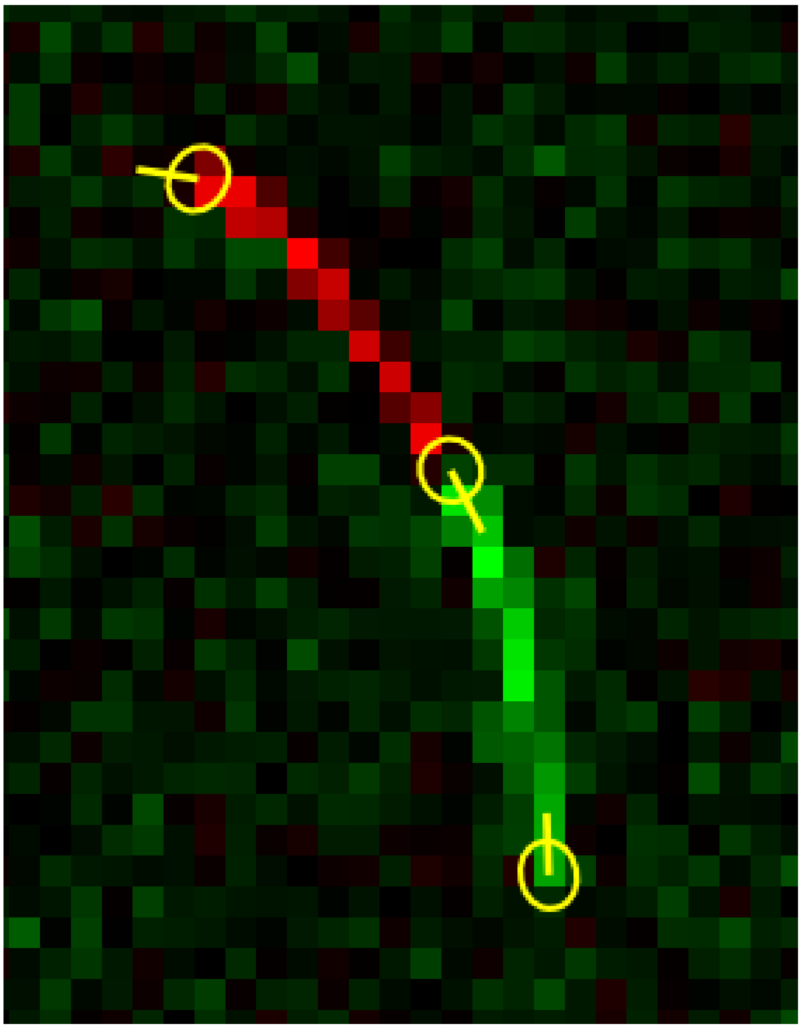


Figure 7.8: *Synth4*: After Meanshift masking, only three maxima remain.

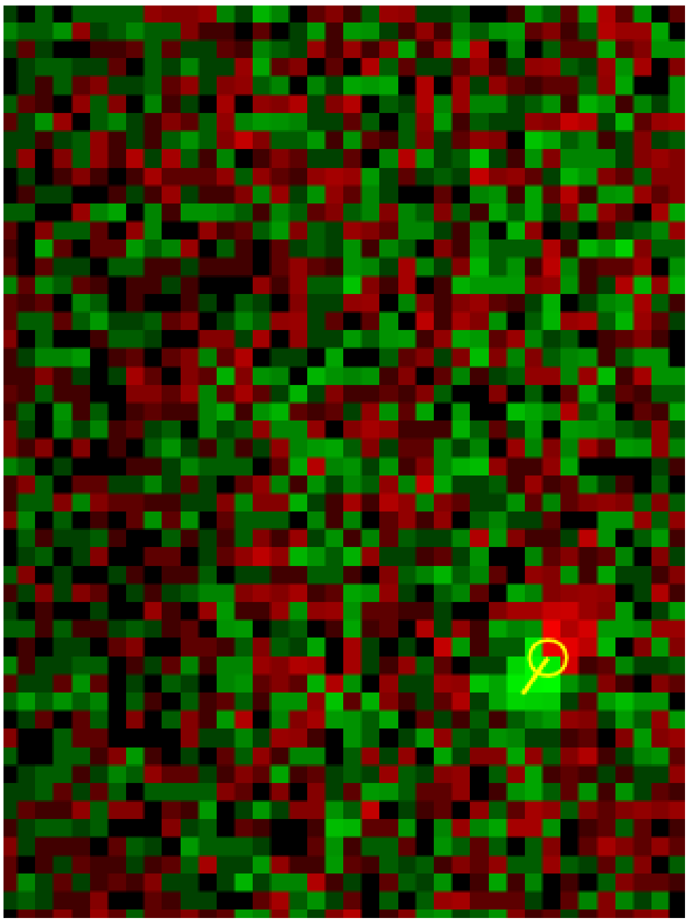


Figure 7.9: Excerpt from *V2*, left camera; acceptable detection of very small, almost spherical particle trace

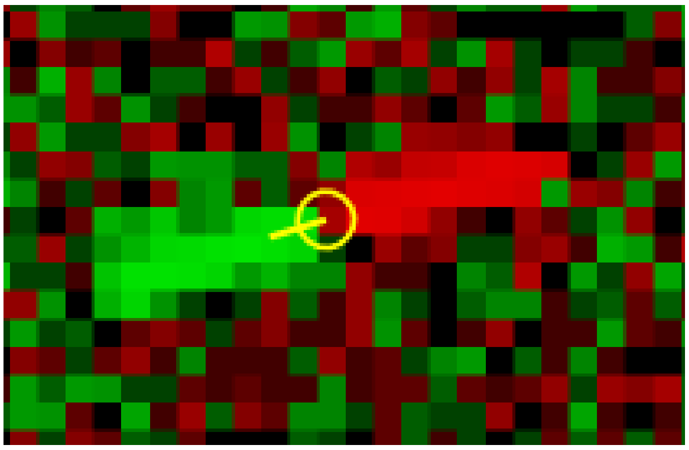


Figure 7.10: Excerpt from *V2*, left camera, frames 1126-1127: 17 possible detections with 9 hits out of 11 (the remaining particles were probably unmasked (too weak))

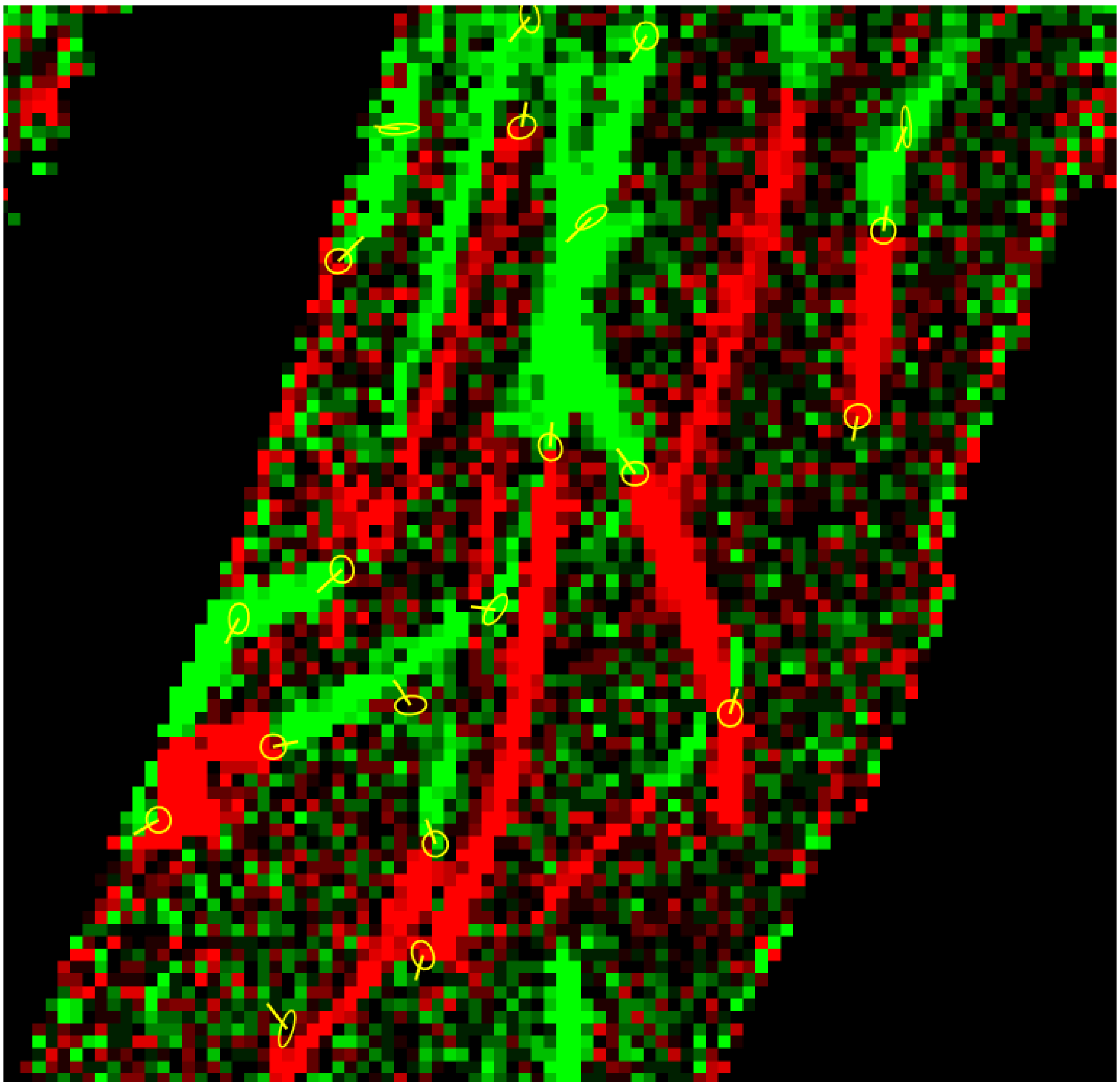


Figure 7.11: Excerpt from *V4*, left camera, frames 8230-8231: 119 possible detections with 42 hits out of 50 unambiguously visible particles, in a very crowded frame (the major part of the undetected particles were found, upon inspection, not to have been masked by Mean Shift.)

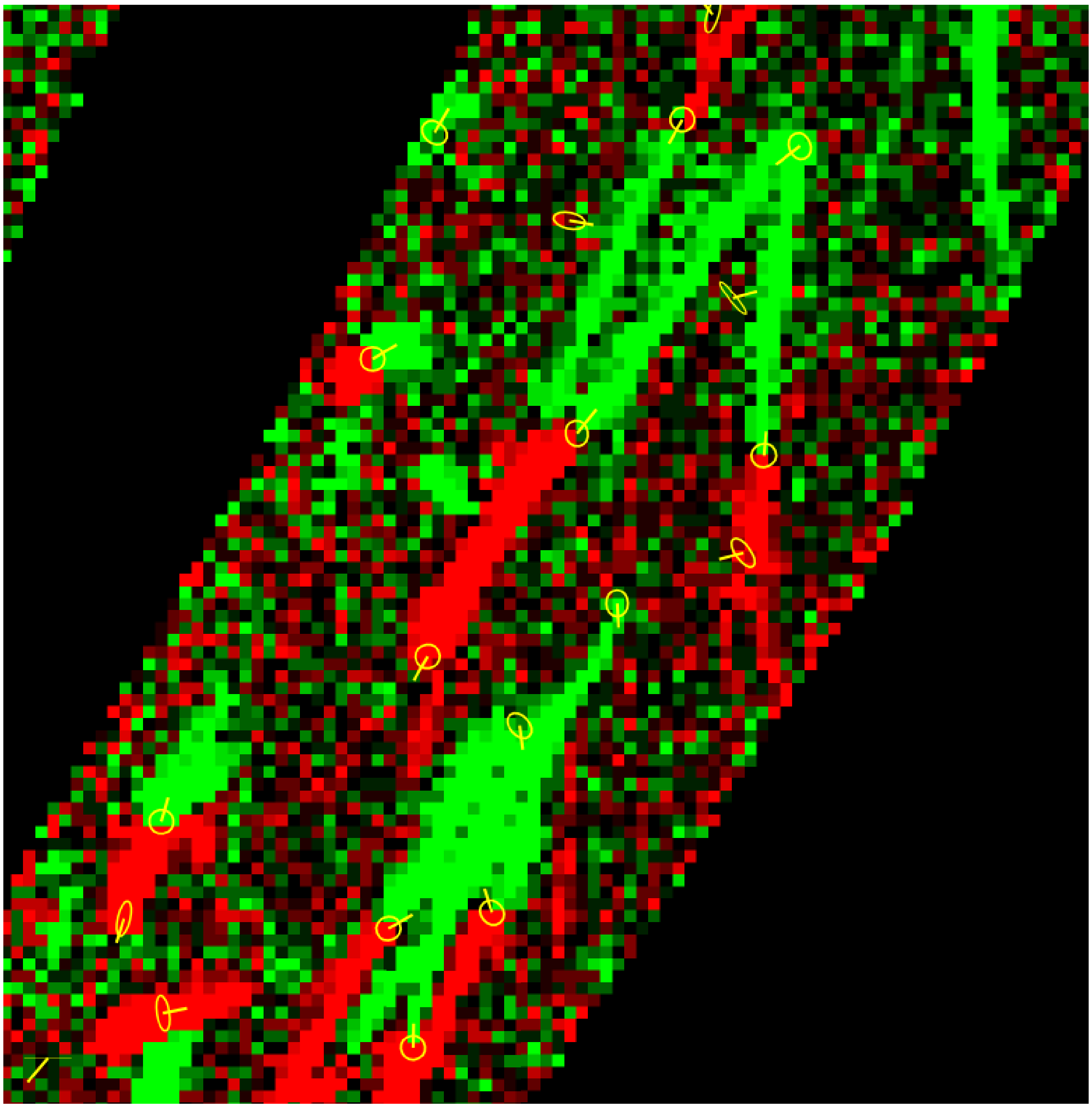


Figure 7.12: Excerpt from *V4*, left camera, frames 8230-8231. Note the accurate pinpointing of locations even in badly discernible places.

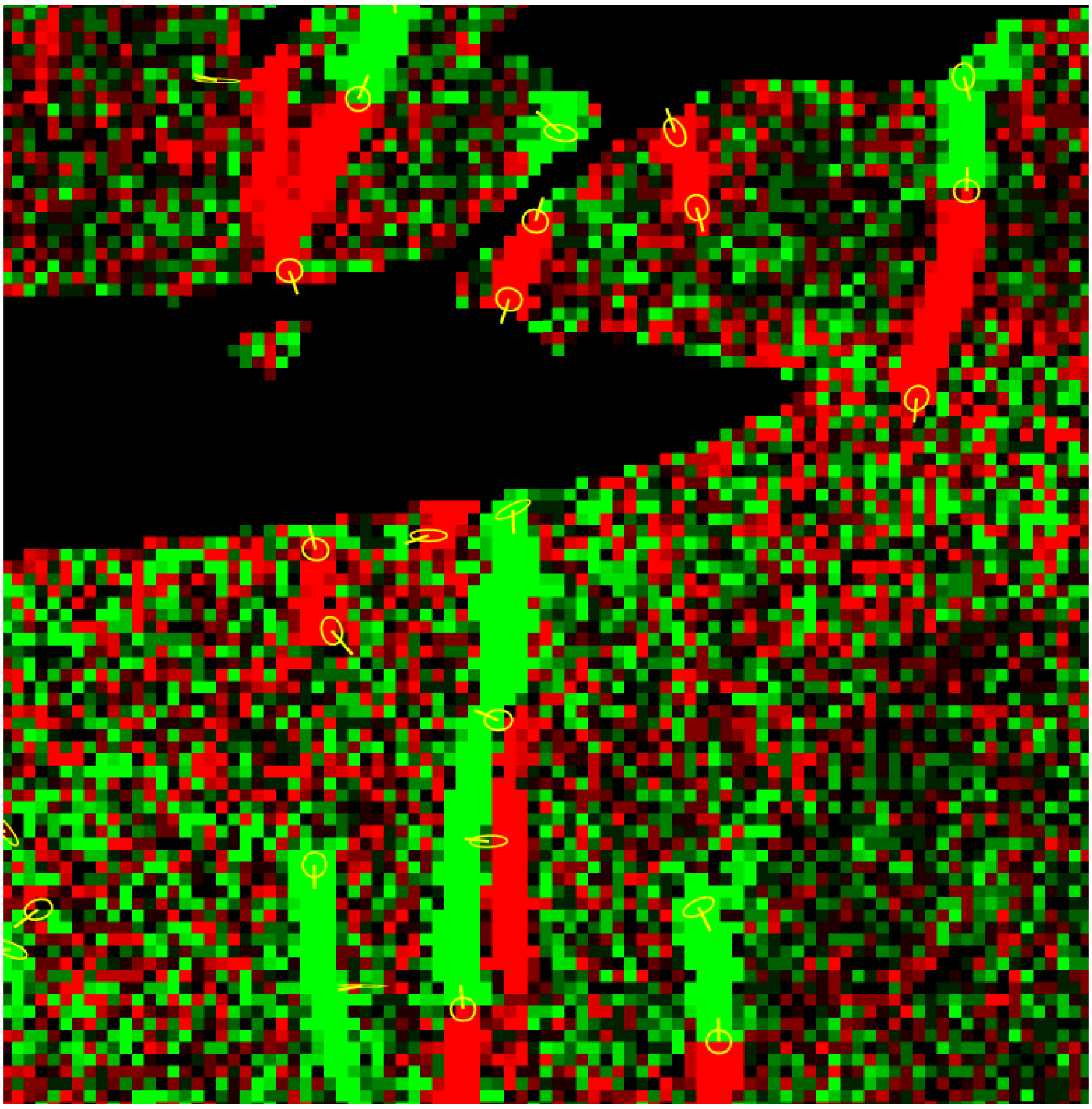


Figure 7.13: Excerpt from *V4*, left camera, frames 8230-8231.

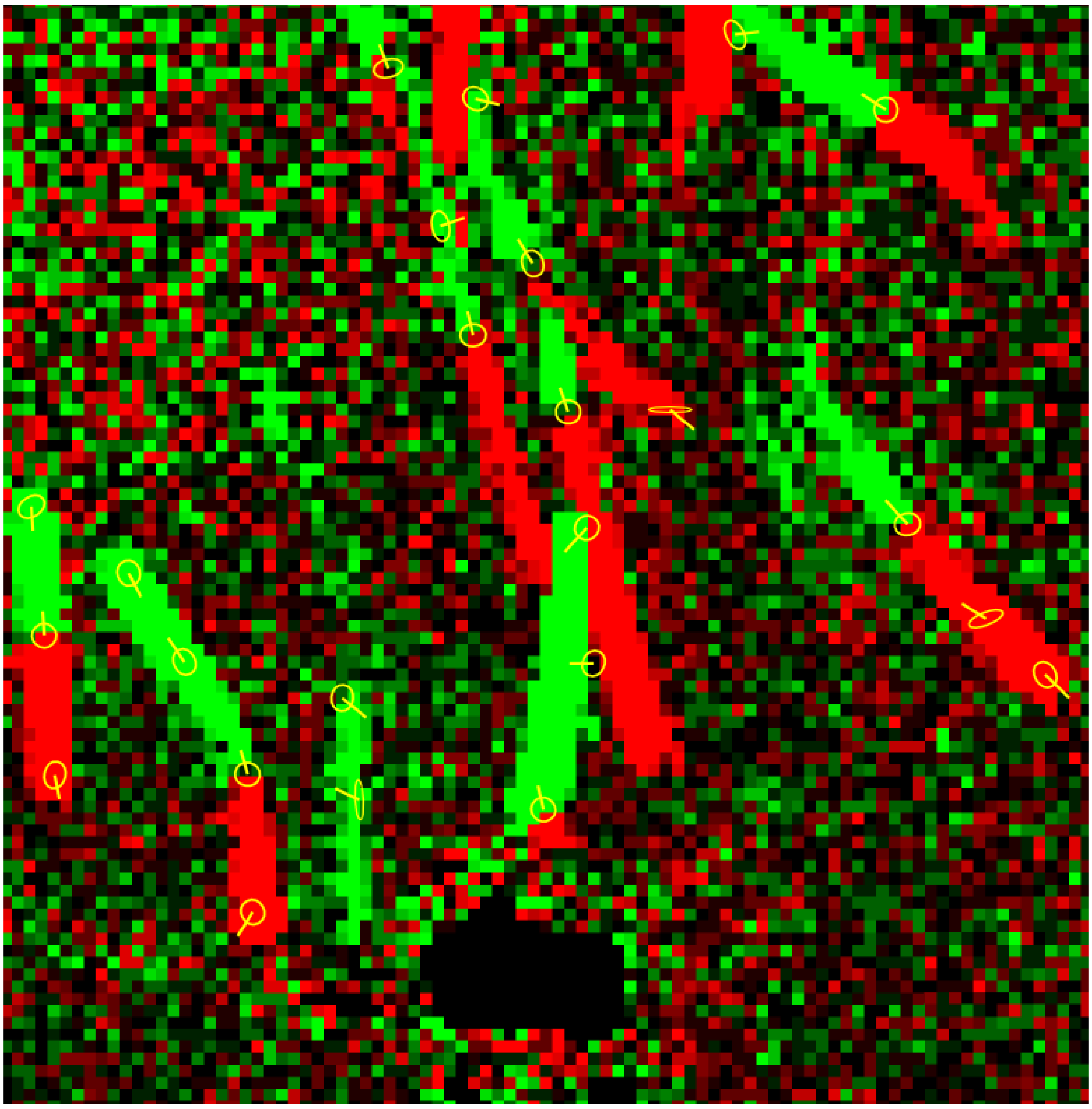


Figure 7.14: Excerpt from *V4*, left camera, frames 8230-8231.

7.4 Stereo measurements

It is interesting to calculate in what regions of space one can measure with least uncertainty. The plots are based on data from real experiments; the rectified case is also included because it was instructive to see – one can see what would happen in an "ideal" setup.

The plots, except the very last one are always on the $y = 0$ plane. In the case of the synthetic images, that constitutes an actual epipolar plane; for the actual images, it is not an epipolar plane but still "near" an epipolar plane.

The "world" coordinate system is always aligned with the first camera (so that its image plane is the plane "Z"=focallength).

Behold first (figures 7.15, 7.16 and 7.17 on the next page) the ranges which are covered by the image sensors. All illustrations in this section have been created with the SAGE metapackage [Gro].

The second set of illustrations (figures 7.18, 7.19 and 7.20) show "disparities", which are pixel distances between corresponding points. Disparity maps are traditionally used in stereo reconstruction of dense scenes (as introduced by Birchfield [BT99]). A rapidly changing disparity value means greater resolution (since the pixel distance changes rapidly with small changes in location).

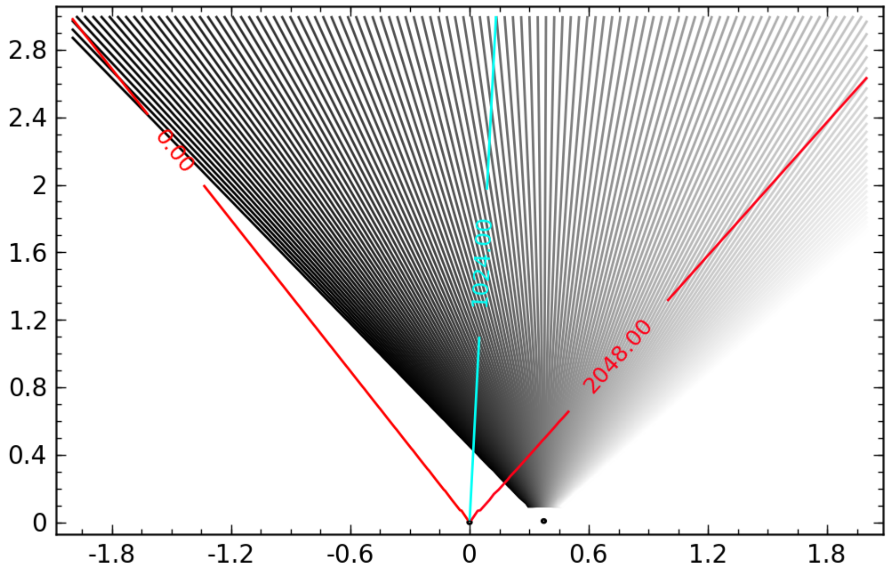


Figure 7.15: Stereo range of $V4$ image experiment ($y = 0$ plane, which is nearly an epipolar plane). Locations of cameras are shown

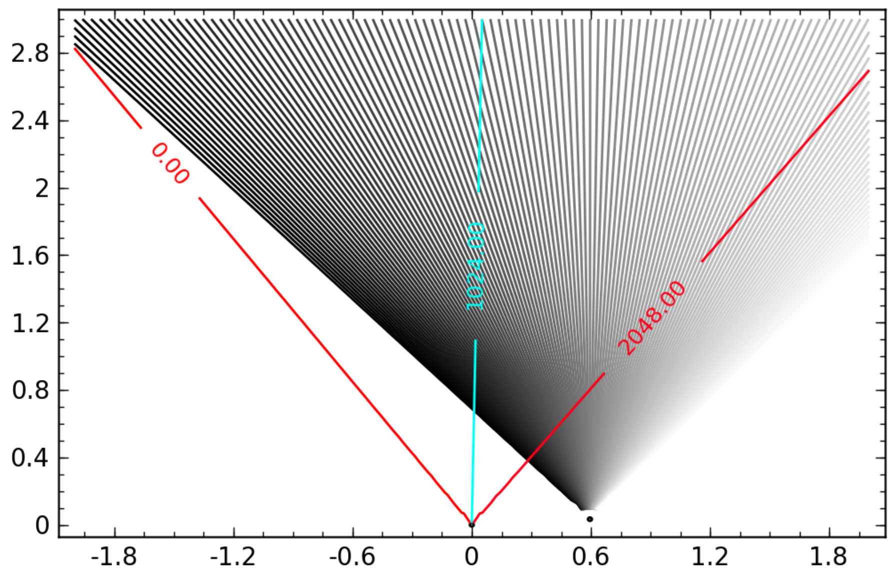


Figure 7.16: Stereo range of $V2$ image experiment. As above.

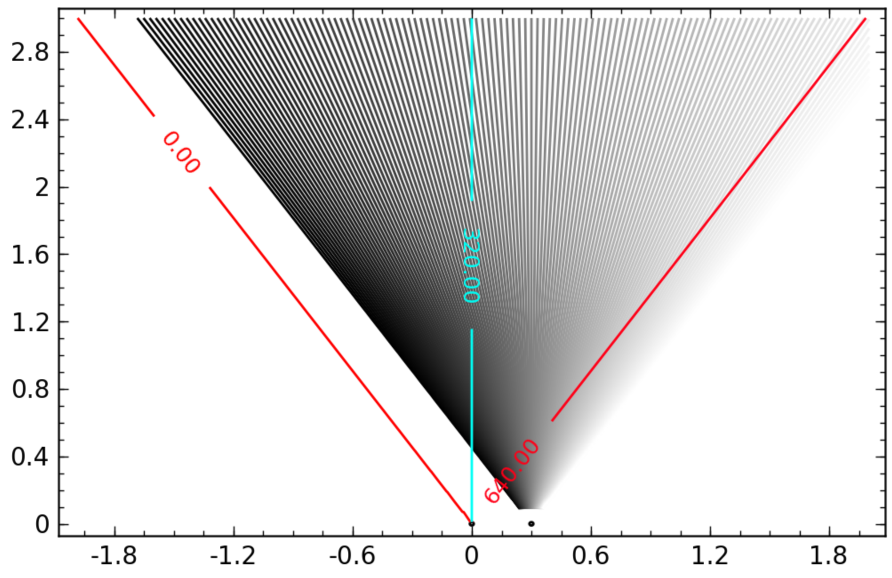


Figure 7.17: Stereo range of synthetic experiments. As above.

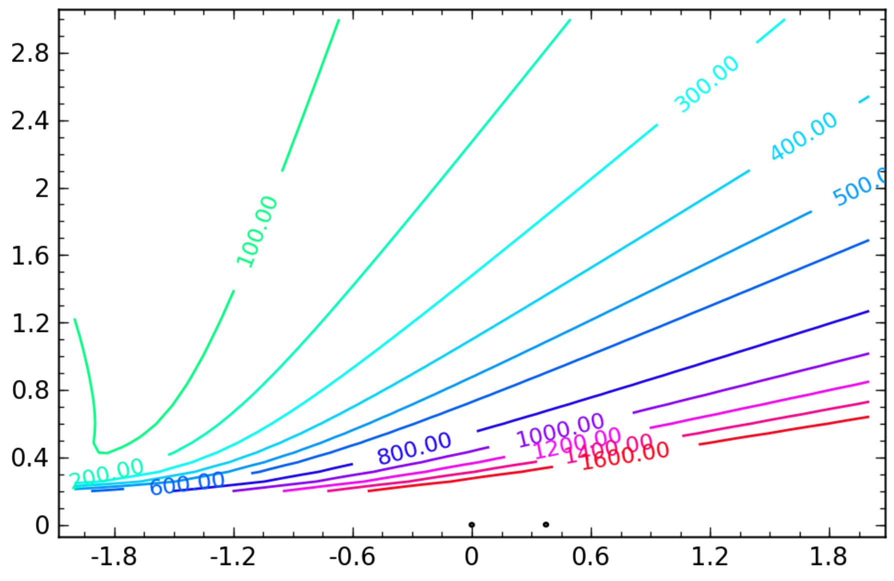


Figure 7.18: Disparity (in pixels) in $V4$ experiment.

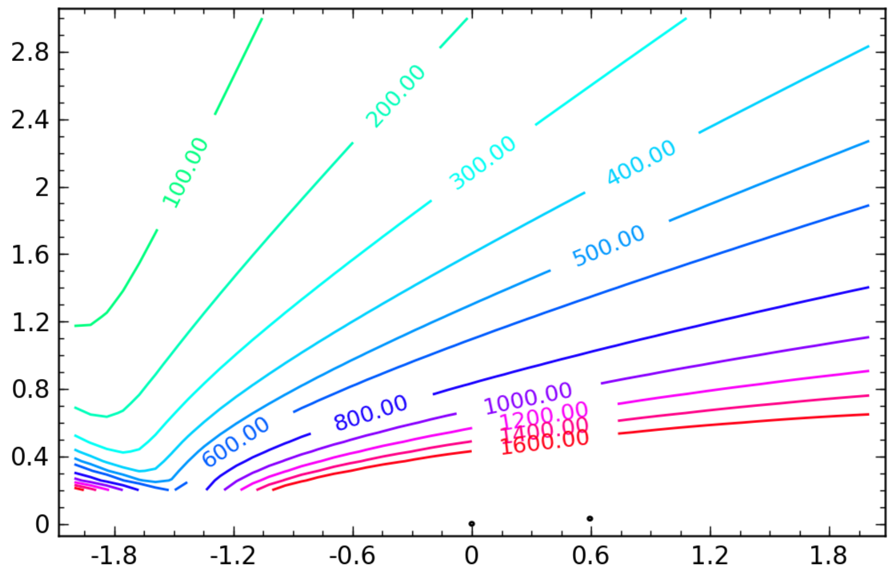


Figure 7.19: Disparity (in pixels) in $V2$ experiment.

Small errors propagating through continuous coordinate transformations, in a first order approximation, are multiplied by the differential of that transformation (a well-known fact). The differential is represented numerically by the Jacobian matrix.

The trace of the Jacobian of the triangulation function introduced in 3.6 on page 56 (normalized by number of target dimensions) is intended as an indicator for the spatial uncertainty in measurements which is caused by the stereo imaging setup: it has length dimension and represents the average scaling of errors. It is shown in the third set of graphics (this time in millimeters rather than meters).

On the other hand, the average uncertainty does not tell the whole story: *depth*, as one could surmise is much less reliable than the directions more or less parallel to the image plane(s). The fourth set of plots deals with the relation between space, Z, X and Y uncertainty.

7.4.1 Conclusion

We conclude that uncertainty arising from the stereo setup in conjunction with the finite resolution of the cameras can be estimated in a meaningful way and maybe even used to optimize the setup.

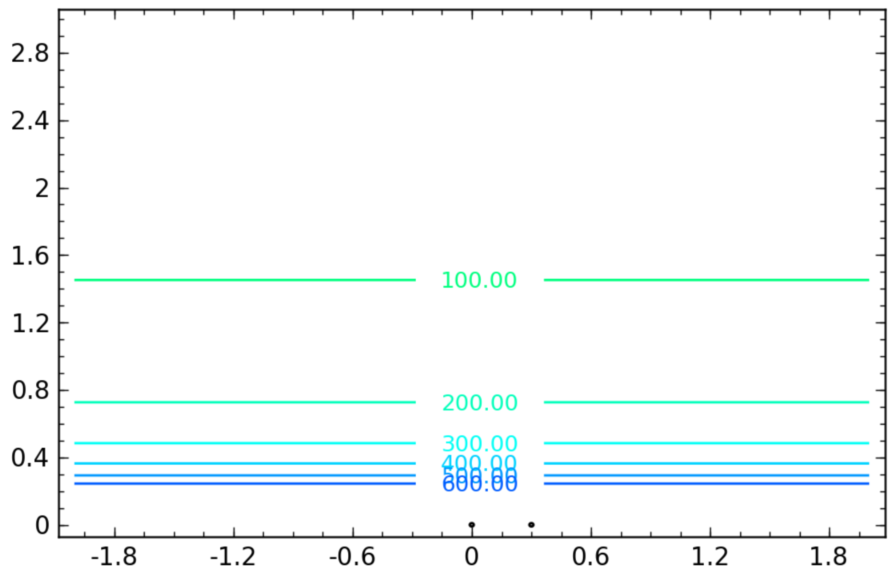


Figure 7.20: Disparity (in pixels) in synthetic experiments. The rectified case is very clean: the lines are parallel; this follows directly from Thales.

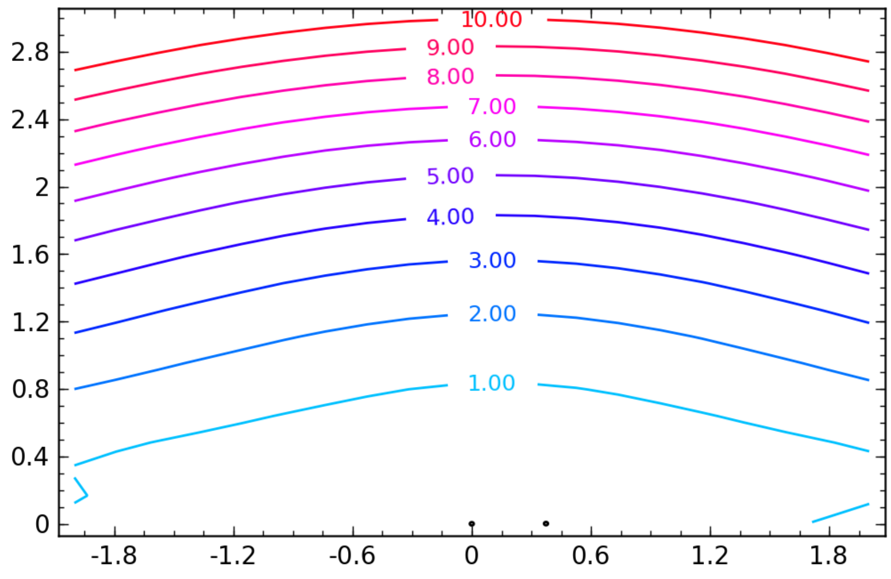


Figure 7.21: $\frac{1}{3}tr(\frac{\partial X_i}{\partial x_i})$ (in $mm \cdot px^{-1}$) in $V4$ experiment.

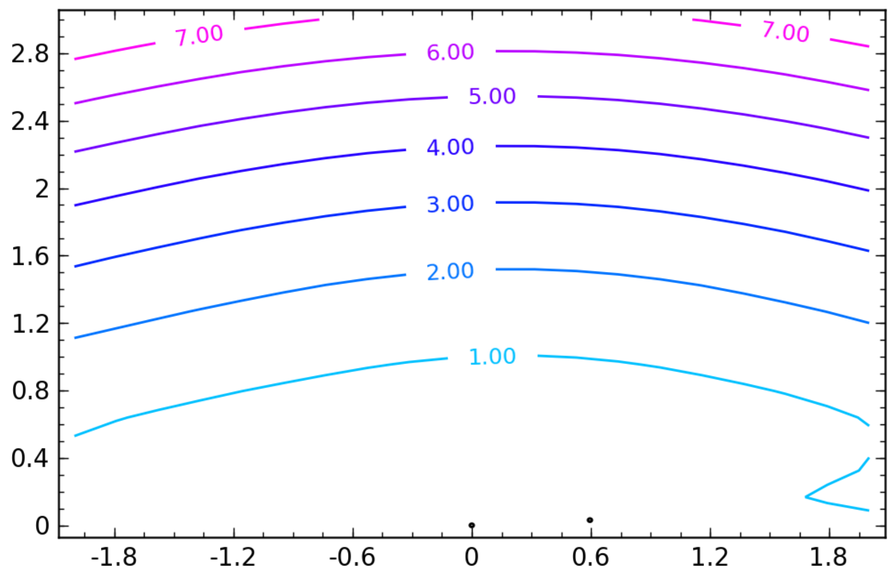


Figure 7.22: $\frac{1}{3}tr(\frac{\partial X_i}{\partial x_i})$ (in $mm \cdot px^{-1}$) in $V2$ experiment.

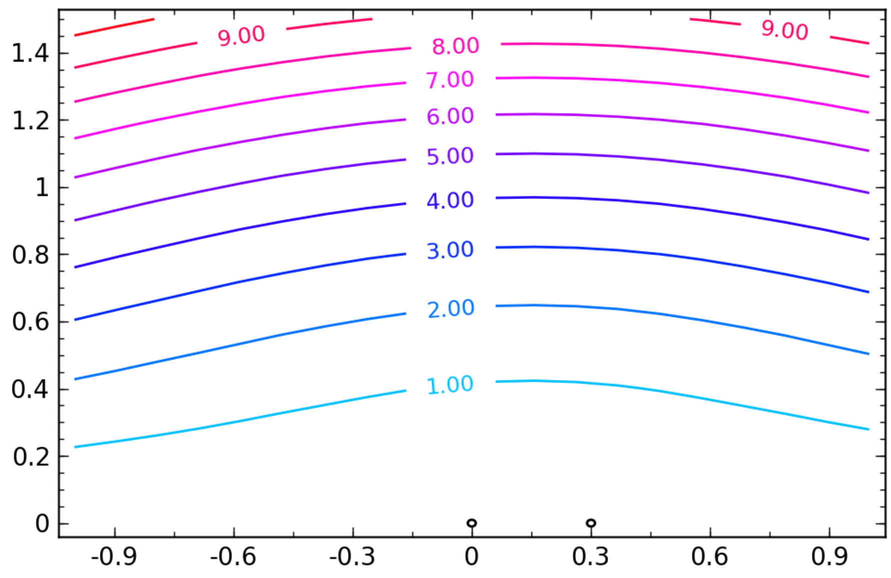


Figure 7.23: $\frac{1}{3}tr(\frac{\partial X_i}{\partial x_i})$ (in $mm \cdot px^{-1}$) in synthetic experiments (note the larger scaling of the axes)

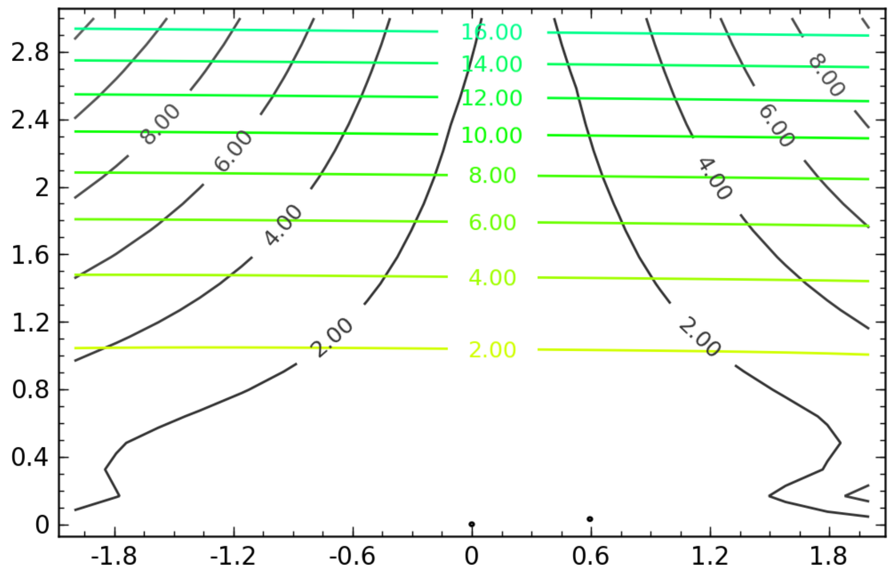


Figure 7.24: Contour lines of $|\frac{\partial x}{\partial x_i}|$ (in black) and $|\frac{\partial z}{\partial x_i}|$ (in light colors; units: $mm \cdot px^{-1}$) in $V2$ experiments. There is a region of good X precision (thus resolution)

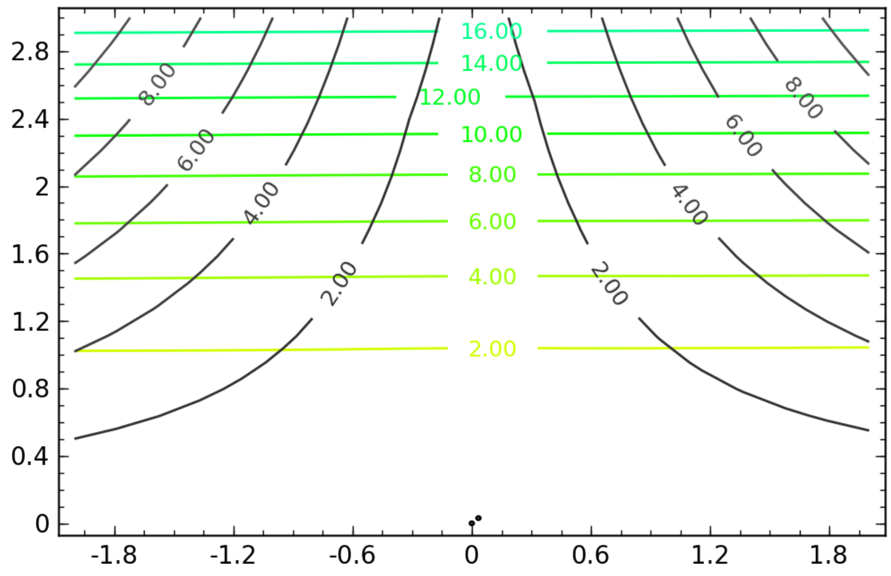


Figure 7.25: $x = 0$ plane view of $|\frac{\partial y}{\partial x_i}|$ (in black) and $|\frac{\partial z}{\partial x_i}|$ (in light colors; units: $mm \cdot px^{-1}$) in $V2$ experiment. Note that the cameras are almost in the same height.

7.5 Particle velocity distribution

In an effort to empirically evaluate the system, we calculated statistics on one of our experiments, which was designed to resemble deployment conditions. The setup to the experiment, labeled, $V2$ is described in section 1.2 on page 5.

As the used particle generator had only crude control mechanisms for controlling the influx particles in the cabin, so the total number of particles in the test environment could not be determined. Figure 7.27 on page 169 shows the particle velocity distribution for the $V2$ experiment.

The average velocity is $0.2659082\frac{m}{s}$ and the median velocity $0.262909\frac{m}{s}$.

Figure 7.28 on page 170 shows the number of frames over which particles were successfully tracked. With a mean tracking length of 12 frames, and a framerate of 7Hz, particles were tracked for about 1.7 seconds, on average. This is considerably lower than the expected lifetime of the particles.

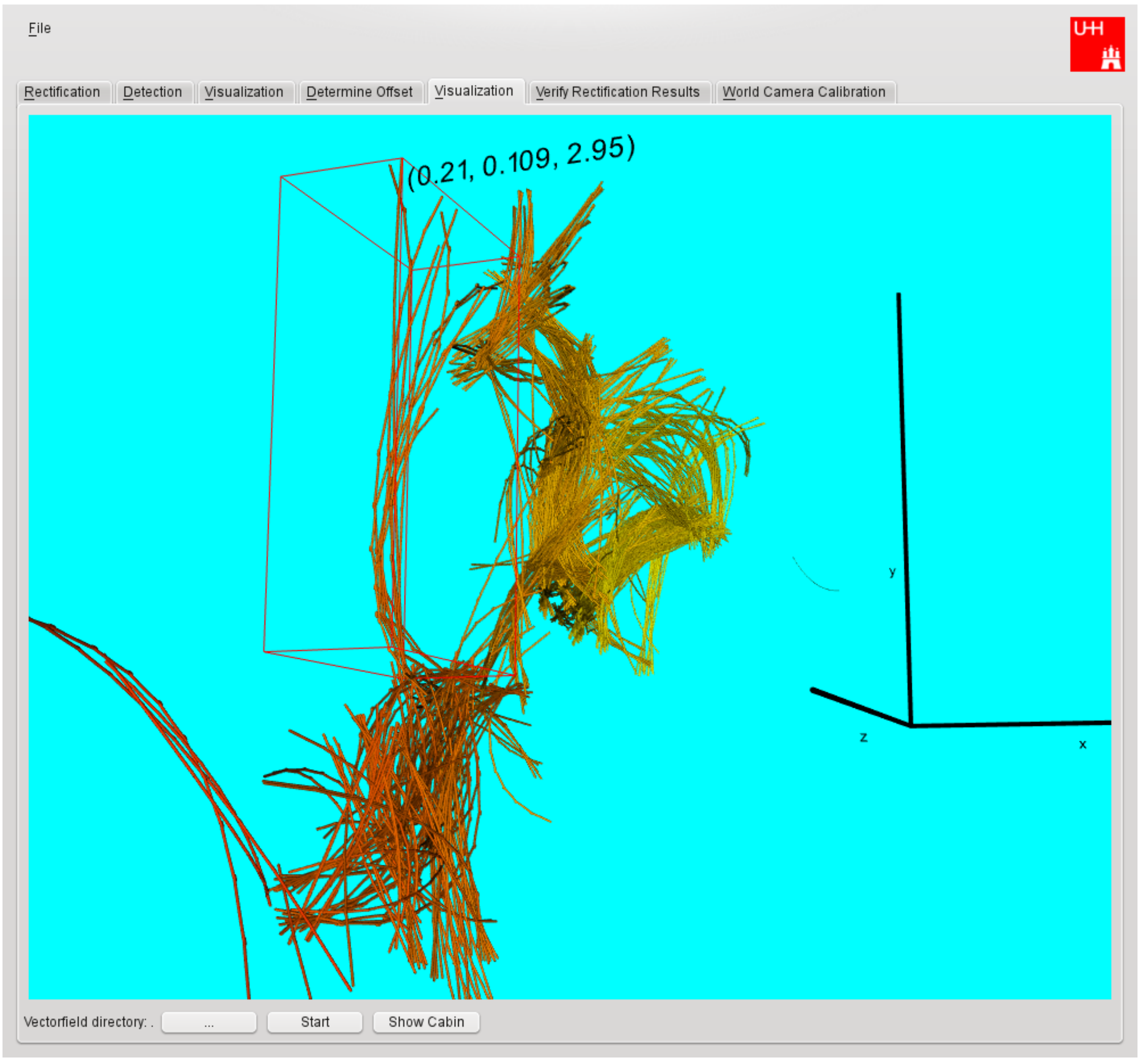


Figure 7.26: Generated particle trajectories from the $V4$ experiment. The scene is viewed from a vantage point which faces the cameras. The text marks the location of a trajectory in the area, which has been illuminated by the row of lights.

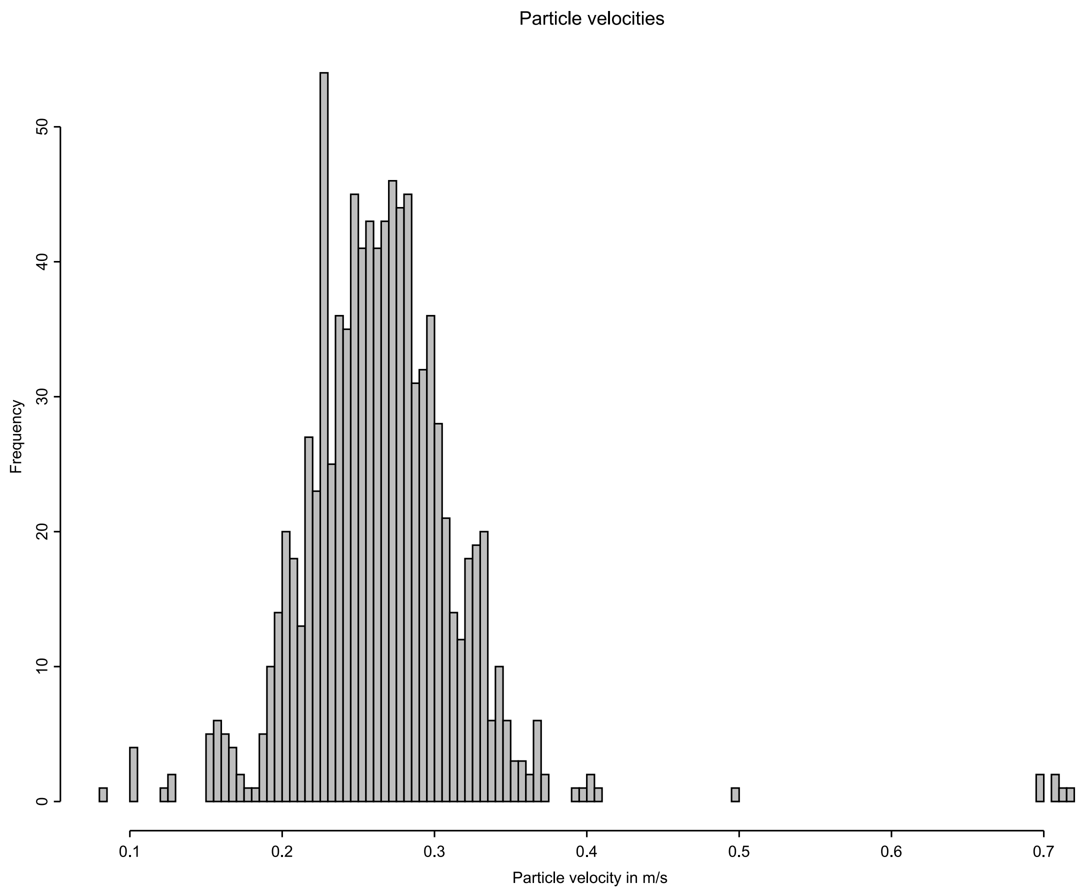


Figure 7.27: Particle velocity distribution of an experiment over 1000 frames sampled into 100 bins.

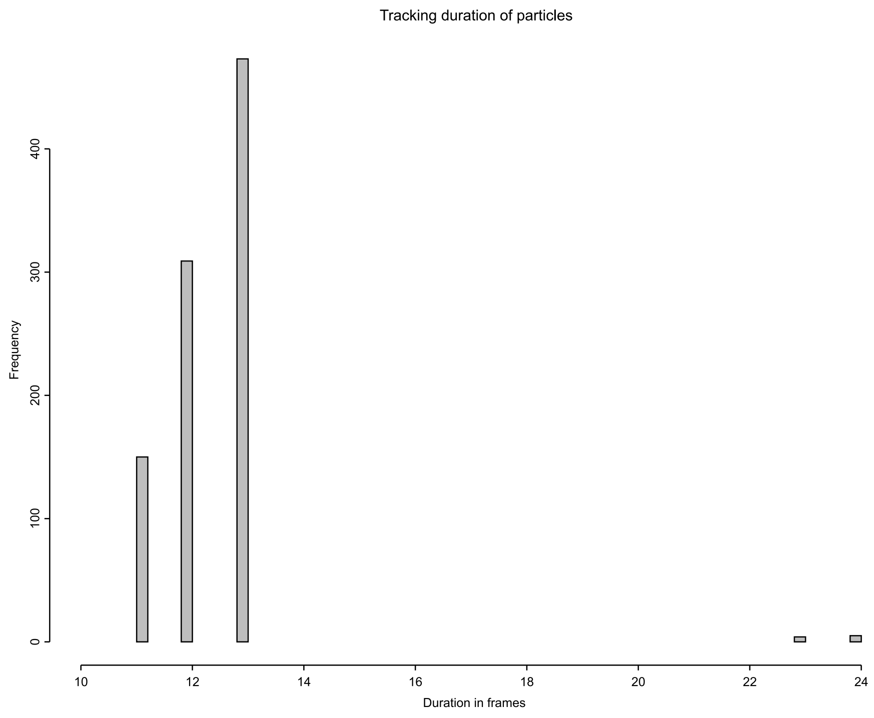


Figure 7.28: Distribution of the number of frames, over which particles were successfully tracked with a framerate of about
7Hz

7.6 Angular particle velocity distribution

Figures 7.30 and 7.31 on the following pages show the velocity distribution obtained by running the prototype on 30 frames of the $V4$ experiment, using a criterion limiting the maximum curvature of the trajectory (so as to exclude excessively bent trajectories, which often come up when joining the wrong candidates). See figure 7.26 on page 168 for an 3D visualization of the measured trajectories. $V4$ presents a field which is fairly dense in a region about $2.6m$ from the cameras, which was near an air inlet. This spatial distribution is visible on figure 7.29 (which shows a histogram of the Z coordinates).

For the statistics, we used the kernel smoothing package for GNU R, by Bowman and Azzalini [BA07].

7.6.1 Conclusion

Even with a very crude criterion, the information extracted from the images seems usable. Figure 7.31 shows the distribution of the directions of motion sampled from the spline curves obtained from thirtya frames of actual experimental data. The azimuthal angle, θ, is aligned with the Y (height) component.

The plot shows consistency with our impression (which is that of a motion with a preferred direction, that is away from the air duct), but the motion was a downward one, overall, which does not figure clearly enough on the plot (also on the backside, the distribution shows weaker, but still clear, concentrations mirrored on the $X = 0$ plane); it is possible that the upward parts are spurious and caused by an error in the calculations.

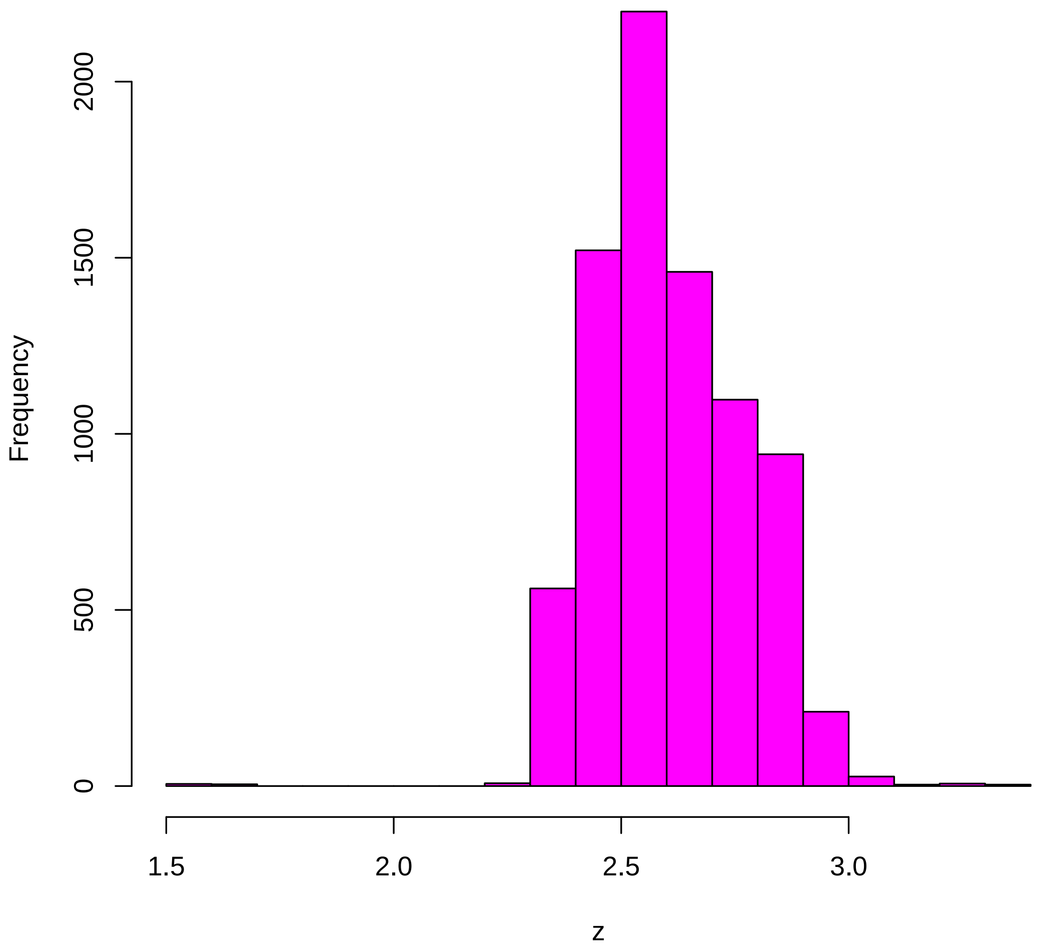


Figure 7.29: Distribution of “depth” coordinate on 30 frames of $V4$ experiment

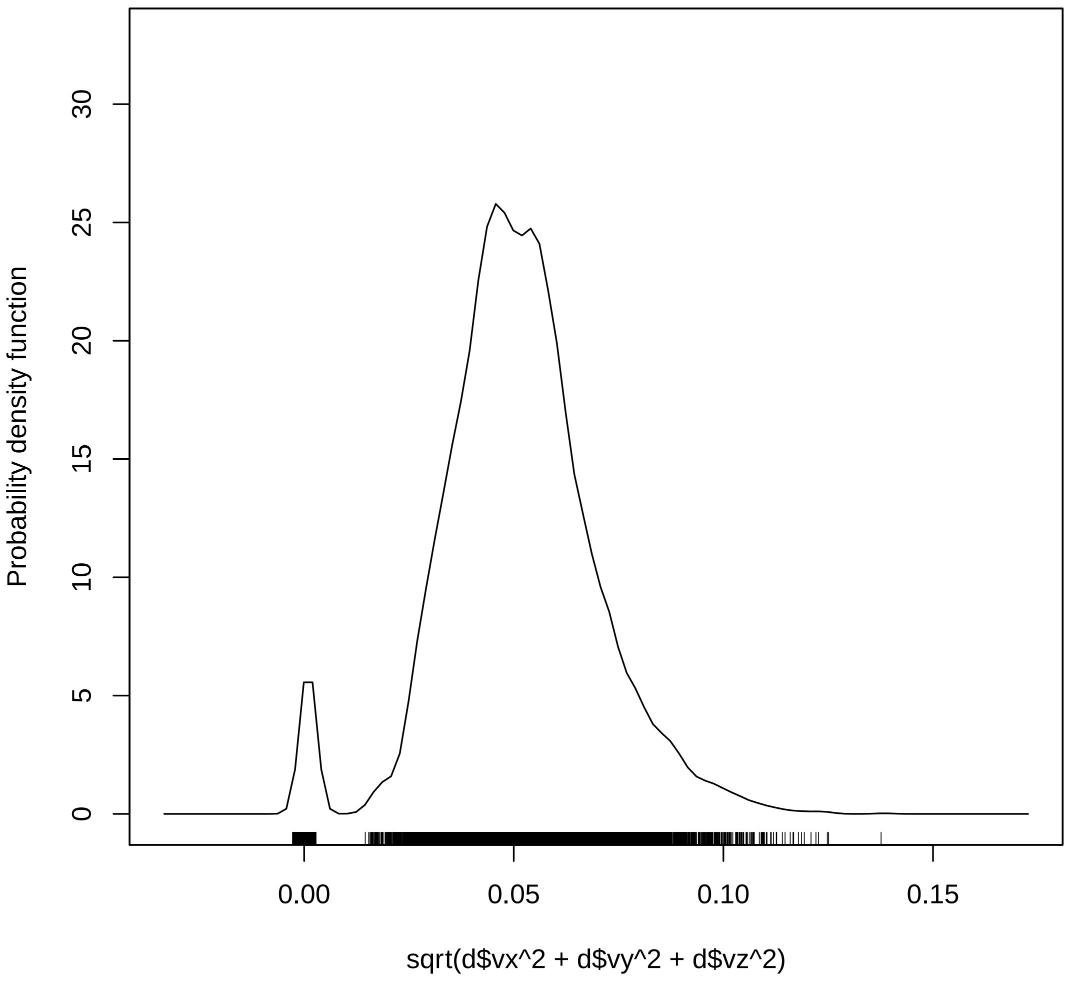


Figure 7.30: Overall velocity distribution on 30 frames of $V4$ experiment; smoothed with Gaussian kernel, $k = 0.002$. The "velocity" axis is in $7m \cdot s^{-1}$, due to the framerate of $7Hz$. The "immobile" particles are probably spurious, and they do not show on the next figure.

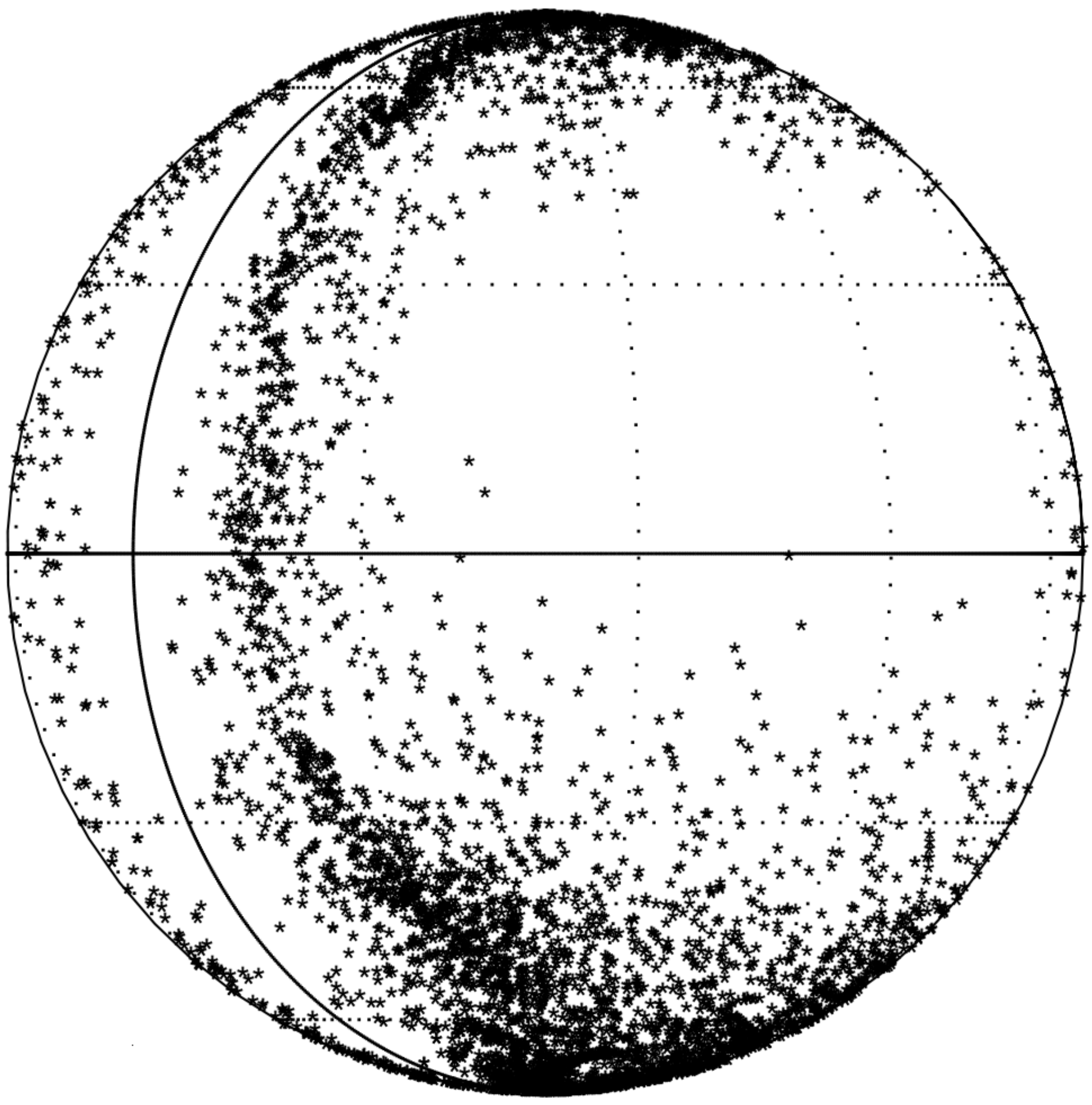


Figure 7.31: Overall velocity distribution as in figure 7.30 on the preceding page; direction is shown. The spherical coordinates are so rotated that the $\phi = 0$ line (shown in bold) corresponds to a motion with no X component, and θ is the arc-cosine of the normalized Y component. The velocities were not weighted, only the direction was counted.

7.7 Conclusion

During our work, we generated a number of experimental and theoretical results.

- attainable accuracy is dependant on both baseline and angles between measured points and optical centres.
- the signal-to-noise ratio for most traces peaks at a similar level.
- the cornerness keypoint criterium is highly sensitive and accurate, which leads to good localization when combined with a Mean Shift mask.
- measured velocities seem mostly plausible, with a low percentage of suspect outliers.
- likewise, recontructed depths are in line with what we expected from the experimental setup.

Chapter 8

Conclusion & further work

Ideas for methodological improvement, some concerning the setup, others the information extraction and the interpretation came up regularly during the preparation of the present work, and there wasn't nearly enough time to investigate all of them depth.

The procedure described in the text is a compromise solution, prepared in a very limited timeframe.

It was never meant as a reference solution, and there is some room for improvements on all levels.

In this chapter some ideas are collected on how to proceed further; some deviate than what strongly from the approach chosen for the prototype but instead are evolutionary: the discussion on the different curve energies in section 5.6 on page 119 can serve as an example here.

Besides obvious extensions like adapting the system to work with more than two cameras, several recalibration and auto-calibration modules, on-line recalibration from known matching and doing away with the synchronization, here are some of the possible improvements, most concerning the latter, higher-level steps:

1. More quantitative analyses
2. Further development of physical foundations
3. Development of more robust feature detection (this pertains to chapter 4) and determination of thresholds as decision limits [DHS01].
4. Development of firmer theoretical foundations concerning information extraction process, including actual estimation of the confidence … maybe aided by the following:

5. Probabilistic interpretations: it would certainly be rewarding to follow through the approach of section 5.4.
6. Choice of a well-adapted curve energy
7. Even further evaluation of all steps of the process

8.1 Beyond particle trajectories

Even in the presence of turbulent motion, the data gathered in the experiment is not quintessentially random – quite on the contrary: there's regularization of very small-scale motion by use of "coarse" sampling via particles (see physical justifications from 2.1 on page 11); most of the remaining motion is laminar in nature and even the turbulence, while unpredictable in detail, follows patterns.

This non-randomness also plays an important role in reconstruction and interpretation, for if minute jitters were important, how could one assign meaning to the recovered data?

The ultimate goal of the present work would be, through the observations, to gain a better understanding of the overall flow patterns.

Of course, having said that, it makes no sense at all to pretend that one could somehow extend the flow information to a dense field. The time-dependent velocity field is grossly spatially under-sampled.

However, if one maintains that such experiments are valid and reproducible, if stochastic (see section 2.2 on page 18), one also has to believe that it is possible to accumulate information about the motions of a fluid temporally, even if one is not in possession of, or cannot obtain a spatially dense field of measurements.

The information one hopes to gather is *eo ipso* entirely statistical in nature.

8.1.1 Statistical properties

The system presented in the first seven chapters is capable of making essentially one-dimensional measurements of a fully $3 + 1$-dimensional spatio-temporal phenomenon. The fact that these measurements are subject to a number of uncertainty factors is known and noted; the uncertainty, as presented in the main text, can be estimated to a certain extent and that information is in principle available for every individual measurement.

An essential aim of the envisaged subsequent interpretation of the results output by our PTV system is a statistical evaluation of the influence of controlled changes in the environment on the overall behavior of flows in the volume of interest.

These are best described not by individual pathlines, which can only serve as the raw material for the projected next step, but by zones of average or statistical behavior, as sampled by means of these flowlines.

An example of a simple statistical feature is whether directions vary a lot at a point. Depending on the nature of the variations, such an observation might point to turbulent flow. If there is not a lot of variation, one might be observing a current which is always there, which is also a meaningful feature to observe.

One must be careful trying to apply the usual statistics to circular dimensions. While the concept of calculating a “mean angle” does make sense under certain conditions, one needs to use the right statistics. Spherical statistics ([Mar75]) are the right means of dealing with such statistics.

However one proceeds afterwards, the collected point samples can hardly be interpreted on their own; one needs to combine them to a probability distribution. There are ways of estimating smooth probability distributions from collections of point samples; kernel density estimation springs to mind ([DHS01], for example, presents Parzen windows as a way to deal with such data).

Kernel density estimation is a process which converges to a smooth probability distribution. As such, it is much more satisfying than simple binning, which only results in ugly staircase functions.

8.1.2 Visualization

Differential properties of vector fields are popular for visualization The second part of [PVW] has an overview of vector field visualizations methods; the techniques developed by Theisel, Weinkauf and others ([TRW, TWHS]), could very well become relevant for further development of the visualization component, even if most of them have been conceived for slightly or totally different applications.

8.1.3 Positioning of cameras

The field of view obtainable in the stereo vision setup is smaller than a that of a single camera.

There is a popular geometrical problem on how to distribute the minimum number of cameras in a space such that every point can be viewed from some camera. This problem occurs always when trying to position cameras in a scene with many occluding elements and it is even more complicated if a certain baseline in a stereo setup is required.

8.2 Improvements: creating sparser representations

We are aware of recent research in choosing good representations for digital images, some which points away from representing images directly as arbitrary pixel sets, distributions, Morse functions ... but instead encode features adaptively.

All of this is rather speculative, and maybe it does not lead to a system that performs better. But the following observation is true and pertinent:

The main performance stopper in the first working version of the software prototype were the large quantities of near-zeroes in the image regions containing no useful information. Before rectification and undistortion (chapter 3) were integrated on a by-need basis (and only applied e.g. to keypoints), the whole images, showing mainly background, had to be run through a warping module, just to be thrown away afterwards.

This quickly led us to consider how such processing of useless information could be avoided – especially since we know something about the structure of the images.

Regularity in the data

Natural images, the subject of study in image processing, are very far from being random.

If a natural image is lossily compressed in a way such that it is not distinguishable (in spirit) from the original, it must be well represented. Of course, our geometry extraction process could be construed as performing such a transformation.

Much of the information in this specific application is spatially concentrated. It could be concentrated into just a few coefficients of a wavelet basis. Generally, drawing on the fact that our hypothetical signal T is sparse, processing could possibly be rendered more efficient by processing only the parts which substantially contribute to the restitution of the signal.

Wavelets are functions which, translated, rotated and scaled, form an orthonormal basis for $L^2(\mathbb{R}^n)$. See [DVDD98] for a classic; the references therein cover all the basics and some information-theoretical developments.

The fact that information is concentrated is already partly captured by the use of masks, but not yet consequently enough! Masks allow us to continue using nearly the same image processing routines we would use on a dense image, and using them was the most likely way to successfully construct the system.

8.3 Lifting the requirement of synchronization

Since the least “off-the-shelf” aspect of the whole setup is the requirement that the two (or n) cameras be synchronized, it is natural to consider whether it can be dropped.

Not all camera systems can be so precisely triggered and the exposure times controlled that the requirement of synchronization is met. It would therefore be sensible to extend the method to the case of non-(perfectly-)synchronous recording.

To synchronize post-factum one could use a light signal whose state (on or off) is equally visible from both views and temporally align its state: this is unproblematic (the task being to estimate well enough the sub-frame instant when the signal is toggled, which is not hard).

But that is not enough; assumptions made on how to match dictate that the images actually be recorded in a synchronous way.

The requirement should indeed be unnecessary as soon as one no longer needs to rely on the extraction of point-like features (except maybe when high-speed frame grabbing is available and curve extraction consequently becomes and easier task).

This requires a deeper look into curve matching; there are good methods already available for that task (cited earlier, in section 3.6.1).

8.4 Increasing performance

The need to test different algorithms for mask generation and key point extraction required a flexible design, which had priority over performance concerns. As both of these image processing steps can be parallelized, they could be turned into threads. Each frame can be handled seperately from all others. However, the potential number of frames far exeeds the number of cores in modern central processing units. To achieve a higher degree of parallelization, distributed processing units, linked via a suitable bus, could be used.

In [MBZ], the authors list several ways in which these units could be implemented:

- as intelligent image sensor, combining sensor and image processing into one chip
- as a dedicated desktop PCs, which would allow the usage of established image libraries
- as embedded processors, located directly on the camera
- or as dedicated hardware, designed for the specific pre-processing purpose

The later suggests itself for the mask generation, as it offers the greatest potential performance, and because the Mean Shift based segmentation is non-parametric ([CM99]), allowing the unit to process images without the need for configuration logic for the algorithm. Such a system could be designed to use a modified Harvard architecture to improve memory access time by using two memory components for shared data and instruction storage.

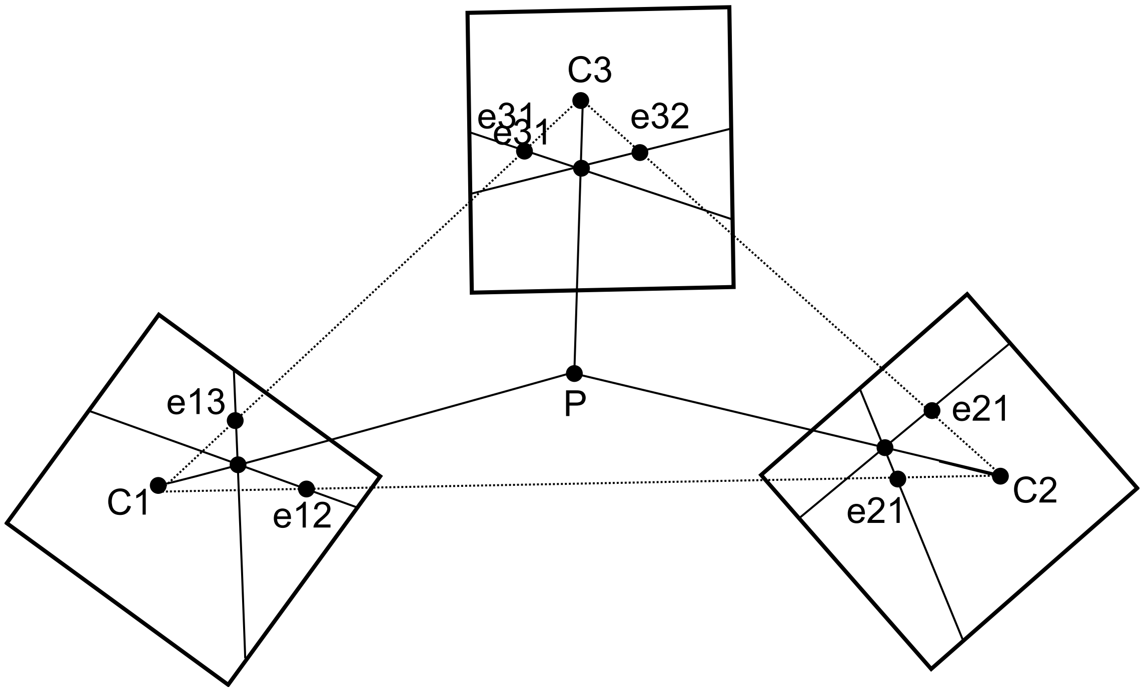


Figure 8.1: Example setup for a tri camera system. In benign circumstances, the projection of a point *P* will be at the intersection of epipolar lines in each image plane

8.5 Oligo-camera setup

This section will investigate the possibility of extending the experimental setup by additional cameras. Required changes to the current system will be discussed along with the advantages of simplifying the matching problem.

In 3D Particle Tracking, tri camera setups are more common than dual camera setups (see 1.1 on page 2). This is because a three camera setup simplifies the matching problem considerably.

With two cameras, the epipolar constraint restricts the matching problem to a line on the respective image planes. If a third camera is used, the correspondences are unambiguous. Even if a search window has to be used in practice, due to imperfect calibration, the number of ambiguities would be vastly reduced. Given that our system requires the construction of paths from all possible stereo matches, this would result in a significant performance gain. Figure 8.1 shows the epipolar geometry of a three camera setup.

The performance of a three camera system degenerates into the behaviour of a stereo camera system whenever the epipolar lines for two views coincidence. For this to occur, the projected point needs to lie in the trifocal plane - the plane which is shared by all three optical centres. In order to avoid degeneracy altogether, a non coplanar setup with at least four cameras is required. For a three camera setup, Maas shows in that in [Maa93] that the ideal way to arrange three cameras for depth reconstruction is by placing them in the corners of an equilateral triangle. A three camera configuration

An additional camera would require calibration, which can be done using the methods we used for the current system (see chapter 3.3 on page 34). The relative orientation between the cameras could be obtained by conducting pairwise stereo calibration. For the rectification, [HR] lists several methods for different setups. However, rectification would not be necessary, as there would be very little need for searching. Mask generation and keypoint extraction would not be affected by the addition of a camera. The reduced number of possible correspondences, however, would yield a dramatically reduced search graph of possible paths to explore.

Rectification is only possible without heavily distorting the images, if the projection centres are not visible in the images planes of the other cameras ([HR]). However, a setup with wider baseline and non parallel cameras could cover a larger area as well as provide better depth information (see chapter 7.1 on page 143). As the main purpose of rectification is to simply the matching problem, which the introduction of an additional camera does to a far greater extend as shown above, losing rectification would be a small sacrifice.

A setup in which the cameras are positioned in the corners of an equilateral allows ray tracing to be used to identify the position of the particles ([PMHD05]). In other setups, correspondence can be established between two cameras using the epipolar constraint as outlined in section 3.5.1 on page 50, and then the result verified by testing if the backprojected location of the candidate matches the image from the third camera.

8.6 Automatic self-calibration

Calibration is a time intensive process, which makes it desirable to automate it as far as possible. If the environment is fixed, Tsai's calibration method (as discussed in section 3.3.3 on page 35) suggests itself, because the calibration marks only have to be positioned once. For the stereo calibration, it is crucial that each calibration point is unique, so that the correspondence of its projection to each camera frame can be established.

8.6.1 Establishing correspondence

The easiest way to establish correspondence is to use markers which are invariant to rotation. Color coding would require switching the camera capture modes between colour and grayscale between calibration and experiment. Detecting the calibration points can be done using histogram analysis, if the colours are sufficiently unique.

8.6.2 Calibration

If the world coordinates of the calibration points are determined once, the points can be reused for calculating the absolute orientation of the cameras (see section 3.3.5 on page 43). Calibration is a time intensive process, and it is therefore desirable to automate it as far as possible. Several approaches for this can the considered, varying in the degree of automation. A fully automatic system could exploit the static nature of the scene (when no particles are present). The location of objects in the world coordinate system does not change. Self calibration is possible, if the world coordinates of a certain number of points are known, and they can be reliably detected. The number of required points depends on whether they are coplanar or not, and on the used calibration method. It would be 8, if Tsai's method is used with non coplanar points, for instance. Detection must succeed on both cameras, and it must also be obvious which points correspond without having knowledge of the stereo geometry. If the intrinsic parameters of the camera haven't changed significantly since the last experiment, the intrinsic matrices may be reused without causing unacceptable errors.

8.7 Conclusion

Let us summarize our work. We strove to develop a depth recovering particle velocimetry system, which can operate under fairly broad conditions, such as requiring only the use of a stereo camera setup and a low sampling frequency.

The system we developed composed of several modules, which we developed independently from bottom up.

- Camera calibration
- Particle segmentation
- Key point extraction
- Key point matching and tracking
- Visualization

Results and problems

Result: Depth reconstruction of sparse flow information is practical using a stereo camera setup.

Result: Tracking of individual particles can be done by generating a graph of possible paths and matching generated curves.

Problem: Selecting a suitable background mask to limit the number of candidate points and prevent the problem from becoming unmanagable from state explosion (super linear growth of possible choices).

Problem: The system is sensitive to bad calibration. It degrades performance by forcing wider search windows.

Problem: Due to the sub optimal conditions in the experimental setup, weak contrast reduced the detection performance of particles. In combination with having to track particles over several frames, this leads to significant detection losses.

Problem: Occlusion by elements in the scene geometry.

Bibliography

[AAB+84] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, *Pyramid methods in image processing*, RCA engineer **29** (1984), no. 6, 33–41.

[ABF99] G. Aubert and L. Blanc-Feraud, *Some remarks on the equivalence between 2D and 3D classical snakes and geodesic active contours*, International Journal of Computer Vision **34** (1999), no. 1, 19–28.

[AMN+94] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, *An optimal algorithm for approximate nearest neighbor searching in fixed dimensions*, ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS, 1994, pp. 573–582.

[ATV00] A. Fusiello, E. Trucco, and A. Verri, *A compact algorithm for rectification of stereo pairs*, Machine Vision and Applications **12** (2000), no. 1, 16–22.

[Bat00] G. K. Batchelor, *An introduction to fluid dynamics*, Cambridge Univ Pr, 2000.

[BCPP01] A. Babiano, J. H. E. Cartwright, O. Piro, and A. Provenzale, *The Transport of Small Particles by a Fluid*, LECTURE NOTES IN PHYSICS-NEW YORK THEN BERLIN- (2001), 114–126.

[BG80] R. L. Bishop and S. I. Goldberg, *Tensor analysis on manifolds*, Courier Dover Publications, 1980.

[BHU00] P. Brigger, J. Hoeg, and M. Unser, *B-spline snakes: A flexible tool for parametric contour detection*, IEEE Transactions on Image Processing **9** (2000), no. 9, 1484–1496.

[BKW+06] J. Bosbach, M. Kühn, C. Wagner, M. Raffel, C. Resagk, R. du Puits, and A. Thess, *Large scale particle image velocimetry of natural and mixed convection*, 13th Int. Symp. on Applications of Laser Techniques to Fluid Mechanics, Lisbon, Portugal, 2006.

[BKW09] J. Bosbach, M. Kühn, and C. Wagner, *Large scale particle image velocimetry with helium filled soap bubbles*, Experiments in Fluids **46** (2009), no. 3, 539–547.

[Boo01] C. De Boor, *A practical guide to splines*, Springer, 2001.

[Bra03] R. N. Bracewell, *Fourier analysis and imaging*, Plenum Pub Corp, 2003.

[BT98] S. Birchfield and C. Tomasi, *A pixel dissimilarity measure that is insensitive to image sampling*, IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998), no. 4, 401–406.

[BT99] ———, *Depth discontinuities by pixel-to-pixel stereo*, International Journal of Computer Vision **35** (1999), no. 3, 269–293.

[Büh07] J. Bührle, *Properties of Time-Dependent Stokes Flow and the Regularization ofVelocity Flucutations in Particle Suspensions*, Ph.D. thesis, Universitätsbibliothek Marburg, 2007.

[BvdPK] R.J.M. Bastiaans, G.A.J. van der Plas, and R.N. Kieft, *The performance of high resolution particle velocimetry: algorithm, simulations and experiments*.

[Can86] J. Canny, *A computational approach to edge detection*, IEEE Transactions on pattern analysis and machine intelligence (1986), 679–698.

[CB76] G. Comte-Bellot, *Hot-wire anemometry*, Annual Review of Fluid Mechanics **8** (1976), no. 1, 209–231.

[CB90] R. Cipolla and A. Blake, *The dynamic analysis of apparent contours*, Computer Vision, 1990. Proceedings, Third International Conference on, 1990, pp. 616–623.

[CC97] T. J. Cham and R. Cipolla, *Stereo coupled active contours*, In Proc. CVPR, 1997, pp. 1094–1099.

[CKS97] V. Caselles, R. Kimmel, and G. Sapiro, *Geodesic active contours*, Int. J. Comput. Vision **22** (1997), no. 1, 61–79.

[CM99] Dorin Comaniciu and Peter Meer, *Mean shift analysis and applications*, ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2 (Washington, DC, USA), IEEE Computer Society, 1999, p. 1197.

[CM02] D. Comaniciu and P. Meer, *Mean shift: A robust approach toward feature space analysis*, IEEE Trans. Pattern Anal. Mach. Intell. **24** (2002), no. 5, 603–619.

[CR75] R. S. Colladay and L. M. Russell, *Flow visualization of discrete hole film cooling for gas turbine applications*, Tech. report, NASA Center: Glenn Research Center, 1975.

[CV01] T. F. Chan and L. A. Vese, *Active contours without edges*, 2001.

[CZZ$^+$07] SY Chen, J. Zhang, H. Zhang, W. Wang, and YF Li, *Active illumination for robot vision*, 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 411–416.

[Day90] M. A. Day, *The no-slip condition of fluid dynamics*, Erkenntnis **33** (1990), no. 3.

[DH73] R. O. Duda and P. E. Hart, *Pattern classification and scene analysis*, John Wiley & Sons, 1973 (english).

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, Wiley New York, 2001.

[DVDD98] D. L. Donoho, M. Vetterli, R. A. Devore, and I. Daubechies, *Data compression and harmonic analysis*, IEEE Trans. Inform. Theory **44** (1998), 2435–2476.

[Fau06] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*, MIT press, 2006.

[FB87] M. A. Fischler and R. C. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, 726–740.

[FI08] A. Fusiello and L. Irsara, *Quasi-Euclidean Uncalibrated Epipolar Rectification*, International Conference on Pattern Recognition (ICPR), 2008.

[Flo97] L. Florack, *Image structure*, Kluwer Academic Publishers, 1997.

[FLP01] O. Faugeras, Q.-T. Luong, and T. Papadopoulo, *The geometry of multiple images*, MIT press Cambridge, 2001.

[FS07] P. Favaro and S. Soatto, *3-D shape estimation and image restoration: exploiting defocus and motion blur*, Springer-Verlag New York Inc, 2007.

[G$^+$02] A. Gruen et al., *Calibration and orientation of cameras in computer vision*, Measurement Science and Technology **13** (2002), 231–232.

[GBTK94] Y. G. Guezennec, R. S. Brodkey, N. Trigui, and J. C. Kent, *Algorithms for fully automated three-dimensional particle tracking velocimetry*, Experiments in Fluids **17** (1994), no. 4, 209–219.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides, *Design patterns: Elements of reusable object-oriented software*, Addison-Wesley Professional, 1994.

[GKRR01] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky, *Fast geodesic active contours*, IEEE Transactions on Image Processing **10** (2001), no. 10, 1467–1475.

[GL96] G. H. Golub and C. F. Van Loan, *Matrix computations*, Johns Hopkins Univ Pr, 1996.

[H$^+$87] B. K. P. Horn et al., *Closed-form solution of absolute orientation using unit quaternions*, Journal of the Optical Society of America A **4** (1987), no. 4, 629–642.

[Hor97] B. K. P. Horn, *Robot vision*, MIT press, 1997.

[Hor00] ______, *Tsai's camera calibration method revisited*, Department of Electrical Engineering and Computer Science Massachusetts Institute of Technology (2000).

[HR] M. Heinrichs and V. Rodehorst, *Trinocular rectification for various camera setups*, Symp. of ISPRS Commission III-Photogrammetric Computer Vision PCV, vol. 6, Citeseer, pp. 43–48.

[HS] J. Heikkilä and O. Silvén, *Calibration procedure for short focal length off-the-shelf CCD cameras*, Proceedings of ICPR, vol. 96, p. 166.

[HS88] C. Harris and M. Stephens, *A combined corner and edge detector*, Alvey vision conference, vol. 15, Manchester, UK, 1988, p. 50.

[HS97] J. Heikkilä and O. Silvén, *A four-step camera calibration procedure with implicit image correction*, CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97) (Washington, DC, USA), IEEE Computer Society, 1997, p. 1106.

[HZ04] R. I. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, second ed., Cambridge University Press, ISBN: 0521540518, 2004.

[Jäg95] Martin Jägersand, *A scale decomposed information measure in images*, Proc. ARPA Image Understanding Workshop In Proc of 5th International Conference on Computer Vision, 1995.

[JN01] F. V. Jensen and T. D. Nielsen, *Bayesian networks and decision graphs*, ASA, 2001.

[Kaj86] J. T. Kajiya, *The rendering equation*, ACM SIGGRAPH Computer Graphics **20** (1986), no. 4, 143–150.

[KF01] A. Kuijper and L. M. J. Florack, *The application of catastrophe theory to image analysis*, 2001.

[KM00] U. Köthe and H. Meine, *VIGRA-vision with generic algorithms: Documentation*, WWW: http://kogs-www.informatik.uni-hamburg.de/koethe/vigra/[visited July, 1st 2006] (2000).

[Köt] U. Köthe, *Edge and junction detection with an improved structure tensor*.

[Köt07] U. Köthe, *Reliable low-level image analysis*, Professorial dissertation, 2007.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes: Active contour models*, INTERNATIONAL JOURNAL OF COMPUTER VISION **1** (1988), no. 4, 321–331.

[Lam32] H. Lamb, *Hydrodynamics*, Dover Publications, 1932.

[Lan96] C. Lanczos, *Linear differential operators*, Society for Industrial Mathematics, 1996.

[Lee00] I.-K. Lee, *Curve reconstruction from unorganized points*, Computer Aided Geometric Design **17** (2000), no. 2, 161–177.

[LF95] Q.-T. Luong and O. D. Faugeras, *The fundamental matrix: theory, algorithms, and stability analysis*, International Journal of Computer Vision **17** (1995), 43–75.

[LFSK06] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang, *Noise estimation from a single image*, 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2006.

[Lin94] T. Lindeberg, *Scale-space theory in computer vision*, Springer, 1994.

[LL59] L. D. Landau and E. M. Lifshitz, *Fluid mechanics. Translated from the Russian by JB Sykes and WH Reid*, Course of Theoretical Physics **6** (1959).

[Lu76] Y.C. Lu, *Singularity theory and an introduction to catastrophe theory*, New York (1976).

[LZ99] C. Loop and Z. Zhang, *Computing rectifying homographies for stereo vision*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 1999, pp. 125–131.

[Maa93] H. G. Maas, *Complexity analysis for the establishment of image correspondences of dense spatial target fields*, International Archives of Photogrammetry and Remote Sensing **29** (1993), 102–102.

[Mar75] KV Mardia, *Statistics of directional data*, Journal of the Royal Statistical Society. Series B (Methodological) (1975), 349–393.

[MB] R. Micheals and T. Boult, *A New Closed Form Approach to the Absolute Orientation Problem.*

[MBZ] A. Maeder, H. Bistry, and J. Zhang, *Intelligent Vision Systems For Robotic Applications.*

[Mei09] H. Meine, *The geomap representation: On topologically correct sub-pixel image analysis*, Ph.D. thesis, Department Informatik, Universität Hamburg, jan 2009.

[Mer87] W. Merzkirch, *Flow visualization*, 2nd ed., Academic Press, 1987.

[MG04] M. McDowell and E. Gray, *Stereo Imaging Velocimetry Technique Using Standard Off-the-Shelf CCD Cameras*, Tech. report, NASA Center for AeroSpace Information, 7121 Standard Dr, Hanover, Maryland, 21076-1320, USA, 2004.

[MK05] H. Meine and U. Köthe, *Image segmentation with the exact watershed transform*, Proceedings of the Fifth IASTED International Conference on Visualization, Imaging, and Image Processing (J.J. Villanueva, ed.), IASTED, ACTA Press, September 2005, pp. 400–405.

[MLKC91] N. J. Marshall, M. F. Land, C. A. King, and T. W. Cronin, *The compound eyes of mantis shrimps (Crustacea, Hoplocarida, Stomatopoda). I. Compound eye structure: the detection of polarized light*, Philosophical Transactions: Biological Sciences **334** (1991), no. 1269, 33–56.

[MMR01] D. Müller, B. Müller, and U. Renz, *Three-dimensional particle-streak tracking (PST) velocity measurements of a heat exchanger inlet flow A new method to measure all three air-flow velocity components in a plane is applied to a steady-state three-dimensional flow*, Experiments in Fluids **30** (2001), no. 6, 645–656.

[Mor77] J. J. Mor, *The Levenberg-Marquardt algorithm: implementation and theory*, Lecture notes in mathematics **630** (1977), 105–116.

[Mou98] D. M. Mount, *Ann programming manual*, Tech. report, 1998.

[MSW63] J. W. Milnor, M. Spivak, and R. Wells, *Morse theory*, Princeton Univ Pr, 1963.

[Nak06] J. Nakamura, *Image sensors and signal processing for digital still cameras*, CRC, 2006.

[NDT04] H. Nobach, N. Damaschke, and C. Tropea, *High-precision sub-pixel interpolation in PIV/PTV image processing*, Proc. 12th Int. Symp. on Appl. of Laser Techn. to Fluid Mechanics, Lisbon, Portugal, 2004.

[NIK91] S. K. Nayar, K. Ikeuchi, and T. Kanade, *Surface reflection: Physical and geometrical perspectives*, IEEE Transactions on Pattern Analysis and Machine Intelligence **13** (1991), 611–634.

[NKKM08] Y. Niwa, Y. Kamiya, T. Kawaguchi, and M. Maeda, *Bubble sizing by interferometric laser imaging*, 10th International Symposium on Application of Laser Technique to on promoting Advanced Technologies in Manufacturing WESIC ‘08 Bucharest, 2008, pp. 25–26.

[PBP02] H. Prautzsch, W. Boehm, and M. Paluszny, *Bezier and B-spline techniques*, Springer Verlag, 2002.

[PDG05] A. C. Parr, R. U. Datla, and J. L. Gardner, *Optical radiometry*, Academic Pr, 2005.

[PLH02] H. Pottmann, S. Leopoldseder, and M. Hofer, *Approximation with Active B-Spline Curves and Surfaces*, Proceedings of the 10th Pacific Conference on Computer Graphics and Applications, IEEE Computer Society Washington, DC, USA, 2002.

[PMHD05] N. F. Ponchaut, C. A. Mouton, H. G. Hornung, and D. Dabiri, *3D particle tracking velocimetry method: Advances and error analysis*, Measurement Science and Technology (2005).

[Poz09] C. Pozrikidis, *Fluid dynamics: theory, computation, and numerical simulation*, Springer Verlag, 2009.

[PP91] J. Porrill and S. Pollard, *Curve matching and stereo calibration.*, Image and Vision Computing **9** (1991), no. 1, 45–50.

[Pra00] A. K. Prasad, *Stereoscopic particle image velocimetry*, Experiments in fluids **29** (2000), no. 2, 103–116.

[PVW] F.H. Post and T. Van Walsum, *Fluid flow visualization*.

[RV06] G. Rote and G. Vegter, *Computational topology: An introduction*, Effective Computational Geometry for Curves and Surfaces (2006).

[She] J. R. Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain*, Computer Science Tech. Report, 94–125.

[SK05] P. Stelldinger and U. Köthe, *Towards a general sampling theory for shape preservation*, Image and Vision Computing **23** (2005), no. 2, 237–248.

[SL94] A. Stepanov and Meng Lee, *The standard template library*, Tech. report, WG21/N0482, ISO Programming Language C++ Project, 1994.

[ST70] M. D. Springer and W. E. Thompson, *The distribution of products of beta, gamma and Gaussian random variables*, SIAM Journal on Applied Mathematics (1970), 721–737.

[Ste08] P. Stelldinger, *Image Digitization and its Influence on Shape Properties in Finite Dimensions*, Ph.D. thesis, 2008.

[TBW03] T. Thormaehlen, H. Broszio, and I. Wassermann, *Robust line-based calibration of lens distortion from a single view*, Proc Computer Vision/Computer Graphics Collaboration for Model-based Imaging, Rendering, Image Analysis and Graphical Special Effects (Mirage) (2003), 105–112.

[Tik63] A. N. Tikhonov, *Regularization of incorrectly posed problems*, Soviet Math. Dokl, vol. 4, 1963, pp. 1624–1627.

[TRK01] Y. Tsin, V. Ramesh, and T. Kanade, *Statistical calibration of CCD imaging process*, Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001. Proceedings, vol. 1, 2001.

[TRW] H. Theisel, C. Rössl, and T. Weinkauf, *Topological representations of vector fields*.

[Tsa87] R. Tsai, *A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses*, IEEE Journal of robotics and Automation **3** (1987), no. 4, 323–344.

[TWHS] H. Theisel, T. Weinkauf, H. C. Hege, and H. P. Seidel, *Saddle connectors – an approach to visualizing the topological skeleton of complex 3d vector fields*, In Proc. IEEE Visualization 2003 (2003, pp. 225–232.

[UAE93a] M. Unser, A. Aldroubi, and M. Eden, *B-spline signal processing. i. theory*, IEEE Trans. Signal Processing **41** (1993), 821–833.

[UAE93b] ———, *B-spline signal processing. II. Efficiency design and applications*, IEEE transactions on signal processing **41** (1993), no. 2, 834–848.

[vN55] J. von Neumann, *Method in the physical sciences*, Collected Works **6** (1955), 491–498.

[Wec67] G. P. Weckler, *Operation of pn junction photodetectors in a photon flux integrating mode*, IEEE Journal of Solid-State Circuits **2** (1967), no. 3, 65–73.

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

[WG92] C. E. Willert and M. Gharib, *Three-dimensional particle imaging with a single camera*, Experiments in Fluids **12** (1992), no. 6, 353–358.

[WPL04] W.P. Wang, H. Pottmann, and Yang Liu, *Fitting b-spline curves to point clouds by squared distance minimization*, ACM Transactions on Graphics **25** (2004).

[XP98] C.Y. Xu and J. L. Prince, *Snakes, shapes, and gradient vector flow*, IEEE TRANSACTIONS ON IMAGE PROCESSING **7** (1998), no. 3, 359–369.

[YMX06] R. Yang, M. Mirmehdi, and X. Xie, *A charged active contour based on electrostatics*, LECTURE NOTES IN COMPUTER SCIENCE **4179** (2006), 173.

[Z$^+$00] Z. Zhang et al., *A flexible new technique for camera calibration*, IEEE Transactions on pattern analysis and machine intelligence **22** (2000), no. 11, 1330–1334.

[Zey91] R. K. Zeytounian, *Mécanique des fluides fondamentale*, Springer, 1991.

[Zha99] Z. Zhang, *Flexible camera calibration by viewing a plane from unknown orientations*, International Conference on Computer Vision, vol. 1, 1999, pp. 666–673.

[ZR96] H. Zhuang and Z.S. Roth, *Camera-aided robot calibration*, CRC, 1996.

Software used

[BA07] A. W. Bowman and A. Azzalini, *R package `sm`: nonparametric smoothing methods (version 2.2)*, University of Glasgow, UK and Università di Padova, Italia, 2007.

[Bou08] J. Y. Bouguet, *Camera calibration toolbox for matlab*, 2008.

[BS08] J. Blanchette and M. Summerfield, *C++ gui programming with qt 4*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2008.

[Cor] Intel Corporation, *Opencv (version 1.1)*, http://opencv.willowgarage.com/, Computer software.

[dt] Boost C++ development team, *Boost c++ libraries (version 1.41.0)*, http://www.boost.org/, Computer software.

[Gro] The Sage Group, *Sage: Open Source Mathematical Software*, http://www.sagemath.org/.

[Kar05] B. Karlsson, *Beyond the c++ standard library*, Addison-Wesley Professional, 2005.

[oVPL] Persistence of Vision Pty. Ltd. (2004), *Persistence of vision raytracer (version 3.6)*, http://www.povray.org/, Computer software.

[SML97] W. Schroeder, K. Martin, and B. Lorensen, *The visualization toolkit: An object-oriented approach to 3d graphics*, Prentice Hall, 1997.

Appendix

Test image series

For the purposes of testing and evaluating the algorithm as a whole and its individual steps, we prepared some image series which serve as a touchstone. These images are referenced many times in the text, so we thought it best to present them.

Legend

Due to the frequent appearance of images which are positive- as well as negative-valued, the greyvalues are colorcoded so as to show the former in green, the latter in red.

Synthetic series

All the synthetic series routinely used as examples consist of 640x480 images, rectified cameras, no undistortion. Series 4 is used most frequently as an example, because non-uniform motion of particles poses a special challenge in long-exposure velocimetry. Series 3 is also a good test case because of the many overlapping traces.

Series 1	One particle in an unaccelerated motion
Series 2	An array of such particles
Series 3	An array of particles moving in random directions
Series 4	One particle in a circular motion

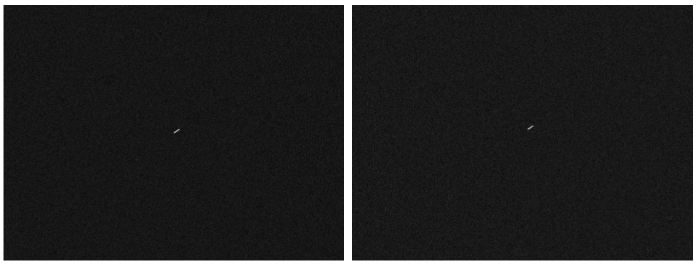


Figure 8.2: Successive frames of a synthetic sequence (*synth*1), single well-lit particle, linear motion, simulated focal blur, added Gaussian noise $\sigma = 2$

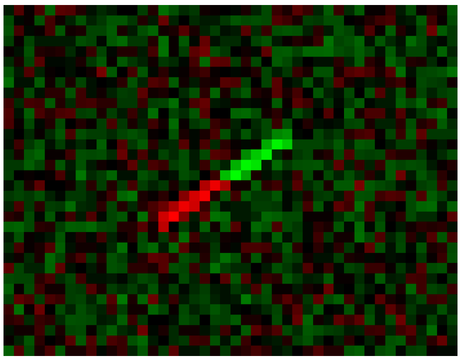


Figure 8.3: Closeup of *synth*1 difference image around location (206,146). Negative values are shown in red; values are scaled logarithmically as to make the noise visible. This is a typical example of an unproblematic case.

Actual Camera Image Series

2048x1100 images, 7Hz frequency, speeds of up to about $50 cm \cdot s^{-1}$ –

Series V2 baseline 60cm, almost parallel cameras. Nearly perpendicular to illumination corridor.

Series V4 baseline 38cm, slightly more angularly oriented cameras; angular view on illumination corridor; lit inlet near center of both views.

Documentation of the prototype

The following is indented as technical documentation to the prototype. Each chapter discusses one of its components, with the exception of the first. We will describe how to operate each component, and will document the data formats we used for input and results.

Calibration using Matlab Toolkit

Time constraints did not allow us to implement a full featured calibration widget for determining intrinsic and extrinisc parameters. Instead, external tools have to be used for this step. We decided to use the Camera Calibration Toolkit, available online at `http://www.vision.caltech.edu/bouguetj/calib_doc/`. This method uses a modification of Zhang's algorithm for calibration . Both single camera and stereo camera calibration are necessary, which are covered in the sections to example 1 and 5 on the website. A printable calibration pattern is available in the files to example 1, but any checkerboard pattern with known geometry will do. Calibration is done by loading the Calibration images for each camera, and selecting the corners of the calibration pattern. It is important that the order in which the corners are selected is identical for both cameras.

Once the calibration data has been obtained, it needs to be made available to the PTV system. Section 8.7 on page 209 shows how to manually insert the data gained from the Toolkit into a fitting format.

Rectification

Rectification (figure 8.4 on the following page) ensures that both camera image planes are (nearly) coplanar, and that both optical centres share the same vertical coordinate. This simplifies the stereo matching problem. The rectification process requires calibration data as input, along with paths to the raw images, and to the output folders, where the rectified images are to be stored. It is no longer required to rectify entire images using this widget, unless one wishes to visually evaluate the quality of the stereo calibration (which determines the quality of the rectification), using the widget designed for that purpose.

1. Select the XML file with calibration data. If no stereo calibration data is available, refer to the section above.
2. Select the paths to the directories with the new images from the left and right camera.

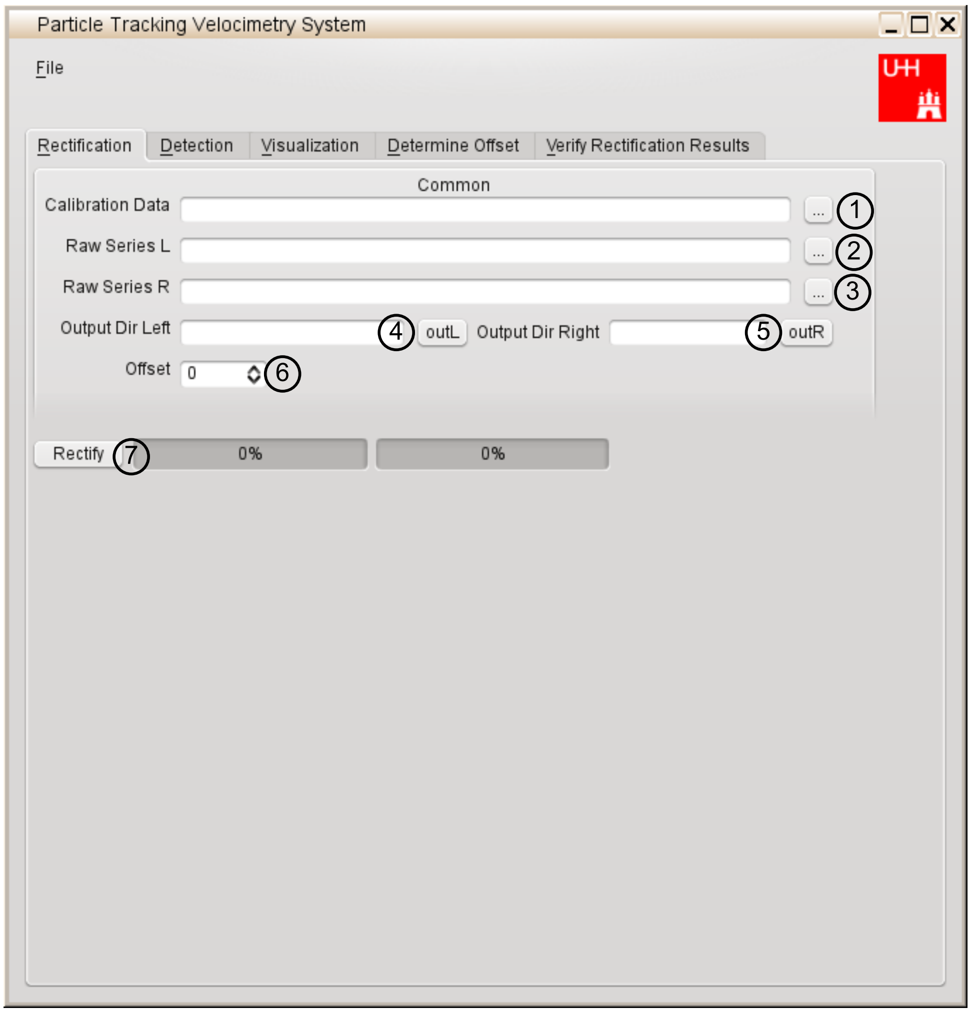


Figure 8.4: Screenshot of the rectification widget.

3. Select the paths to the directories with the new images from the left and right camera.
4. Chose the output directories for the rectified images.
5. Chose the output directories for the rectified images.
6. Select the offset, after which both cameras acquire images. This can be determined manually, or through a widget , which is described in the next section.
7. Once all input parameters have been set, the rectification process can be started by clicking on the “Rectify” button.

Determining the frame offset

This widget (figure 8.5 on the next page) assists in finding the frame after which both cameras acquire synchronous images. A necessary step, as both cameras receive the signal to start capturing separately. During each capture phase, a visual signal should be given. Examples for suitable signals are flashes of light, or a digital clock. Afterwards, the pair of frames in which the signal is visible needs to be found.

The widget receives its data paths from the rectification widget, and the *Determine Offset* button is disabled until the paths have been set. Once the button has been activated and pressed, the first image in each folder is loaded.

1. Shows the current offset.
2. Show the current frame index for the left folder. Changing the index updates the displayed image.
3. Show the current frame index for the right folder. Changing the index updates the displayed image.
4. Display the maximum frame index for the left folder.
5. Display the maximum frame index for the right folder.
6. Lock the current left index. Once the index for both folders are locked, the offset shown by (1) is adapted by the rectification widget.
7. Lock the current right index. Once the index for both folders are locked, the offset shown by (1) is adapted by the rectification widget.
8. Display the current left image. The view can be scrolled by pressing and holding the left mouse button. Using the mouse wheel, zooming in and out of the image is possible.
9. Display the current right image. The view can be scrolled by pressing and holding the left mouse button. Using the mouse wheel, zooming in and out of the image is possible.

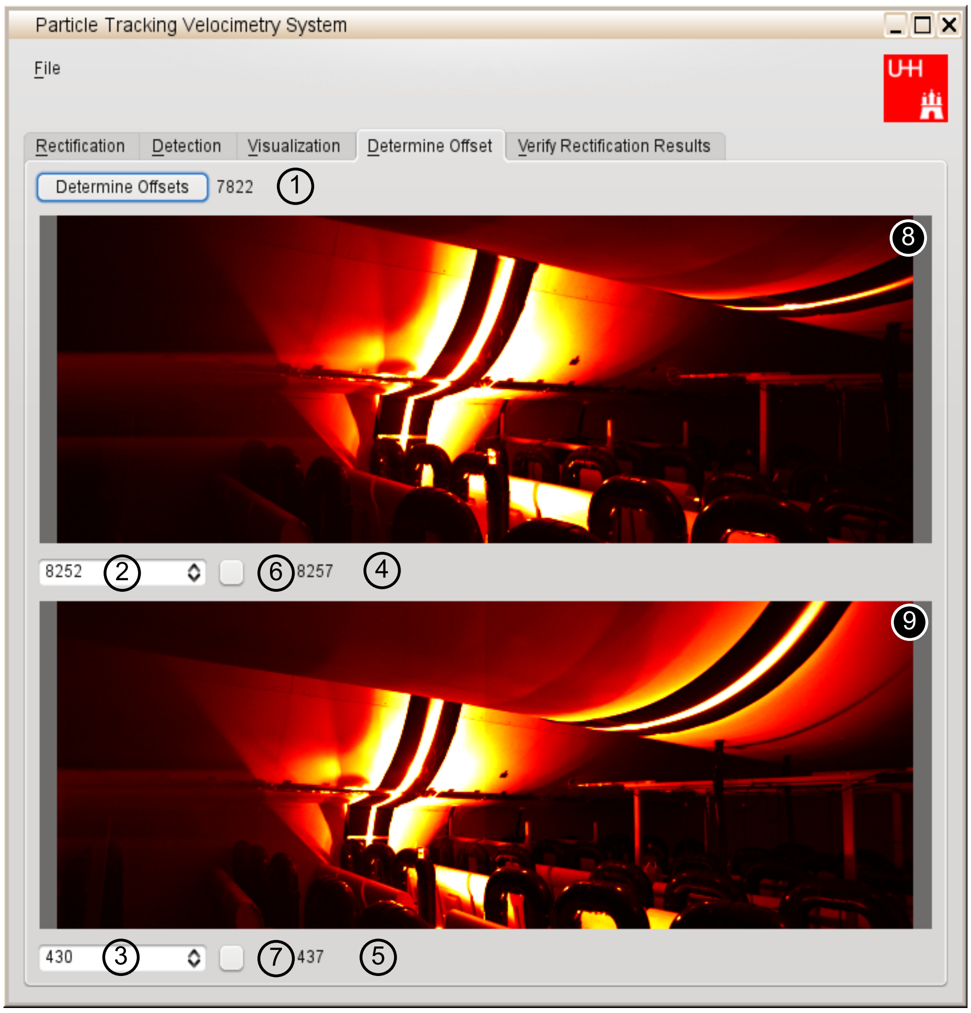


Figure 8.5: Screenshot of the widget, which assists the user in determining the frame offset between both cameras.

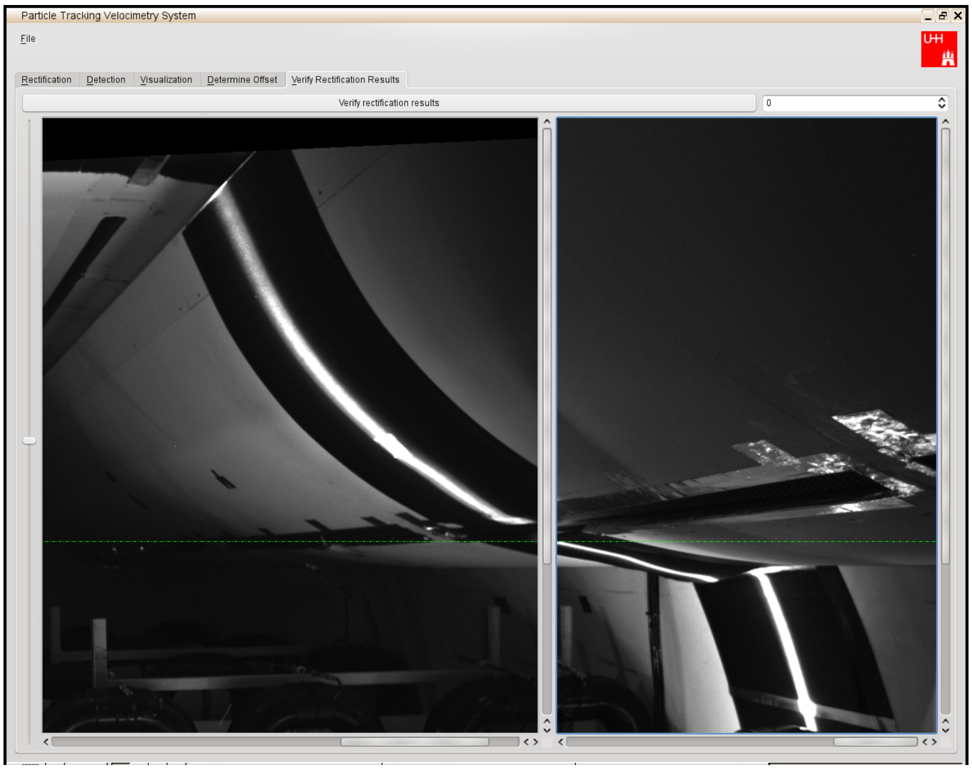

Particle Tracking Velocimetry System
File
Rectification
Detection
Visualization
Determine Offset
Verify Rectification Results
Verify rectification results
0

Figure 8.6: Rectification widget, showing a pair of rectified frames. The green line marks common vertical coordinates on both images.

Verification of the rectification results

It is advisable to verify the rectification results before starting the detection process, as inaccurate rectification will lead the stereo matching algorithm to fail. The widget (figure 8.6) requires paths to the folders with the rectified images to be set. This can be done in the Rectification widget.

The slider moves the dotted line in y direction on both images. If the rectification was successful, points in the scene should have common vertical coordinates on the images. An aberration of a few pixels is tolerable, but significant disparities will lead to errors. The user can browse through the images using the spinbox.

Note that this verification is only possible, if the user opted to rectify the input images, which is no longer required in recent version of the prototype.

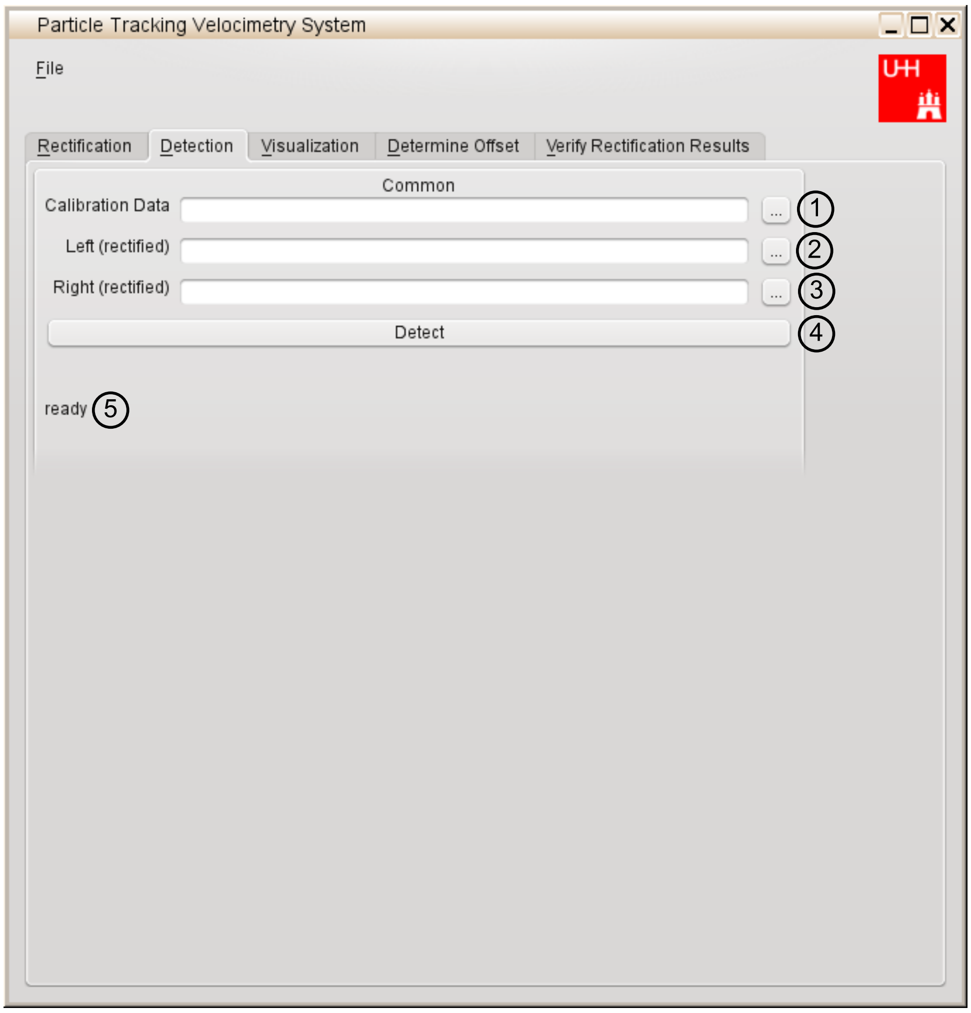


Figure 8.7: Screenshot of the detection widget

The detection widget

While the detection widget (figure 8.7) is the key component of the system, it requires little user interaction. Datapaths set in the rectification widget are automatically reused. Otherwise they can be set in the appropriate fields. It is possible to run a rectification and detection process in parallel, but they cannot operate on the same dataset - the detection widget requires a complete set of rectified images. Once the *Detect* button has been pressed, the system extracts the traces. A progress bar will be displayed, and is updated, as the system proceeds.

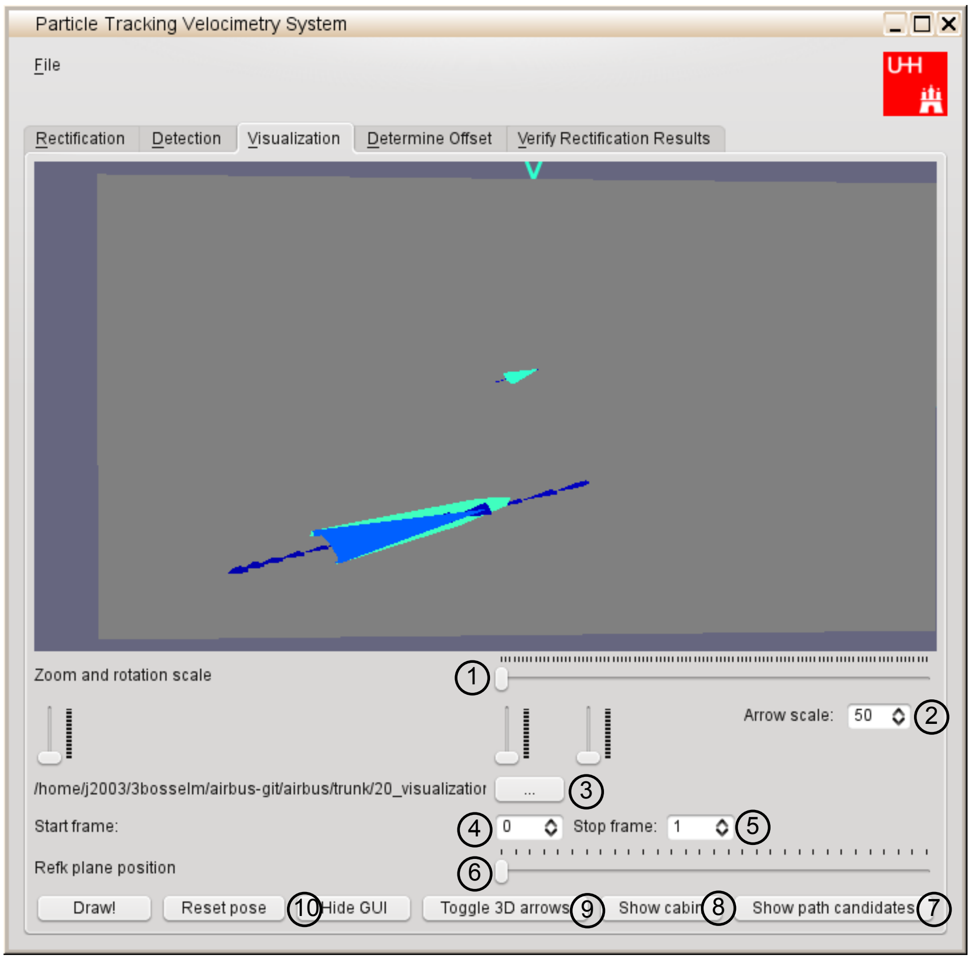


Figure 8.8: Screenshot of the Visualization Widget with some particle traces and 3D arrows pointing indicating the direction of the movement, and their size being relative to their speed. A couple of spurious paths are visible.

Visualization of particle traces 1

The visualization widget (figure 8.8 on the preceding page) offers one way of visualizing results of the PTV system. While it is more difficult to operate than the other visualization method, it scales better with increasing numbers of particles. The final vectorfield over time can be displayed, along with all path candidates for each frame. Moving the mouse while pressing and holding the left mouse button allows scrolling. Pressing and holding the right mouse button allows rotation around the x- and y-axis. Page up and Page down adjust the zoom level. Traces are colour coded according to their velocity.

1. Adjustments how much zoom and rotation are changed by input.
2. Sets the scale for 3D arrows. Depending on the zoom level, high values might be necessary to make the direction of the traces clearly visible.
3. The input directory with the results from the detection widget. This is the programme directory by default, but can be set to any folder.
4. Sets the interval of frames which are to be displayed. Select 0 and 1, if the traces should be displayed over time.
5. Sets the interval of frames which are to be displayed. Select 0 and 1, if the traces should be displayed over time.
6. Changes the z position of the reference plane.
7. Toggles display between the actual traces, and path candidates for an arbitrary interval of frames.
8. Toggles the display of a cabin model (the model is placed at the origin of the stereo camera coordinate system, and not at its real position in the world coordinate system)
9. Toggles display of 3D arrows. 3D arrows are displayed as cones, with the narrow end pointing in the direction of the particle's movement.
10. Toggles the display of some GUI components, in order to maximize the display for the visualization component.

Visualization of particle traces 2

This visualization widget (Figure 8.9 on the next page) is the second option for visualizing the generated particle trajectories. It renders the trajectories as splines.

Moving the mouse while pressing and holding both mouse buttons allows scrolling. Pressing and holding the left mouse button causes rotation around the x- and y-axis, and pressing and holding the right mouse button leads to zoom.

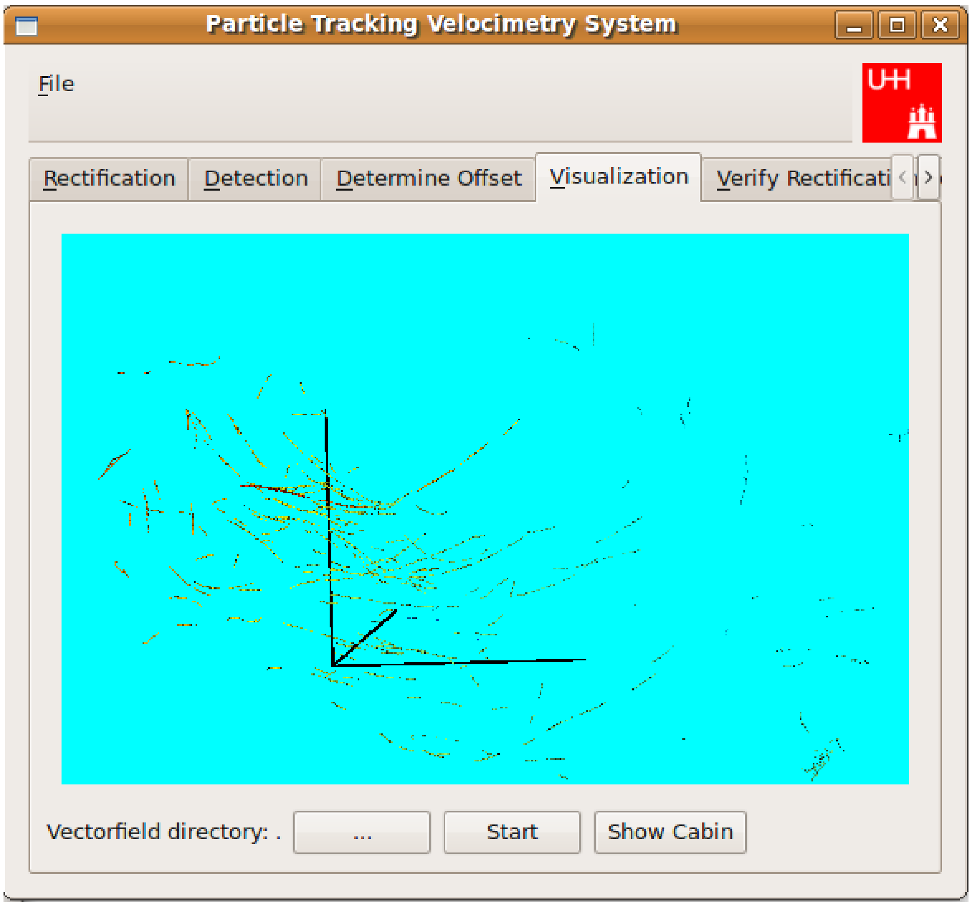


Figure 8.9: Screenshot of the visualization widget with a loaded set of particle trajectories

By moving the mouse cursor over a particle trajectory, and pressing *p*, information to that trajectory can be displayed. Pressing *f* will lead the camera to fly to the picked object. The camera can be reset by pressing *r*.

Determination of world to camera coordinate transform

This widget estimates the rotation and translation between the scene and the camera coordinate system. If this step is not conducted, the actual position of the traces in the scene remains unknown, which can be easily seen by the wrongly aligned cabin model in the visualization widget. In order to estimate the transformation, a file with at least three points in the scene coordinate system must be supplied (Listing 8.1). The *item* tags of each *vt* element store the coordinates in x,y,z order.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="5">
<worldPoints class_id="0" tracking_level="0" version="0">
  <count>3</count>
  <item_version>0</item_version>
  <item class_id="1" tracking_level="0" version="0">
    <first>first</first>
    <second class_id="2" tracking_level="0" version="0">
      <vt>
        <count>3</count>
        <item_version>0</item_version>
        <item>1.1</item>
        <item>2.4</item>
        <item>3.5</item>
      </vt>
    </second>
  </item>
  <item>
    <first>second</first>
    <second>
      <vt>
        <count>3</count>
        <item_version>0</item_version>
        <item>30.7</item>
        <item>70.5</item>
        <item>90.1</item>
      </vt>
    </second>
  </item>
  <item>
    <first>third</first>
    <second>
      <vt>
        <count>3</count>
        <item_version>0</item_version>
        <item>1013.45</item>
        <item>246.23</item>
        <item>378.1</item>
      </vt>
    </second>
  </item>
</worldPoints>
```

Listing 8.1: Serialized world coordinates

Once the scene points have been loaded, they have to be selected in the widget. It is important to select them in the order in which they are listed in the file. Markers can be placed on the left and right canvas with a mouseclick (and removed with a rightclick on the marker). Once three pairs of points have been selected, the widget automatically performs the necessary triangulations and estimates the transformation, which is printed to the console and serialized to *C_to_W_Transform.xml* (listing 8.2).

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="5">
<Rect_wid_settings class_id="0" tracking_level="0" version="0">
  <size1>4</size1>
  <size2>4</size2>
  <data class_id="1" tracking_level="0" version="0">
    <size>16</size>
    <item>1</item>
    <item>0</item>
    <item>0</item>
    <item>0</item>
    <item>0</item>
    <item>1</item>
    <item>0</item>
    <item>-1</item>
    <item>0</item>
    <item>0</item>
    <item>1</item>
    <item>0</item>
    <item>0</item>
    <item>0</item>
    <item>0</item>
    <item>1</item>
  </data>
</Rect_wid_settings>
</boost_serialization>
```

Listing 8.2: World to camera transformation data

Visualization output formats

The movement vectors over time are stored in the file *splinefield0.xml*, whose format can be seen in listing 8.3.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="5">
<vf class_id="0" tracking_level="0" version="0">
<count>80</count>
<item_version>0</item_version>
<item class_id="1" tracking_level="0" version="0">
<line.line class_id="2" tracking_level="0" version="0">
<count>2</count>
<item_version>0</item_version>
<item class_id="3" tracking_level="0" version="0">
<point.x>-0.16816781398397795</point.x>
<point.y>0.057320373725824637</point.y>
<point.z>1.7477117407420109</point.z>
</item>
<item>
<point.x>-0.16542142436836615</point.x>
<point.y>0.058808616872696937</point.y>
<point.z>1.7458286278243829</point.z>
</item>
</line.line>
<line.labels class_id="4" tracking_level="0" version="0">
<count>0</count>
<item_version>0</item_version>
</line.labels>
</item>
```

Listing 8.3: Vector output

Count specifies the total number of vectors. Each vector is stored as an *item* tag. An *item* tag consists of the actual vector, stored in the *line* tag, and additional information, which is stored in the *labels* tag. The same format is used for the storage of path candidates. One *xxx*.xml* file is generated for each frame.

Calibration result file format

Calibration results are stored as text in a XML file format. The file is generated by serializing matrices from the opencv computer vision library. However, it is editable with any text editor. An example output from a stereo calibration, using the Matlab Toolkit, is shown in listing 8.4 (irrelevant information has been omitted).

```
Intrinsic parameters of left camera:
Focal Length:        fc_left = [ 1413.57400   1420.35403 ]
Principal point:     cc_left = [ 999.64805   593.04928 ]
Skew:                alpha_c_left = [ 0.00000 ] ? [ 0.00000  ]
Distortion:          kc_left = [ -0.12301   0.15433   0.00046   -0.00482
     0.00000 ]

Intrinsic parameters of right camera:
Focal Length:        fc_right = [ 1134.12674   1142.16153 ]
Principal point:     cc_right = [ 995.61545   609.94739 ]
Skew:                alpha_c_right = [ 0.00000 ] ? [ 0.00000  ]
Distortion:          kc_right = [ -0.15766   0.09402   -0.00073
    -0.00387  0.00000 ]

Extrinsic parameters (position of right camera wrt left camera):
Rotation vector:  om = [ -0.01448   0.02476  0.04163 ]
Translation vector: T = [ -0.59456   -0.03337  -0.03091 ]
```

Listing 8.4: Calibration data generated by the Matlab Toolbox

Listing 8.5 shows an example of a calibration file, as it is used in the PTV system. The results of the Toolkit calibration have been inserted into the appropriate places.

```
<?xml version="1.0"?>
<opencv_storage>
<intrinsic_left type_id="opencv-matrix">
  <rows>3</rows><cols>3</cols><dt>f</dt>
  <data>1413.6 0. 999.648 0. 1420.2 593.0493 0. 0. 1.</data>
</intrinsic_left>
<intrinsic_right type_id="opencv-matrix">
  <rows>3</rows><cols>3</cols><dt>f</dt>
  <data>1134.1 0. 995.6154 0. 1142.2 609.9474 0. 0. 1.</data>
</intrinsic_right>
<translation type_id="opencv-matrix">
  <rows>1</rows><cols>3</cols><dt>f</dt>
  <data>-0.5946   -0.0334 -0.0309</data>
</translation>
<rotation_axis type_id="opencv-matrix">
  <rows>1</rows><cols>3</cols><dt>f</dt>
  <data> -0.0145 0.0248 0.0416</data>
</rotation_axis>
<dcoeff_left type_id="opencv-matrix">
  <rows>1</rows><cols>4</cols><dt>f</dt>
  <data>-0.123 0.1543 0. 0.</data>
</dcoeff_left>
<dcoeff_right type_id="opencv-matrix">
  <rows>1</rows><cols>4</cols><dt>f</dt>
  <data> -0.1577 0.094 0. 0.</data>
</dcoeff_right>
</opencv_storage>
```

Listing 8.5: Single and stereo camera calibration data

The intrinsic matrix has the form $A = \begin{bmatrix} a_u & \gamma & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$,where $a_u, a_v$is the focal length in x and y direction (fc_left/right in the Matlab data) , γ is the skew factor (alpha_c_left/right), and $u_0, v_0$are the x and y coordinates of the principle point (cc_left/right in the Matlab data).

XML File	Matlab data
kc_left	dcoeff_left
kc_right	dcoeff_right
translation	T
rotation_axis	om

Table 8.1: Corresponding fields in the Matlab Toolkit output and the XML input file for the prototype

The other correspondences can be extracted from table 8.1.

Acknowledgments

We would like to thank everybody who made this thesis possible. First of all, our supervisors, Prof. Dr. Jianwei Zhang and Dr. Peer Stelldinger. Professor Zhang especially for establishing and overseeing the cooperation, and Dr. Stelldinger for advice regarding the planning and execution. Both granted us great freedom in our research and were always available and open to questions.

Next, we cannot overstate the support lent by Denis Klimentjew who provided us with a wealth of practical suggestions as well as facilitating much of the communication and coordination between AIRBUS and the TAMS and KOGS groups of the University of Hamburg, and Hannes Bistry for proof-reading the text and offering some very constructive criticism.

Speaking of AIRBUS, we are grateful for having been able to work with Mr. Mohand Terzi and his crew at the Cabin Innovation Center, and for Mr. Helmut Tripmaker for granting us clearance to use pictures obtained during our experiments for this work.

TAMS and KOGS graciously provided the necessary computer infrastructure, and we are indebted to the KOGS group for being able to use their VIGRA library (including the development branch).

Aufteilung der Gruppenarbeit

Da diese Arbeit von Till Bosselmann und Erik Flick gemeinsam erstellt wurde, werden im Folgenden einzelne Kapitel dem jeweiligen Autor zugeordnet.

Von Till Bosselmann, Matr.-Nr.: 5630677, erstellte Kapitel: 1, 2.5, 3 (au”ser 3.2, 3.4, 3.6), 5.0, 5.1, 6 (bis auf 6.3), 7.0, 7.5, 7.6, 7.7, 8.4 bis 8.7, Technische dokumentation

Von Erik Flick, Matr.-Nr.: 5519040, erstellte Kapitel: 1.0, 1.3, 2 (bis auf 2.5), 4 (bis auf 4.2), 5, 6.3, 7.1 bis 7.4, 7.6, 8.0 bis 8.3

Die hier nicht aufgeführten Textpassagen sowie solche, die bei Beiden (auch implizit) aufgelistet sind, genauso wie der gesamte Quellcode wurde in Zusammenarbeit erstellt beziehungsweise programmiert.

Erklärung

Ich, Till Bosselmann, Matr.-Nr: 5630677, versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

(Ort,Datum) (Unterschrift)

Ich, Erik Flick, Matr.-Nr: 5519040, versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntrlich gemacht.

(Ort,Datum) (Unterschrift)

Index