Timo Tossavainen

Virtual Reality and Posturography Applied to Postural Control Research

ACADEMIC DISSERTATION

To be presented, with the permission of the Faculty of Information Sciences of the University of Tampere, for public discussion in the B1097 Auditorium of the University on February 3rd, 2006, at 12 noon.

DEPARTMENT OF COMPUTER SCIENCES
UNIVERSITY OF TAMPERE

A-2005-4

TAMPERE 2006

Supervisor: Prof. Martti Juhola
University of Tampere

Opponent: Prof. Jussi Parkkinen
University of Joensuu

Reviewers: Prof. Jari Hyttinen
Tampere University of Technology

Prof. Pekka Loula
Tampere University of Technology, Pori

Department of Computer Sciences
FIN-33014 UNIVERSITY OF TAMPERE
Finland

Electronic dissertation
Acta Electronica Universitatis 507
ISBN 951-44-6563-6
ISSN 1456-954X
http://acta.uta.fi

Abstract

This thesis describes the design and development and some applications of an integrated stimulation and measurement system for postural control research based on virtual reality (VR) methods and force platform posturography. The system exposes test subjects to visual stimulation with immersive computer-generated environments and measures their responses to the stimulation using a force platform. Analysis is carried out on the measured stabilograms.

Our first experiments show that virtual environments affect balance and that they can be designed to cause desired effects, such as leaning in different directions, in test subjects. We investigated the efficacy of a head-mounted display and the CAVE display for visual stimulation by exposing test subjects to the same virtual environments using both displays. There were significant differences in the responses between the two displays. Next, we constructed an integrated VR posturography system in a laboratory at the Hearing Center of Tampere University Central Hospital and tested it on control subjects and patients with diagnosed balance disorders. Responses of control subjects and patients with Ménière's disease differed significantly and provided good discrimination between the two groups. Because the stabilograms are difficult to intepret, we applied pattern recognition methods to summarize the differences in them between the two groups.

The applications described in this thesis show that VR is a versatile and effective visual stimulation method for use in postural control research. Many of the experimental setups used the same hardware to implement a sequence of balance tests. The combination of VR visual stimulation and posturography provided an easy way to quickly and comprehensively characterize a test subject's postural stability.

Keywords: virtual reality, posturography, postural control, postural stability.

Acknowledgements

I wish to thank my supervisor Professor Martti Juhola, Ph.D., for his guidance and my other co-authors Professor Ilmari Pyykkö, M.D., Ph.D., Docent Esko Toppila, Ph.D., Pia Forsman, Ph.Lic., Heikki Aalto, Ph.D., Pekka Honkavaara, M.D., Ph.D., and Professor Jukka Starck, Ph.D., for lending me their expertise.

This work was carried out at the Department of Computer Sciences of The University of Tampere, headed, in turn, by Professors Seppo Visala, Ph.D., Pertti Järvinen, Ph.D., and Jyrki Nummenmaa, Ph.D. My coworkers have made the department a pleasant working environment. In particular, I wish to thank Isto Aho, Ph.D., Heikki Hyyrö, Ph.D., Kati Iltanen, Ph.D., Jorma Laurikkala, Ph.D., Jouni Mykkänen, Ph.D., Tapio Niemi, Ph.D., Timo Poranen, Ph.D., and Markku Siermala, Ph.D., for their friendship and support. Also, the department administration has been immensely helpful during my stay there. Special thanks to Tuula Moisio, Marja-Liisa Nurmi, and Minna Parviainen for their help and patience.

The laboratory staff, Minna Kokkonen, Eeva Korhonen, and Pia Lindberg, working at the Hearing Center of Tampere University Hospital collected part of the data used in my work, and Kalevi Nieminen designed and built the portable tilting force platform, a vital component in our studies. Their work is greatly appreciated.

I thank the Tampere Graduate School in Information Science and Engineering, Oskar Öflund Foundation, Finnish Cultural Fund, Finnish Cultural Fund (Pohjois-Savo regional fund), Emil Aaltonen Foundation, Foundation for Technology Promotion, The Finnish Work Environment Fund, The University of Tampere, and Alfred Kordelin Foundation for financial support of my work.

Last, but not least, I wish to thank Maija for her love and support throughout this “project”.

Contents

Publications

I. Tossavainen, T., Juhola, M., Pyykkö, I., Toppila, E., Aalto, H., & Honkavaara, P. (2001), Towards Virtual Reality Stimulation in Force Platform Posturography *in* Patel, V. L., Rogers, R. & Haux, R., eds., 'Proceedings of the 10th World Congress on Medical Informatics', IOS Press, Amsterdam, pp. 854–857.

II. Tossavainen, T., Juhola, M., Pyykkö, I., Aalto, H. & Toppila, E. (2003), 'Development of Virtual Reality Stimuli for Force Platform Posturography', *International Journal of Medical Informatics* **70**, 277–283.

III. Tossavainen, T. (2004), Comparison of CAVE and HMD for Visual Stimulation in Postural Control Research, *in* Westwood, J. D., Haluck, R. S., Hoffman, H. M., Mogel, G. T., Phillips, R. & Robb, R. A., eds., 'Medicine Meets Virtual Reality 12', IOS Press, Amsterdam, pp. 385–387.

IV. Tossavainen, T., Toppila, E., Pyykkö, I., Forsman, P., Juhola, M. & Starck, J. (2005), 'Virtual Reality in Posturography', *IEEE Transactions on Information Technology in Biomedicine*, accepted.

V. Tossavainen, T. (2005), Feature Subset Selection for a Diagnostic Test of Human Balance, *in* Engelbrecht, R., Geissbuhler, A., Lovis, C. & Mihalas, G., eds., 'European Notes in Medical Informatics', EFMI, pp. 1117–1122.

Chapter 1

Introduction

The ability to maintain upright stance is fundamental to many human activities. The standing human body is inherently unstable and requires constant corrective movements to stay upright. A complex control process integrates sensory input, plans the response, and controls muscles to accomplish these movements. Sometimes this process is impaired, which increases the risk of falling over. These conditions are generally known as *balance disorders*. Falling over is a common cause of injury and even death. In the elderly, hip fractures are a cause for concern. Therefore the early detection and treatment of balance disorders are of great importance.

A natural starting point for evaluation of postural stability is to visually observe subjects performing different tasks. Techniques developed for measuring movement or muscle activity are used in more objective studies; see (Winter 1990, ch. 2,4) for review. Observations of a standing subject provide information on how the postural control system handles disturbances emerging from heartbeats, breathing, and other intrinsic factors. However, this approach can not assess the individual contributions of components of the postural control system—external stimulation is needed for this purpose. It can be a moving visual environment or a tilting platform, for example. Comprehensive characterization of a subject's postural control thus requires a number of tests to be conducted under different conditions. From a practical point of view it would be beneficial if many of these test could be carried out using the same hardware.

This thesis describes the development of an integrated stimulation and measurement system for postural control research, motivated specifically by balance disorder research. The measurement and stimulation techniques used, force platform posturography and virtual reality (VR), were specified by my colleagues and my supervisor. The main design goal was versatility. The force platform is a simple non-invasive measurement device that measures forces occurring under the feet of a standing subject. The term *virtual reality* refers to computer-generated immersive environments achieved with computer graphics, stereoscopic displays, and sensors to track movements. VR offers a flexible way to implement and administer visual stimulation, as the same equipment can generate a variety of environments that do not even have to obey the laws of physics.

During the development of the VR posturography system, I worked as a part of a multidisciplinary research group. My colleagues provided the expertise for the

exceedingly complex application domain. Its treatment in this thesis is superficial at best; I have had to treat these aspects mostly as a black box. As a computer scientist, I focus on the technical aspects of the measurement system and the data analysis and model-based interpretations of the measurements, and leave the physiological and medical aspects and interpretations to other members of our group.

The introductory part of this thesis is structured as follows: Chapter 2 gives a brief overview of postural control and its research. Chapters 3 and 4 discuss force platform posturography and computer graphics, the two main components of the measurement and stimulation system developed in this thesis. The implementation and applications are presented in Chapter 5. Finally, Chapter 6 contains the concluding remarks.

We aim to show that virtual reality methods are a versatile stimulation technique for postural control research. The best argument one can give is to obtain results by applying the method to different research problems in postural control. These applications are detailed in Chapter 5. Minor contributions to analysis of stabilograms are made in the process.

Chapter 2

Postural control research

2.1 Postural control process

Postural control is the process of maintaining upright stance. The body is unstable and constantly perturbed by heartbeats and breathing. Balance requires active corrections to posture. A fundamental requirement is that the center of gravity (COG) of the body must remain above the base of support (BOS), the area under and between the feet. The central nervous system (CNS) maintains equilibrium by observing the body state using afferent sensory input and executing appropriate motor control over muscles via efferent motor neurons. This task is difficult as muscles are incapable of producing exactly constant force and sensory feedback is noisy and delayed. The process can be divided into 4 components: the body to be controlled, senses observing it, the controller (CNS) integrating sensory input and issuing motor commands, and muscles executing the motor commands. The control system is hierarchically organized: A central system controls specialized subsystems (Nashner 1985). The postural control process is depicted in Figure 2.1.

2.1.1 Musculoskeletal system

The purpose of muscles in postural control is to move the skeleton supporting the body. Several types of joints connecting bones enable movement. Tendons connect muscles over joints to bones or cartilage. Muscle contraction causes the bones to move around the joints. The degree of freedom in the movement depends on the joint's type; for example, the elbow joints rotate around a single axis, wrists rotate around two axes, and shoulders rotate around three axes.

Muscle fibers (cells) are organized in bundles called fascicles. Inside muscle cells, muscle filaments form contractile elements called sarcomeres. Muscle fibrils (myofibrils) contain sarcomeres connected in series. Muscles also contain parallel connective and series elastic tissue (e.g. tendons) that affects their biomechanical properties. The parallel tissue increases tension non-linearly when the muscle lengthens and the series tissue lengthens slightly when the muscle contracts. (Winter 1990, ch. 7)

A *motor unit* is a set of muscle fibers activated by a single motor axon. Their sizes vary depending on the fineness of control required by a motor task. Motor units function according to the all-or-nothing -principle: they can only contract

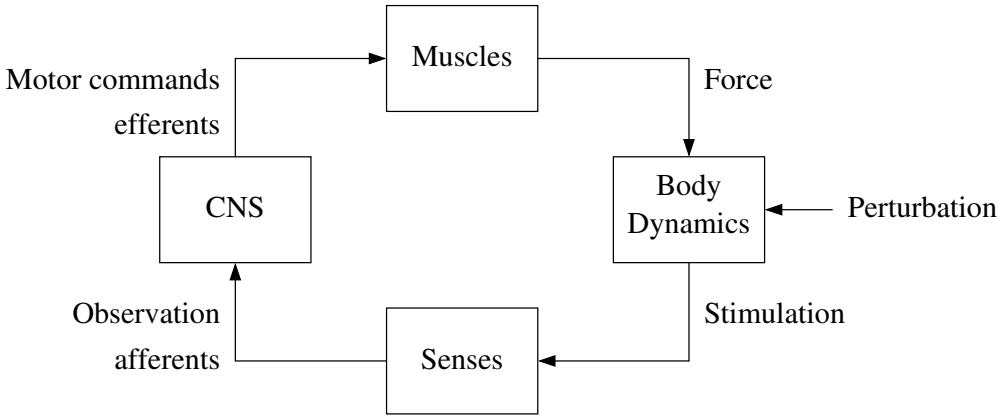


Figure 2.1. Postural control process

fully. Contractions consist of muscle twitches caused by neural impulses. A twitch generated by an impulse approximately follows

$$F(t) = F_0 \frac{t}{T} e^{-t/T},$$

where $F(t)$ is the force as a function of time, T is the contraction time, and F_0 is a constant for the motor unit. The contraction time T varies between slow-twitch and fast-twitch motor units. Each impulse triggers a twitch and the twitches are summed. Tension can be increased by increasing the rate of stimulation or recruiting more motor units. The number and size of activated motor units regulate the force generated by muscles. In general, the motor units activate according to the size principle: smaller units first. This way, movements that require less force have finer control. A new motor unit is recruited after the previous units are functioning near their maximum activation, resulting in smooth increase of tension. Force generated by muscles also depends on their length and velocity. Muscles can only pull (contract), they cannot push. Many muscles pulling in different directions are needed to accomplish movement around a joint. (Winter 1990, ch. 7)

2.1.2 Sensory systems

In control theory, there are two basic types of control: open-loop and closed-loop. Open-loop control does not observe the controlled system while controlling it. It generally does not work as the controlling actions will eventually deviate from the system state if there are any inaccuracies. Closed-loop control uses observations of the system to guide the controlling actions. It is required in unpredictable environments. Sensory feedback is therefore essential for postural control. Human senses are based on complex processing of input from receptors that convert energy to neural impulses. The visual, vestibular, and proprioceptic systems are the most important ones for postural control.

Vision is perhaps the most complex sense. Light travels through the adjustable lens of the eye and hits the retina. On the retina cone and rod receptors convert light to neural impulses. After some processing on the retina, the neural impulses travel through the optic nerver fibers into the lateral geniculate nucleus of the

brain and onto the visual receiving area, striate cortex. Processing in the brain turns this stream of neural impulses into a sensation of a 3-dimensional world. (Goldstein 2002, ch. 2–9)

Proprioception is the sense of body position and movement. It is based on skin mechanoreceptors and muscle proprioceptors. The mechanoreceptors, Merkel receptors, Meisner corpuscles, Ruffini cylinders, and Pacinian corpuscles, respond to touch or pressure (Goldstein 2002, ch. 13). Golgi tendon organs (GTOs) and muscle spindles respond to changes in muscle and tendon tension (Kalat 1984, pp. 183–184).

The vestibular system in the inner ear senses orientation and acceleration of the head. The utricular and ventricular otoliths respond to linear acceleration, for example gravity, and semicircular canals to angular acceleration caused by head rotation. The otoliths comprise hair cells embedded in a jellylike membrane. The cells sense acceleration via the movements of the jelly. Three circular ducts, each lying on a different plane and filled with with endolymphatic fluid, form the semicircular canals. When the head is moved the fluid will also start to move, but with a delay due to inertia. Hair cells in the ducts sense this fluid movement with respect to the head (Carola, Harley & Noback 1990, pp. 447–451). One function of the semicircular canals is to stabilize vision during head movement via the vestibulo-ocular reflex.

2.1.3 Central nervous system

The CNS integrates sensory input to observe the body state. Senses are partially overlapping and redundant for postural control. For example, vision is not essential as we can remain upright and walk with our eyes closed. Information from different senses can be contradictory. These conflicts have to be solved in the integration process. Conceptually, the resolution gives different weights to different senses (Brandt, Paulus & Straube 1986). The approximate frequency ranges of different senses are given in (Redfern, Yardley & Bronstein 2001) as $f < 0.1$ Hz for vision, $f < 0.5$ Hz for otoliths, $0.5 < f < 1.0$ Hz for semicircular canals and $f > 0.1$ Hz for proprioception.

Information of the body state is used in motor control over muscles. The motor neurons receive signals from the pyramidal and the extrapyramidal system. The pyramidal system controls fine movements; its long axons originate in the cerebral cortex and descend through the corticospinal tract into the spinal cord; most of the axons connect to spinal interneurons. The extrapyramidal system, defined as anything in the CNS outside the pyramidal system affecting movement, is responsible for coarse movements such as posture. (Kalat 1984, ch. 7)

2.1.4 Balance disorders

Balance disorders impair the functioning of the postural control system. A deficiency in any component of the postural control loop or a combination of several factors may be the cause. For example, Ménière's disease affects the vestibular system Inflammation of the acoustic or vestibular nerve (vestibular neuronitis) or a

tumor in their vicinity (acousticus neurinoma, for example) may impede neural signals from the vestibular organ into the brain. (Carola et al. 1990, pp. 469). Small dislodged calcium carbonate crystals floating in the semicircular canals and occasionally hitting the nerve cells cause benign positional vertigo (BPV). Lesions of the sensory organs or the brain also cause problems with balance (Hufschmidt, Dichgans, Mauritz & Hufschmidt 1980). Neurodegenerative diseases, such as Parkinson's, degrade transmission of neural signals and lead to motor problems (Mitchell, Collins, Luca, Burrows & Lipsitz 1995). Aging related degeneration may also decrease postural stability.

2.2 Balance measurements

The basic research on postural control and diagnosis of balance problems is based on observations of the postural control process. The simplest methods of diagnosis observe a patient during different tasks. For example, a physician conducting Romberg's[1] test compares the patient's stability in quiet standing with eyes open to stability in quiet standing with eyes closed. The patient is said to exhibit Romberg's sign, or the test is said to be positive, if there is a marked difference in stability.

Modern quantitative approaches use measurements. Cameras, potentiometers, accelerometers, measurements of electrical muscle activity (EMG), and force platforms are common. Posturography measures movements using a force platform. There are two basic subcategories: static and dynamic posturography. Static posturography refers to measurements without external stimulation. Romberg's test is often implemented on the force platform by measuring a patient with eyes open and eyes closed and comparing measures of postural stability in the two conditions. However, the senses are operating beneath their thresholds in quiet standing making it difficult to assess the contribution of different sensory systems.

Dynamic posturography uses external perturbations or altered sensory surroundings to stimulate the postural control process and measures the outcome. For example, the subject may be asked to perform different tasks such as leaning into different directions. Foam rubber placed on a platform causes pressoreceptors of the sole to sense approximately constant pressure. Linearly moving or tilting platforms provide mechanical perturbations (Nashner 1971). Moving mechanical surroundings (Bles & Roos 1991), video (van Asten, Gielen & van der Gon 1988), and more recently using virtual reality methods (Kramer, Roberts, Shelhamer & Zee 1998, Kim, Yoo & Im 1999, Kuno, Kawakita, Kawakami, Miyake & Watanabe 1999, Keshner & Kenyon 2000, Lee, Cherng & Lin 2004) have been used for visual stimulation. Galvanic stimulation has been used to interfere with the vestibular system (Fransson, Hafström, Karlberg, Magnusson, Tjäder & Johansson 2003).

Visual stimulation plays an important part in our research. Brandt et al. (1986) review research concerning moving visual scenes, visual acuity, object distance, light level and stroboscopic illumination. In general they found that lower quality visual input worsens balance. Rotating visual scenes were found to cause leaning. Another review of later research can be found in (Redfern et al. 2001). Two different

[1]Moritz von Romberg (1795–1873)

computer-generated visual environments were used by van Asten et al. (1988) to investigate how the different environments, motion in a tunnel and motion relative to a wall, affect balance. The result was that both environments cause clear effects, both peripheral and central vision contribute to balance, and that the amount of texture present in the scene is an important factor. The last result is important for the design of visual stimuli. The stimulus should contain easily perceivable motion cues. Kramer et al. (1998) duplicate classic eye movement measurements using virtual reality techniques and obtain comparable results. Jacobson, Redfern, Furman, Whitney, Sparto, Wilson & Hodges (2001) present a mini CAVE, called Balance NAVE, based on polarized light for balance investigations. In addition to the publications in this thesis, only a few virtual reality experiments with movement measurements can be found in the literature. Keshner & Kenyon (2000) apply the CAVE to basic research. The system presented in (Lee et al. 2004) for balance testing of children is fairly similar to our construction.

Chapter 3

Posturography

3.1 Force platform

Force platform posturography uses a platform equipped with force transducers. The transducers measure forces supporting an object on the platform—in the case of a test subject, the forces occurring under the feet during movements (Figure 3.1). The device used in our research consists of a platform with three pressure-sensitive sensors (Figure 3.2). The sensors at points **a**, **b**, and **c** form a triangle beneath the platform. They are based on the strain-gauge principle, a part changes resistance in response to pressure, and only register vertical forces. In this section, we discuss the properties of this measurement device; there is no point in measuring anything unless we understand what is being measured. First, let us set up a coordinate system relative to a person on the platform: The x-axis is to the right, y-axis is forward and the z-axis is up. We also assume that the platform lies in the xy-plane and is massless, planar, and rigid.

The term *center of pressure* (COP), or the *center point of force*, ofter occurs in conjunction with force platforms. It is an abstraction of the distribution of forces acting on the platform: The forces can be combined to a single resultant force acting through COP causing the same observed result. A *stabilogram* is the COP as a function of time. The *center of mass* (COM) often refers to the horizontal position of the actual COM.

The platform is held in equilibrium by forces $\mathbf{F}_a$, $\mathbf{F}_b$, and $\mathbf{F}_c$ acting on three non-collinear points **a**, **b**, and **c**. Now, if a force **F** acts on the platform through point **p**, then the conditions for translational and rotational equilibrium are

$$\mathbf{F}_a + \mathbf{F}_b + \mathbf{F}_c + \mathbf{F} = \mathbf{0}, \tag{3.1}$$

$$\mathbf{a} \times \mathbf{F}_a + \mathbf{b} \times \mathbf{F}_b + \mathbf{c} \times \mathbf{F}_c + \mathbf{p} \times \mathbf{F} = 0. \tag{3.2}$$

Expanding equation (3.2) to components gives

$$a_x F_{az} + b_x F_{bz} + c_x F_{cz} = -p_x F_z, \tag{3.3}$$

$$a_y F_{az} + b_y F_{bz} + c_y F_{cz} = -p_y F_z, \tag{3.4}$$

$$\begin{vmatrix} a_x & a_y \\ F_{ax} & F_{ay} \end{vmatrix} + \begin{vmatrix} b_x & b_y \\ F_{bx} & F_{by} \end{vmatrix} + \begin{vmatrix} c_x & c_y \\ F_{cx} & F_{cy} \end{vmatrix} = - \begin{vmatrix} p_x & p_y \\ F_x & F_y \end{vmatrix}. \tag{3.5}$$

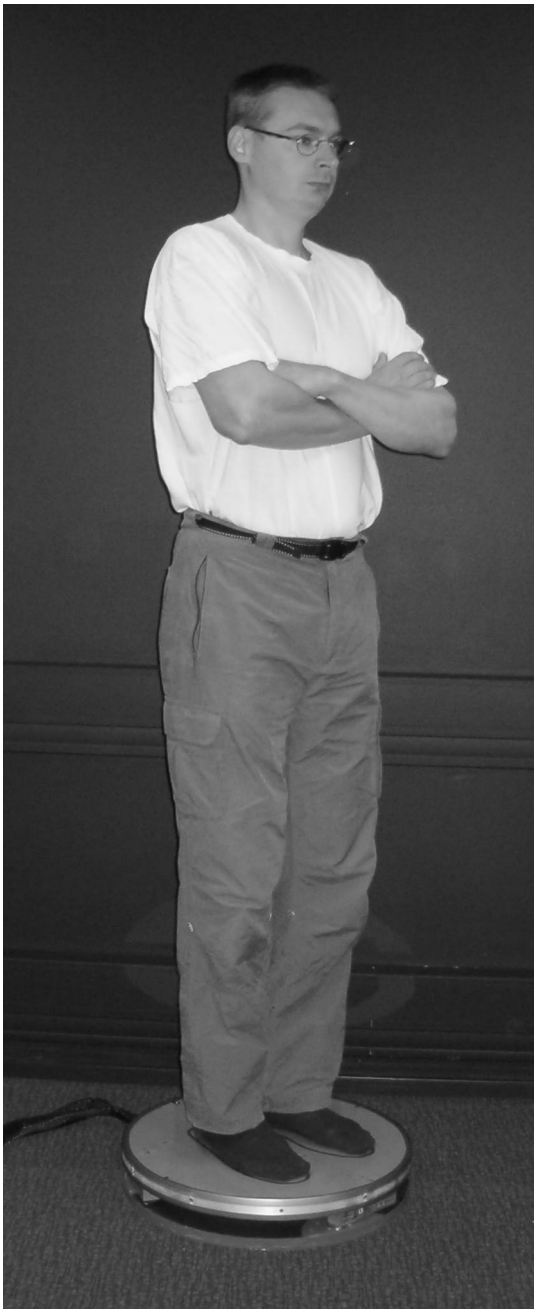


Figure 3.1. A force platform measures the forces exerted by a standing subject while maintaining balance.

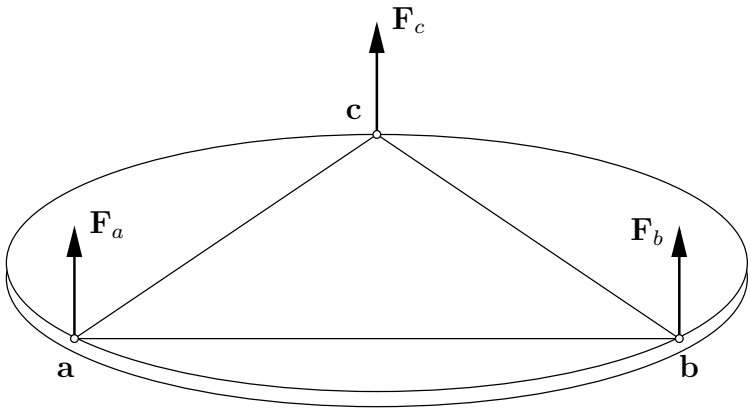


Figure 3.2. A force platform with 3 vertical force sensors measuring supporting forces at points **a**, **b**, and **c**.

The vertical forces satisfy

$$\begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} F_{az} \\ F_{bz} \\ F_{cz} \end{bmatrix} = -F_z \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \tag{3.6}$$

by (3.1), (3.3), and (3.4). Equation (3.6) has an unique solution for the supporting vertical forces, because the points **a**, **b**, and **c** are non-collinear. It also gives **p**, when the vertical forces are known. The horizontal components of $\mathbf{F}_a, \mathbf{F}_b$ and $\mathbf{F}_c$ can not be uniquely determined without additional assumptions. The measured forces are usually transformed into a set of orthogonal components: the COP, seen immediately from (3.6) to be **p**, and $-F_z$ usually close to gravity acting on the object on the platform. In summary, F_{az}, F_{bz}, and F_{cz} are the barycentric coordinates of **p** with respect to the triangle **abc** scaled by $-F_z$.

3.1.1 Calibration

The force transducers measure forces as voltages between -5 and +5 V. Because of the mass of the platform, we can assume that the relation between measurements and vertical force is linear:

$$F_i = G_i(V_i - V_{0,i}), \qquad i = a, b, c,$$

where F_i is the vertical force acting on sensor i, G_i is an unknown gain, V_i is the measured voltage, and $V_{0,i}$ is an unknown DC-offset. Unfortunately, there is no simple way to measure the forces on each sensor while the platform is mounted on them, but their sum is known when a calibrated weight is placed on the platform. We should then have

$$\mathbf{F}_a + \mathbf{F}_b + \mathbf{F}_c = m\mathbf{g},$$

where g is gravitational acceleration and m is the mass of the weight, regardless of the position of the weight on the platform. There is still one minor problem of finding the DC-offsets. A simple solution is to measure the offset with an empty platform and calibrate it to show zero force.

Algorithm 1 Calibration of the force platform using a calibrated weight.

1: Measure DC-offsets $\mathbf{V}_0 = (V_{0,a}, V_{0,b}, V_{0,c})$ with an empty platform.
2: Take a set of measurements

$$\mathbf{V}_i = (V_{a,i}, V_{b,i}, V_{c,i}), \qquad i = 1, 2, \ldots, n,$$

with the weight of mass m at different positions on the platform.
3: Find $\mathbf{G}$ that minimizes $\|\mathbf{M} - \mathbf{V}\mathbf{G}\|^2$, where

$$\mathbf{M} = \begin{bmatrix} mg \\ mg \\ \vdots \\ mg \end{bmatrix} \in \mathbb{R}^n, \quad \mathbf{V} = \begin{bmatrix} \mathbf{V}_1 - \mathbf{V}_0 \\ \mathbf{V}_2 - \mathbf{V}_0 \\ \vdots \\ \mathbf{V}_n - \mathbf{V}_0 \end{bmatrix} \in \mathbb{R}^{n\times 3}, \quad \mathbf{G} = \begin{bmatrix} G_a \\ G_b \\ G_c \end{bmatrix} \in \mathbb{R}^3.$$

The solution is $\mathbf{G} = \mathbf{V}^\dagger\mathbf{M}$, where $\mathbf{V}^\dagger = (\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T$.

Now, using a calibrated mass and moving it on the platform we can obtain a set of measurements with known total mass on the platform. In the ideal situation any three non-collinear measurements are enough to calibrate the platform, but in practice we need more measurements. Setting up an overdetermined system of linear equations from the measurements and DC-offset and finding the minimum squared error (MSE) solution solves the problem. The complete procedure is given in Algorithm 1.

This approach virtually eliminates the possibility of systematic error caused by human error in calibration. It is fast and can be automated so that it only requires moving the weight. Naturally, we have assumed that the sensor positions are known.

3.2 Analysis of stabilograms

Stabilograms measured from test subjects are very irregular. Figures 3.3 and 3.4 show two standard ways to visualize them. Conceptually, the stabilogram summarizes body movements into 3 signals: anteroposterior (AP) and mediolateral (ML) position of COP and the instantaneous weight (or vertical force). This presents some problems as there are, at least in theory, movements that are not visible in the stabilogram. Thus, in most models, let alone in reality, it is impossible to infer the state of the object on the platform from the measurements with certainty. Baratto, Morasso, Re & Spada (2002) even state that there is no widespread consensus on how to analyze the signals.

Force platforms are widely used, for example in rehabilitation, sports, and medicine, and the wide range of application of these measurements is reflected in the variety of methods developed to analyze them. It is a safe bet that most common signal analysis methods have been tried on stabilograms. Stabilograms are usually quantified by computing parameters from them. The choice of parameters depends on the application, and the test setups and equipment vary, so it is often difficult to compare results between research groups (Baratto et al. 2002, Schmid,

Conforto, Camomilla, Cappozzo & D'Alessio 2002).

The signals reflect both the postural control (forces) and the state (position). Next, we look at interpreting the stabilograms in terms of simple physical models.

3.2.1 Physical models

In this section we consider a 2-dimensional force platform in the anteroposterior direction for simplicity. As before, the platform is in equilibrium supported at two points **a** and **b** with forces $\mathbf{F}_a$ and $\mathbf{F}_b$, where we now place the origin at **a** and the point **b** on the x-axis. Here, the platform is effectively approximated by a system of 2 particles. Assuming central forces, forces act along lines connecting particles, and a point mass approximation of the object on the platform, we can derive a simple relationship for COP and COM. If a force $\mathbf{F}$ acts on a particle of mass m at $\mathbf{r}$ through the platform, then the reaction from the platform must be $-\mathbf{F}$ so that

$$\mathbf{F}_a + \mathbf{F}_b - \mathbf{F} = \mathbf{0},$$

where the central forces assumption gives the directions of $\mathbf{F}_a$ and $\mathbf{F}_b$. Solving for F_{bz} gives

$$F_{bz} = \frac{1}{b_x}(F_z r_x - F_x r_z).$$

The fundamental relation linking COP and COM in every model that reduces to a point mass approximation with central forces is

$$p = \frac{F_{bz}}{F_{az} + F_{bz}} b_x = \frac{F_{bz}}{F_z} b_x = r_x - \frac{F_x}{F_z} r_z = r_x - \frac{\ddot{r}_x}{\ddot{r}_z} r_z. \tag{3.7}$$

Here $\ddot{\mathbf{r}}$ is the acceleration caused by forces occurring between the particle and the platform. A geometric interpretation is shown in Figure 3.5. The situation changes slightly when we include gravity as an external force. If the model is not equivalent to a point mass, then the above relationship is not valid. We can then model the standing subject as a system of particles, and the change of angular momentum has to be taken into account (Shimba 1984).

In Gurfinkel's (1973) classic inverted pendulum model, the torso is modeled as a massless rigid rod with a weight on one end. The ankle joint connects the rod to a triangular foot, and torque applied at the ankle controls the pendulum. Suppose an inverted pendulum model of length l is placed on the platform with the ankle at **c** and denote by ϕ the angle between the pendulum and vertical (Figure 3.6). Now, ϕ, $\dot{\phi}$ and $\ddot{\phi}$ are related to COM state by

$$\mathbf{r} = \mathbf{c} + l \begin{bmatrix} -\sin\phi \\ \cos\phi \end{bmatrix}, \qquad \ddot{\mathbf{r}} = l \begin{bmatrix} \dot{\phi}^2 \sin\phi - \ddot{\phi}\cos\phi \\ -\dot{\phi}^2 \cos\phi - \ddot{\phi}\sin\phi \end{bmatrix}.$$

Here $\dot{\phi}$ and $\ddot{\phi}$ are the net angular velocity and acceleration, also affected by gravity $m\mathbf{g}$. The acceleration due to supporting forces on the platform is $\ddot{\mathbf{r}} - \mathbf{g}$ so that

$$p = c_x - l\sin\phi - \frac{l(\dot{\phi}^2 \sin\phi - \ddot{\phi}\cos\phi)}{g - l(\dot{\phi}^2\cos\phi + \ddot{\phi}\sin\phi)}(c_z + l\cos\phi).$$

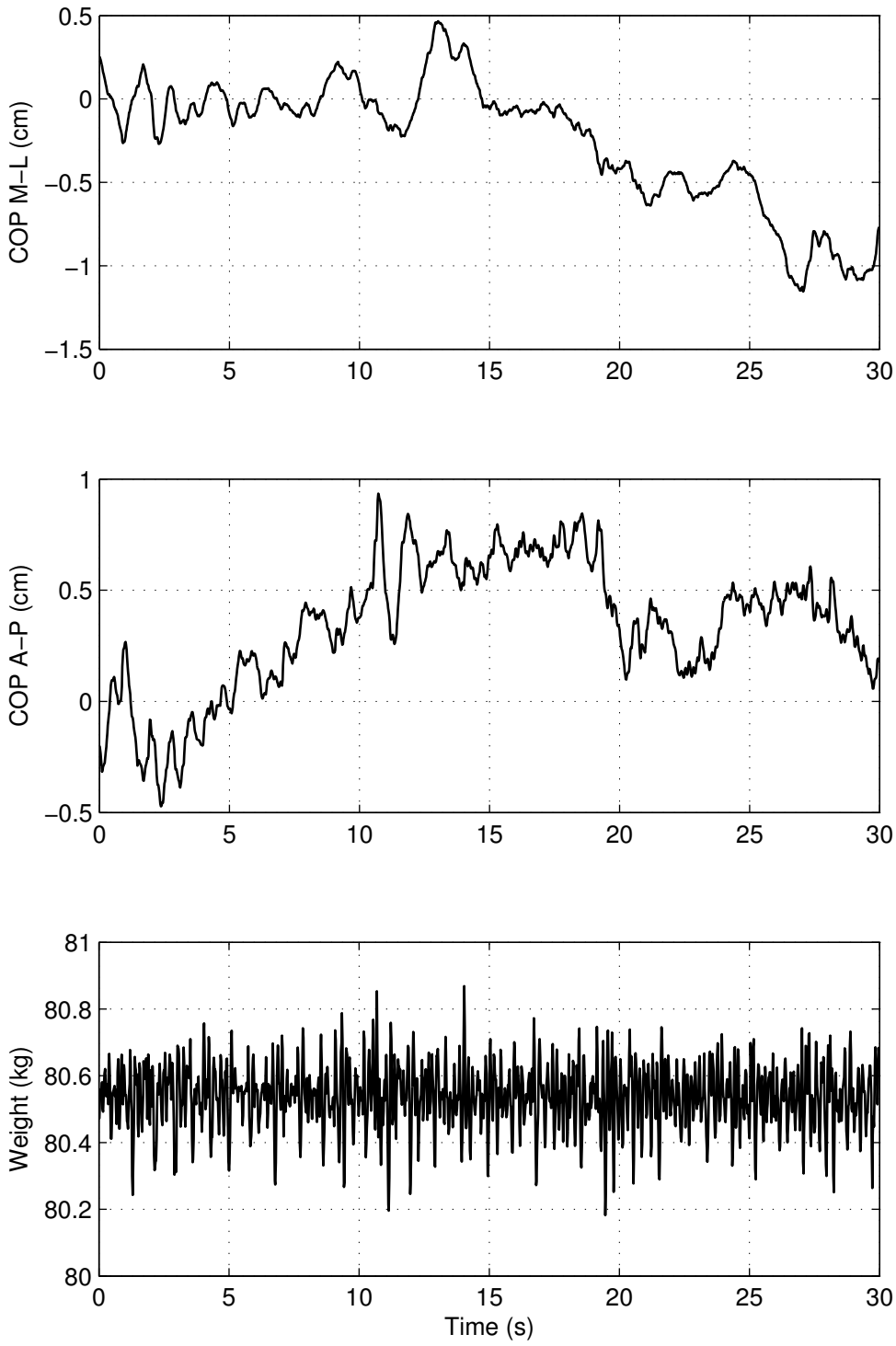


Figure 3.3. Stabilogram measured from a test subject during 30 seconds of quiet standing.

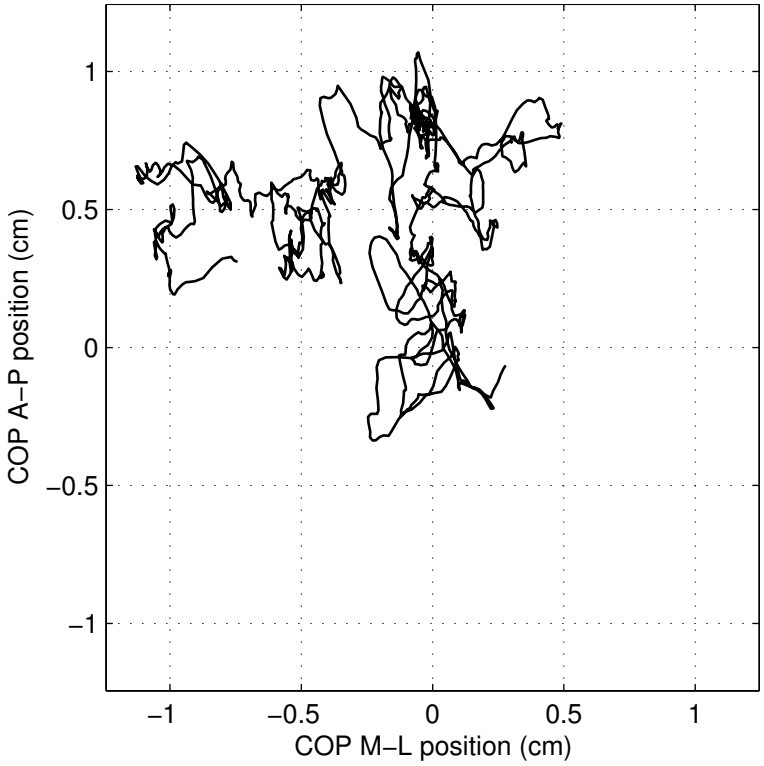


Figure 3.4. Another view of the stabilogram.

by (3.7). Gurfinkel showed that the COP is not always directly below the COM, which was a common misconception at the time. The displacement of COP from under COM is

$$p - r_x = l\frac{c_z(\ddot{\phi}\cos\phi - \dot{\phi}^2\sin\phi) + l\ddot{\phi} - \ddot{\phi}\sin^2\phi - l\dot{\phi}^2\sin\phi\cos\phi}{g - l(\dot{\phi}^2\cos\phi + \ddot{\phi}\sin\phi)}.$$

Of course, equation (3.7) already shows this in simpler form. The pendulum model is often linearized by considering small oscillations around the upright position. If we take $\sin\phi \approx \phi$ and $\cos\phi \approx 1$, then

$$\mathbf{r} = \mathbf{c} + l\begin{bmatrix}-\phi\\1\end{bmatrix}, \qquad \ddot{\mathbf{r}} = -l\begin{bmatrix}\ddot{\phi}\\0\end{bmatrix}.$$

This is (3.7) with $r_z = c_z + l$, $\ddot{r}_x = -l\phi$, and $\ddot{r}_z = g$. Thus, the pendulum model linearized around the upright position is equivalent to a particle at constant height.

Gurfinkel used this model to describe how the stabilogram reflects movements of the COM. Making the same assumption, that COM height h is constant, in (3.7) gives the relation

$$P(\omega) = \left(1 + \frac{h}{g}\omega^2\right)R_x(\omega),$$

where ω is frequency, between the Fourier transforms of $p(t)$ and $r_x(t)$. The relative contribution of COM position to COP at frequency f is given by

$$\left(1 + \frac{4\pi^2 f^2 h}{g}\right)^{-1}.$$

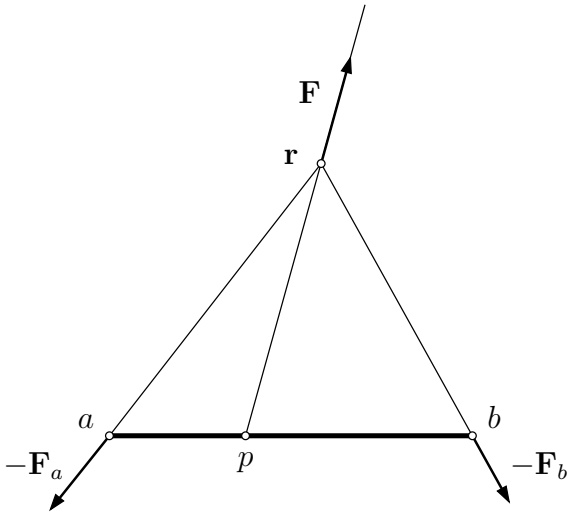


Figure 3.5. Geometric relationship between COP p and COM $\mathbf{r}$.

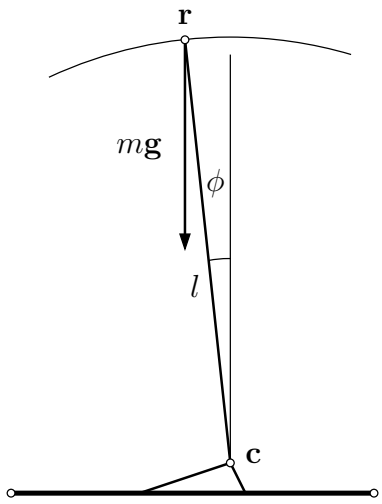


Figure 3.6. Gurfinkel's inverted pendulum model.

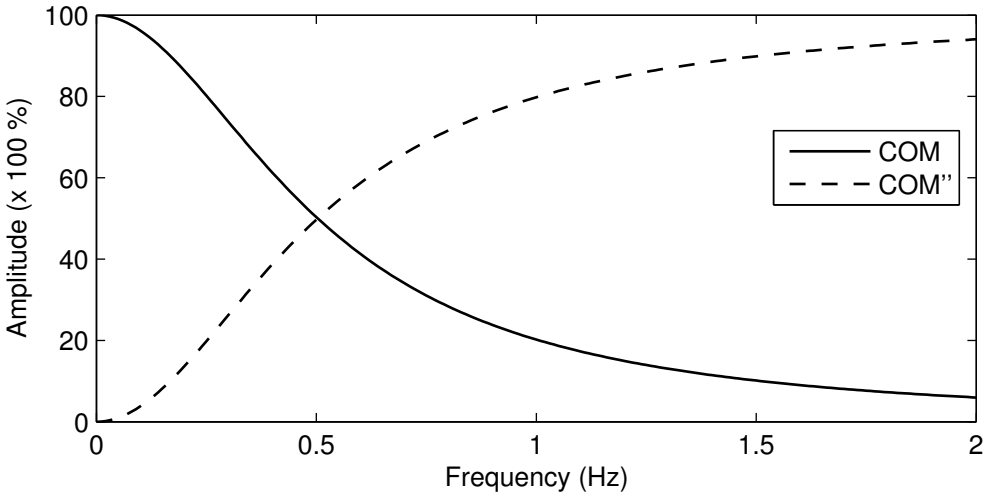


Figure 3.7. Approximate proportions of COM position and acceleration in stabilograms at different frequencies.

Gurfinkel took $h = 1$ m and $g = 10$ m/s^2. Figure 3.7 shows how the proportions change with movement frequency with these assumptions.

3.2.2 Estimation of COM from COP

The relationship between COM and COP is important from a functional point of view: The COM must remain above the base of support, or a step must be taken to avoid falling. Thus, COM position and velocity are the state of a system that must be controlled, and COM accelerations represent the control. If the COM movements can be recovered from stabilograms, then the situation can be analyzed in these more intuitive terms. There are a number of algorithms for this purpose. Here we simplify the notation by denoting the COM horizontal position by x, the continuous signals by $x(t)$ and $p(t)$, and the discrete measured signals by $x(n)$ and $p(n)$, $n = 1, \ldots, N$.

Caron, Faure & Brenière (1997) suggest an FFT filtering approach. They derived a relationship between the Fourier transforms of $x(t)$ and $p(t)$, given by

$$X(\omega) = \frac{\omega_0^2}{\omega_0^2 + \omega^2} P(\omega), \tag{3.8}$$

where ω_0 ranges from 2.8 to 3.2 rad/s. The algorithm uses (3.8) on amplitudes of the discrete Fourier transform (DFT) of $p(n)$ and then returns to the time domain using the inverse DFT. This is bad from a signal processing perspective, as it results in circular convolution. The error should be visible near the end points of the estimated COM curve, if the end points of the COP signal in the analysis window are at different levels. The values of ω_0 correspond to COM heights 0.96–1.25 m in the point mass approximation with $g = 9.81$ m/s^2. The algorithm was most accurate with $\omega_0 = 3.2$ rad/s that corresponds to an average person of height 1.7 m.[1] We develop a time-domain version of this algorithm below.

[1]COM is about 55% of body height from the ground.

Morasso, Spada & Capra (1999) use splines. They assume that the COM movement path is a spline curve, form the corresponding observation from (3.7) assuming constant height, and fit the observation to the stabilogram. The procedure fits the spline to a sliding window over the stabilogram and evaluates it at the window center to extract an estimate of COM position. We also improve this algorithm.

Some force platforms can measure horizontal forces. When the forces are avaible, the position of COM can be recovered by double integration of the measured forces with a guess of the initial state, in theory. Shimba (1984) uses least squares fitting to (3.7) with unknown initial conditions and assuming COM height to be constant. The zero-point-to-zero-point integration method (King & Zatsiorsky 1997, Zatsiorsky & King 1998) also uses horizontal forces, but instead of a single integration over the entire measurement, the algorithm assumes that the COP is directly below COM when the horizontal forces are zero and integrates between these points. This results in shorter numerical integrations. Levin & Mizrahi (1996) use two force platforms and develop a COM motion estimation algorithm that takes the angular momentum into account using a biomechanical model.

Lafond, Duarte & Prince (2004) compare the algorithmic approaches to a method based on 3D kinematic measurements. They find that the lowpass filtering approach does not work as well as the zero-point-to-zero-point integration method. However, in this paper the moments of inertia around the ankle joint in different directions are taken from another paper as $0.0533mH^2$ and $0.0572mH^2$, where m is the mass and H is the height of the test subject. These values are completely unrealistic as this places the COM at about 1/4th of body height.[2] It is also apparent in the figures that the filter is not smoothing enough. Simulations with a 4-segment model have been used to compare the performance of some of these algorithms (Lenzi, Cappello & Chiari 2003).

Caron et al.'s (1997) algorithm can be implemented in the time-domain as follows. It follows from the assumption that COM height is constant, that $\ddot{r}_z = g$ in (3.7). Now, the linear operator $1 - (h/g)\mathrm{d}^2/\mathrm{d}t^2$ relates $x(t)$ to $p(t)$, and finding its inverse and applying it to $p(t)$ gives $x(t)$. This can be done using Laplace transforms (Widder 1989, ch. 13,14), and we can use non-causal processing on measured stabilograms after the measurement. Suppose that $x(t)$ is twice continuously differentiable and that its bilateral Laplace transform $X(s)$ converges in some region overlapping $|\Re(s)| < \sqrt{g/h}$. Transforming (3.7) gives

$$P(s) = X(s) - \frac{h}{g}s^2X(s) = \left(1 - \frac{h}{g}s^2\right)X(s).$$

and

$$X(s) = \frac{g/h}{g/h - s^2}P(s) = \frac{\sqrt{g/h}}{2}\left(\frac{1}{s + \sqrt{g/h}} - \frac{1}{s - \sqrt{g/h}}\right)P(s) \tag{3.9}$$

using partial fractions. Inverting (3.9) in the region $|\Re(s)| < \sqrt{g/h}$ gives the non-causal stable inverse

$$x(t) = \frac{\sqrt{g/h}}{2}e^{-|t|\sqrt{g/h}} * p(t), \tag{3.10}$$

[2]The moment of inertia $I = ml^2$ for a pendulum of length l.

Algorithm 2 Estimation of horizontal COM position $x(n)$ given height h of COM from platform and measured COP $p(n)$ with sampling interval T.

1: $a \leftarrow \exp(-T\sqrt{g/h})$.
2: $f(0) \leftarrow p(1)/(1-a)$.
3: **for** i = 1, 2, ..., N **do** ▷ Filter forward.
4: $\quad f(i) \leftarrow p(i) + af(i-1)$.
5: $b(N+1) \leftarrow p(N)/(1-a)$.
6: **for** i = N, N-1, ..., 1 **do** ▷ Filter backward.
7: $\quad b(i) \leftarrow p(i) + ab(i+1)$.
8: **for** i = 1, 2, ..., N **do**
9: $\quad \hat{x}(i) \leftarrow T\sqrt{gh^{-1}}(f(i) + b(i) - p(i))/2$.

where '$*$' denotes convolution (Figure 3.8). Discretizing (3.10) using the impulse invariance method (Proakis & Manolakis 1996, pp. 671–676) gives the discrete convolution

$$\hat{x}(n) = \frac{T\sqrt{g/h}}{2} e^{-|nT|\sqrt{g/h}} * p(n), \tag{3.11}$$

where T is the sampling interval. The impulse invariance method gives a good match of the continuous and discrete filter responses at low frequencies, provided that the sampling rate is high enough and that the filter response decays rapidly with increasing frequency. The sampling of the impulse response causes aliasing if it is not bandlimited below the Nyquist frequency[3]. The bilinear transformation can also be used to discretize the continuous filters on the right hand side of (3.9). Figure 3.9 shows a comparison of the frequency response of this discretized filter to the ideal one; the sampling frequency is 50 Hz as used in all experiments in this thesis. The convolution (3.11) can be implemented efficiently with Algorithm 2.

Algorithm 2 decomposes the convolving double exponential into two exponentials and computes (3.11) by forward and backward filtering. Subtraction of $p(n)$ removes a double accumulation at time lag 0, that results from the filterings. The algorithm is given in this concrete form to make it easy to implement even without knowledge of signal processing. The individual 1-pole IIR filters have the transfer function

$$H(z) = \frac{1}{1 - az^{-1}},$$

and the initial states $f(0)$ and $b(N+1)$ were set by assuming the signal to be constant before and after the measurement. This extends $p(n)$ to infinity by replicating the first and last samples. The initial states represent the contribution of signal outside the measurement. Compared to Caron et al.'s (1997) $O(N \log N)$ algorithm, this is an $O(N)$ algorithm; it is much faster in practice, and it corrects the circular convolution defect. An example of the result is in Figure 3.10. The extension of $p(n)$ can also be done for the DFT-based algorithm by padding the beginning and the end of $p(n)$ for a duration of about 2 seconds with the first and last samples, respectively, and then discarding the padding from $\hat{x}(n)$. Ideally, when using this

[3]The Nyquist frequency is half the sampling frequency. Continuous signals bandlimited below it can be unambiguously reconstructed from their sampled representations by the sampling theorem.

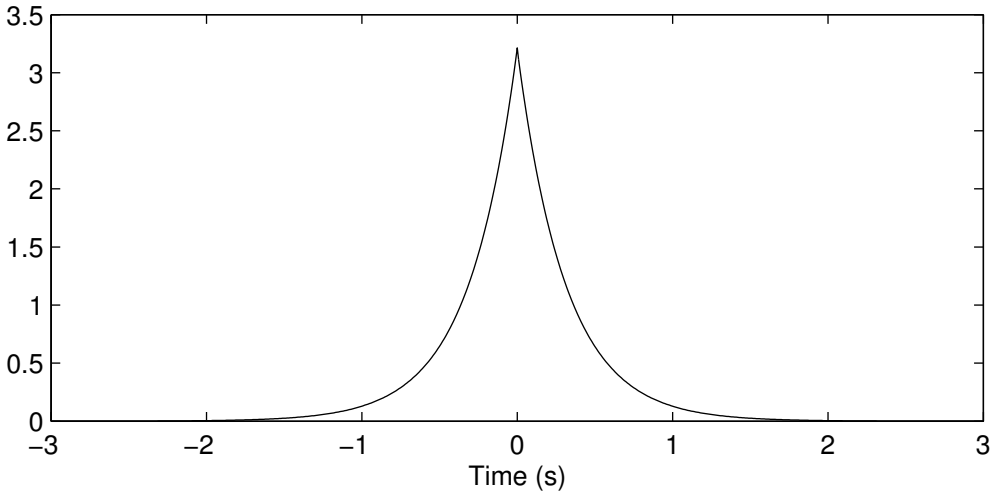


Figure 3.8. Impulse response of the ideal filter to recover COM from COP.

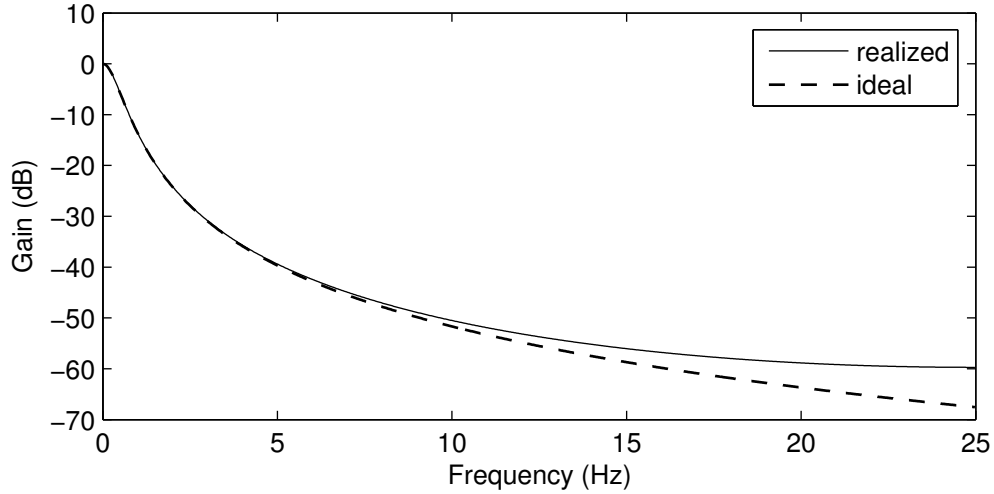


Figure 3.9. Frequency response of realized (discrete) and ideal filter for estimating COM from COP. Higher frequencies differ because of aliasing from the impulse invariant discretization.

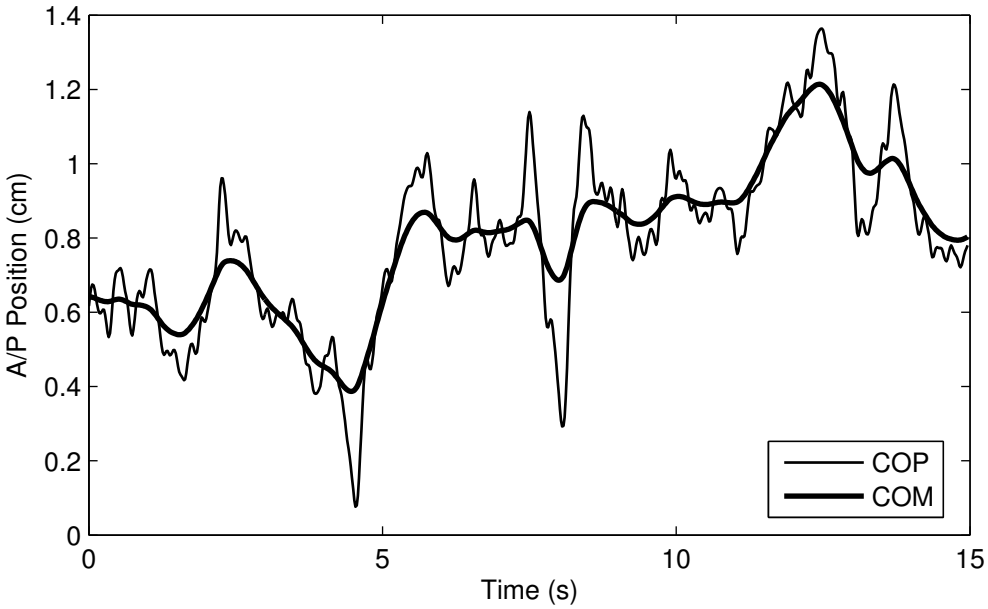


Figure 3.10. Result of COM estimation from stabilogram.

algorithm, we should discard, in addition, about 2 seconds of estimated COM from both ends of the signal so that the estimate is based on measured data. The time-domain version is less accurate than the DFT-based method when the signal is long, but the difference can be made negligible with a suitable discretization of the continous filters.

Morasso et al.'s (1999) algorithm uses (3.7) and solves the curve fitting problem

$$\left(\mathbf{B} - \frac{h}{g}\ddot{\mathbf{B}}\right)\mathbf{a} \approx \mathbf{p},$$

where $\mathbf{B}$ is a matrix of basis functions (B-splines in their case) evaluated at the points to be fitted, $\ddot{\mathbf{B}}$ is a matrix of the corresponding second derivatives, and $\mathbf{a}$ is a vector of weights, for a sliding window $\mathbf{p}$ over the stabilogram. The result $\hat{x}$ at each window position is the spline evaluated at center of the window. The authors state that in the solution

$$\mathbf{a} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{p},$$

where $\mathbf{A} = \mathbf{B} - hg^{-1}\ddot{\mathbf{B}}$, the pseudoinverse $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ can be computed only once. Nevertheless, this is still an $O(bwN)$ algorithm, where b is the number of columns in $\mathbf{B}$, and w is the length of the signal window. If we place the values of the basis functions at the window center in $\mathbf{b}^T$, then

$$\hat{x} = \mathbf{b}^T(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{p}.$$

Now, $\mathbf{b}^T(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is a row vector, so the algorithm reduces to linear FIR filtering, unless the knot positions change. This possibility was mentioned, but not discussed in the paper. The time complexity improves to $O(wN)$. In the case of constant knot positions, the algorithm contains an unnecessary matrix multiplication when a simple dot product suffices. The authors suggest to use one B-spline

knot every 100–150 ms for a time window of about 2 s. Cubic B-splines with 20 knots have 22 parameters; the reduction to FIR filtering is about 20 times faster for the recommended settings. Also, the algorithm is revealed to be a simple linear filter with a complex rationale. It can do no better than a directly designed one.

3.2.3 Parameterization of stabilograms

Researchers often simplify the analysis of stabilograms by computing parameters to summarize their properties. Baratto et al. (2002) survey some of the parameters and group them into two groups: global and structural. Global parameters quantify the "size" of the stabilogram, and structural parameters consider the dynamics or the structure of the process.

Classical analysis methods try to quantify the amount of swaying from the stabilogram. Their rationale is roughly that more swaying means worse balance. The parameters are heuristic in nature and measure the size of the stabilogram or the level of activity in some sense. For example, Hufschmidt et al. (1980) use *mean amplitude* (MA), *sway path* (SP), and *mean frequency* (MF) to discriminate between patients with cerebellar and labyrinthine lesions. MA is the average distance of COP from its average position, SP is the length of the curve traced by COP, and MF is the number of complete circles in a circular motion on a circle with radius MA with a distance traveled SP. The parameters are defined

$$\mathrm{AMP} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{p}(n), \qquad \mathrm{MA} = \frac{1}{N}\sum_{n=1}^{N} d(\mathbf{p}(n), \mathrm{AMP}),$$
$$\mathrm{SP} = \sum_{n=2}^{N} d(\mathbf{p}(n), \mathbf{p}(n-1)), \qquad \mathrm{MF} = \frac{\mathrm{SP}}{2\pi \mathrm{MA}},$$

and the interpretation is that MA measures stability and SP and MF measure controlling activity. The parameters are usually normalized by measurement duration. The authors also consider MA, MF, and SP restricted to anteroposterior and mediolateral movements, quotients of parameters measured under different conditions, and histograms of COP movement angles. Sway path is also called trace length and mean velocity (MV), when normalized with the measurement duration.

Bräuer and Seidel have applied power spectral density estimation and parametric time series modelling to stabilograms, and have suggested their use to assess biomechanical and interindividual differences, functional changes in controlling systems, efficiency of control, and sensitivity of otoliths. They used autoregressive models to explain about 60–94% of the variance of suitably highpass filtered stabilograms (Bräuer & Seidel 1978), interpreted model order as complexity of control (Seidel, Bräuer, Bastek & Issel 1978), linked the frequency ranges $0.025 < f < 0.2$ Hz, $f > 1$ Hz, $0.025 < f < 0.1$ Hz to COM movements, COM accelerations, and otolith functionality, respectively (Bräuer & Seidel 1979), and demonstrated significant differences in autoregressive models of stabilograms measured under varying conditions: different foot positions, after of fatiguing load, and consciously controlling sway (Bräuer & Seidel 1980). Kuczyński (1999) modeled COM accelerations,

that were estimated by filtering, using a second order autoregressive model and converted the parameters to stability margin and peak frequency for analysis.

Roy, Ladin & De Luca (1987) criticize the standard analysis methods of stabilograms for neglecting the dynamics of the process. They suggest a two-dimensional bounded Brownian motion model for stabilograms, and introduce diffusion plots and angle histograms to analyze the movements. A diffusion plot shows the mean squared distance between COP positions as a function of time difference, that is mean $d^2(\mathbf{p}(t), \mathbf{p}(t + \Delta t)))$ against Δt. Collins & De Luca (1993) refine these ideas and present stabilogram diffusion analysis. They model the stabilogram as normal and fractional Brownian motion (fBm). In standard Brownian motion, displacements obey

$$E(\Delta x)^2 = 2D\Delta t,$$

where D is the diffusion coefficient and E denotes mathematical expectation, and in fractional Brownian motions, they obey

$$E(\Delta x)^2 = (\Delta t)^{2H},$$

where H is the scaling exponent. The displacements are uncorrelated for $H = 1/2$ (classical random walk), negatively correlated for $H < 1/2$ (tend to be in opposite directions), and positively correlated for $H > 1/2$ (tend to continue in the same direction). The second behavior is *anti-persistent*, and the third behavior is *persistent*. Collins and De Luca use slopes of lines fitted to linear and log-log diffusion plots as estimates of diffusion coefficients and scaling exponents, respectively. Diffusion plots computed from measurements show two scaling regions: First the COP displacement increases quickly and then its increase slows down. COP movement can be considered persistent in the short term and antipersistent in the long term. Collins and De Luca speculate that the scaling regions correspond to short term open-loop and long term closed-loop control. The parameters from the analysis are diffusion coefficients and scaling exponents for both scaling regions and critical point coordinates. The critical point coordinates are the squared distance and time when the scaling region changes.

Collins & De Luca (1994) consider the possibility that stabilograms are chaotic but deterministic. They are unable to distinguish phase-randomized surrogates from the original data using the analyses, however, and conclude that stabilograms should be modeled as bounded random walks as done in their previous studies. More elaboration on the stabilogram diffusion analysis can be found in (Collins & De Luca 1995*c*). The pinned polymer model of posture control has been proposed to explain the observed diffusion plots (Chow & Collins 1995), and a simple PID controller with a time delay and disturbing noise also produces similar observations (Peterka 2000). Stabilogram diffusion analysis has become quite popular. It has been applied in research of Parkinson's disease (Mitchell et al. 1995), contribution of vision to postural control (Collins & De Luca 1995*b*), and effects of spaceflight on postural control (Collins & De Luca 1995*a*), for example. Chiari, Cappello, Lenzi & Croce (2000) parameterize the diffusion plots differently: They replace the critical point coordinates with constant factors of lines fitted to log-log diffusion plots.

More exotic methods to analyze stabilograms include correlation dimension and Lyapunov exponents (Yamada 1995), recurrence quantification analysis (Riley,

Balasubramaniam & Turvey 1999), time-frequency analysis (Schumann, Redfern, Furman, El-Jaroudi & Chaparro 1995, Ferdjallah, Harris & Wertsch 1999), sway-density (Baratto et al. 2002) and statistics of COP displacements of local averages (Grzegorzewski & Kowalczyk 2001). Önell (2000) analyzes peaks in vertical force.

System identification builds models of input-output-relationships in systems from known inputs and observed outputs. It requires an observable perturbation to be applied to the unknown system and observing the response, and is therefore mostly applicable to dynamic posturography. The resulting models can predict responses to perturbations, and their parameters parameterize the stabilograms. Maki (1986) discusses selection of suitable perturbations, and applies system identification model effects of linear displacements of a moving platform with moving surroundings (Maki & Fernie 1988). Johansson, Magnusson & Åkesson (1988) perturb test subjects with vibrators placed on calf muscles, model the subject as an inverted pendulum under PID control with disturbances, and identify the controller parameters called swiftness, stiffness, and damping to describe the postural control process. There are also system identification methods to analyze simultaneous application of multiple stimuli to evaluate contributions of individual sensory systems (Johansson, Magnusson, Fransson & Karlberg 2001).

Given the number of different parameters, there is reason to suspect that they are at least partially redundant. For example, the standard deviation of horizontal ground reaction force, the standard deviation of COP, and MV are significantly correlated, but they are not significantly correlated with a movement strategy score and the standard deviation of vertical ground reaction force (Karlsson & Frykberg 2000). Rocchi, Chiari & Cappello (2004) use principal components analysis to select 10 parameters from a set of 37 parameters to characterize the stabilograms. Some of the parameters were computed separately in AP and ML directions and including both directions. The selected parameters were COP RMS (combined, AP, ML), low frequency interval containing 95 % of signal power (combined, AP), mean velocity (AP, ML), frequency dispersion (combined, AP), and angular deviation from AP sway.

3.2.4 Data acquisition issues

Factors such as sampling rate, sampling duration, filtering prior to analysis, subject antrophometry, and fitness affect parameters computed from stabilograms, which has to be taken into account when designing experiments. There may be transients in the beginning of a measurement, and variability may be high in short measurements; fatigue is an issue in longer measurements. Recommended trial durations range from 20–30 seconds (Le Clair & Riach 1996) to 60 seconds (Carpenter, Frank, Winter & Peysar 2001).

Filtering removes noise, but it also has an effect on the parameters. Schmid et al. (2002) treat it in detail and recommend a cutoff frequency of 10 Hz. However, they use forward-backward filtering to ensure a linear phase response with an FIR filter. This is unnecessary, as FIR filters can be designed to have a linear phase response, which is the usual reason for their use. A filter that has a linear phase response only affects the amplitudes of sinusoids making up the input signal and delays the output.

Simply shifting the output signal can compensate for the delay. Forward-backward filtering is typically used to ensure that IIR (not FIR) filters have a zero-phase response, and it has the effect of squaring the magnitude response. These facts cast some doubt on whether the recommended cutoff frequency really is 10 Hz or somewhat lower.

Chiari, Rocchi & Cappello (2002) consider the effects of 17 biomechanical parameters on 55 different stabilometric parameters. They find that height, weight, foot width, area of base of support, and angle of feet affect the results the most, and can explain up to 50% of variation in some parameters.

Chapter 4

Computer graphics and virtual reality

4.1 Introduction

Computer graphics is probably one of the most researched fields in computer science. The aim of this chapter is to give a short conceptual introduction to modern realtime rendering and to showcase some of the problem-solving methods. Specifically we will develop a simplified abstract model of how a graphics pipeline works, deal with speeding up rendering with certain higher level optimizations, and briefly discuss motion specification for animation.

The goal of computer graphics is to take a specification of an environment and turn it into an image. Eyes detect the influx of light from the surroundings. For a picture of a scene to be perceived on some surface, for example a painting or on a monitor screen, it has to emit or reflect light as the scene to be displayed would. Most displays are reasonably planar and rectangular, so we take the surface to be a rectangle. The plane containing the rectangle is called the *view plane* and the rectangle is called the *view plane window*. Practical displays are not infinitely accurate, but use a regular grid, *raster grid*, of picture elements, *pixels*, that emit light. Compared to an ideal continuous image, this discretization causes some artifacts as fast variations can not be captured accurately.

We assume that i) the world consists of objects (medium) delineated by surfaces, ii) light travels between and inside objects in straight lines, iii) the direction of light may only change by hitting surfaces of objects and iv) light may enter and leave an object only by passing through its surface. It follows that light reaches the eye directly from some point on some surface and there is a one-to-one correspondence between rays of light entering the eye and points on visible surfaces. If we consider the eye as a point at the origin looking down the z-axis through the view plane at $z = 1$ then light arriving to the eye from a point (x, y, z) passes through the point $(x/z, y/z, 1)$ on the view plane. The mapping from points to the view plane is called *perspective projection* (Figure 4.1). Sending from each point of the view plane window a ray of light similar to that which would reach the eye through that point gives a perfect picture of the scenery. Of course, taking the eye to be a point is a simplification and neglects some of the properties of the visual system. For

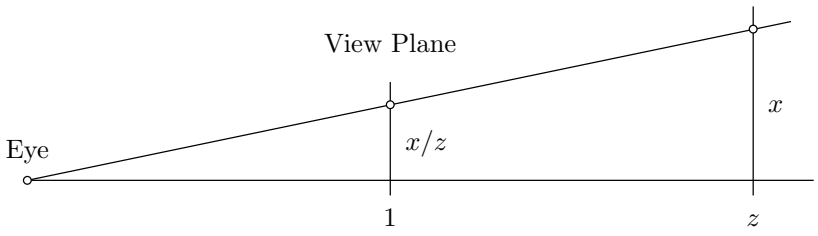


Figure 4.1. Perspective projection maps points on lines originating from the eye to the view plane.

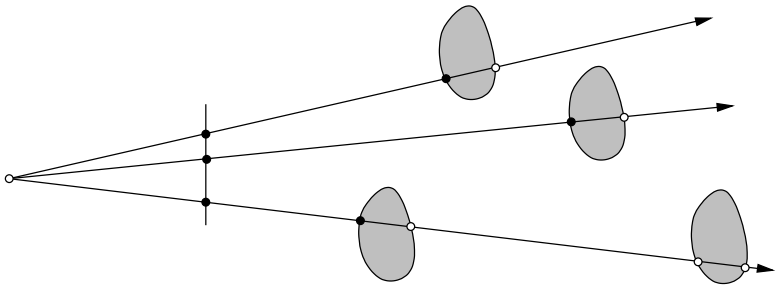


Figure 4.2. Visibility.

example, the resulting picture will be in perfect focus for all distances.

It follows from the assumptions that light reaches the eye only from surface points closest to it along each ray through the view plane. Determination of these points and the corresponding surfaces given the viewpoint is known as the *visibility problem* in computer graphics. One simple solution is *ray casting*. The closest surface point along a ray cast from the eye is visible (Figure 4.2). If $(x, y, 1)$ is a point on the view plane, then all points that can reflect light from that direction lie on the ray $\alpha(x, y, 1)$, $\alpha > 0$. For each point $(x, y, 1)$ in the view plane window, we want to find the smallest $\alpha > 0$ such that the point $\alpha(x, y, 1)$ is on some surface and the corresponding surface if such a point and surface exists.

Computation of the aforementioned intersections requires a suitable representation of surfaces. This representation should be able to approximate the objects to be displayed and the resulting computations should be efficient. Of course, it is preferable to use few primitives. Triangles are most common. They are inefficient for representing curved surfaces, but this is made up by the simplicity of the resulting algorithms. There are also methods to simulate the appearance of curved surfaces using triangles. Under the most common transformations—scaling, rotation, translation, and perspective projection—the images of triangles are still triangles, and only the images of vertices are needed in the mapping.

Color and intensity of light reflected from a surface gives the surface its appearance. Properties of surfaces as light transmitters is the topic of *local reflection* models, we might also use the term *material modelling*. In reality each surface receives light from the surroundings as light travels from one surface to another. Thus, the whole scene has to be considered to make a realistic image; the approach is called *global illumination*. The archetypal global illumination algorithms are ray tracing (Whitted 1980) and radiosity (Goral, Torrance, Greenberg & Battaile 1984). Re-

altime rendering usually only considers *local illumination*, where light reaches each surface directly from light sources. Rest of the scene, for example intervening surfaces that might cast shadows, are not considered in these models, but their effect can be simulated in other ways.

4.2 Mathematical preliminaries

A suitable mathematical formalism is needed to describe the relationships of objects in space and perform computations such as determining visibility. Mathematics in computer graphics is a mixture of different fields with seemingly little standardization. Most authors of standard textbooks, for example Foley, van Dam, Feiner & Hughes (1989) and Watt (2000), prefer an informal approach. Without question, calculus, elementary linear algebra, and geometry are the foundation of computer graphics, but results and formalisms from a variety of other fields of mathematics are frequently used. Rigorous development of a suitable formalism would take us too far afield. Hence, the reader is assumed to have at least some familiarity with linear algebra and calculus.

An abstract axiomatic approach develops a formalism based on algebraic properties of objects. Goldman (2002) compares a few possible approaches and introduces Grassman spaces as a suitable model. Quite often the formalism contains a clean separation of points and vectors, e.g. in affine spaces (Godement 1968, ch. 25) a vector space of translations acts on a set of points. Goldman (1985) discusses interpretations of these expressions in computer graphics. A grand unified theory will not be pursued here; instead, our approach will be concrete. The subject matter, a model of the graphics pipeline, is so simple that the added abstraction is not really worth it. More mathematical sophistication will be needed to discuss animation. Computations will be in Euclidean space and employing Cartesian coordinates where it simplifies the discussion.

A brief review of linear algebra and Euclidean spaces is in order to introduce notation. We denote vectors by $\mathbf{u}, \mathbf{v}, \mathbf{w}, \cdots \in \mathbb{R}^n$, scalars by $\alpha, \beta, \gamma, \cdots \in \mathbb{R}$ and linear mappings $\mathbb{R}^n \to \mathbb{R}^m$ and their corresponding matrices by $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$. We use column vectors as is customary. The components of a vector $\mathbf{u}$ are $u_1, \ldots, u_n$. We write $\mathbf{u} \cdot \mathbf{v}$ for the standard inner (or dot) product. The Euclidean norm is denoted by $\|\mathbf{u}\|$ and the Euclidean distance by $d(\mathbf{u}, \mathbf{v})$. Two vectors $\mathbf{u}$ and $\mathbf{v}$ are *orthogonal* if $\mathbf{u} \cdot \mathbf{v} = 0$, we then write $\mathbf{u} \perp \mathbf{v}$. A set of vectors is *orthogonal*, if all its members are pairwise orthogonal, and *orthonormal*, if, in addition, they have unit norm. The inner product, the norm, and the metric are related by

$$\|\mathbf{u}\|^2 = \mathbf{u} \cdot \mathbf{u}, \tag{4.1}$$

$$d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|. \tag{4.2}$$

Euclidean n-space is the linear space $\mathbb{R}^n$ together with the standard inner product and the norm and metric the product induces by (4.1) and (4.2). We shall also use the cross-product $\mathbf{u} \times \mathbf{v}$, which is usually not considered in linear algebra, but is useful for many computations in $\mathbb{R}^3$.

The form $\alpha\mathbf{u} + \beta\mathbf{v}$ is called a *linear combination* of $\mathbf{u}$ and $\mathbf{v}$. The combination is *affine*, if $\alpha + \beta = 1$, and *convex*, if, in addition, $\alpha, \beta \geq 0$. Combinations of more

than two vectors are defined as iterated combinations of two vectors. A subset of $\mathbb{R}^n$ is a linear space, an affine space, or a convex set if it is closed under corresponding finite combinations. A set's *linear span*, *affine span*, and *convex hull* are the smallest linear, affine, and convex sets containing it, respectively. A set of nonzero vectors is *linearly independent* if none of its vectors is the linear combination of the others. Similarly, we define *affine* and *convex independence*. A linearly independent set is called a *basis* for its linear span. The *dimension* of a linear space is the number of vectors in its basis. The vectors $\mathbf{e}_i$, whose ith element $= 1$ and the rest are zero, form the standard basis for $\mathbb{R}^n$. Sets of the form $X + \mathbf{u} = \{\mathbf{u} + \mathbf{v} : \mathbf{v} \in X\}$ are called *translates* of X. If X is an affine subspace and $\mathbf{u} \in X$ then $X - \mathbf{u}$ is a linear subspace. Conversely, if X is linear then $X + \mathbf{u}$ is affine for any $\mathbf{u}$. Affine subspaces are translates of linear subspaces.

Homogeneous coordinates have to be mentioned as they are frequently used in computer graphics, even though we shall not use them. Linear transformations, translations, and projective transformations in $\mathbb{R}^3$ can be expressed as linear transformations of $\mathbb{R}^4$ by identifying $(u_1, u_2, u_3, u_4) \in \mathbb{R}^4$, $u_4 \neq 0$, with $u_4^{-1}(u_1, u_2, u_3) \in \mathbb{R}^3$. The resulting structure is a representation of a projective plane, that has some topological oddities to be observed particularly in clipping. The coordinates can be given extra meaning. For example, free vectors can be represented in homogeneous coordinates by $u_4 = 0$ so that translations have no effect on them (Goldman 1985).

4.2.1 Geometric primitives

To perform the essential geometric computations in computer graphics, we need clear definitions of geometric objects and results on their properties. After this, most problems can be solved by straightforward calculations. Objects are usually given in two basic forms: parametric or implicit. Here we briefly discuss the most common objects, lines, segments, planes, and triangles, and mention some others in passing.

The affine span of $\mathbf{u}$ and $\mathbf{v}$ is a *line through* $\mathbf{u}$ *and* $\mathbf{v}$ and their convex hull $\mathbf{uv}$ a *line segment* with *end points* $\mathbf{u}$ *and* $\mathbf{v}$, if $\mathbf{u} \neq \mathbf{v}$. Points on the line are given by

$$(1 - \alpha)\mathbf{u} + \alpha\mathbf{v}, \quad (\alpha \in \mathbb{R}),$$

and points on the segment similarly with $0 \leq \alpha \leq 1$. Rays from $\mathbf{u}$ through $\mathbf{v}$ are similar objects with the restriction $\alpha \geq 0$. Lines, rays, and line segments are images of the real line and its intervals; their properties follow almost directly from properties of real numbers. Requirements like $\mathbf{u} \neq \mathbf{v}$ above are often violated in computer graphics resulting in *degenerate* objects. For example, a point is a degenerate line segment.

The affine span of any three affinely independent points $\mathbf{u}, \mathbf{v}, \mathbf{w}$ is called a *plane* and their convex hull $\mathbf{uvw}$ is a *triangle*. All points $\mathbf{x}$ on the plane are given by

$$\mathbf{x} = \alpha_1\mathbf{u} + \alpha_2\mathbf{v} + \alpha_3\mathbf{w}, \tag{4.3}$$

for some $\alpha_1 + \alpha_2 + \alpha_3 = 1$. The coefficients α_1, α_2, and α_3 in (4.3) are called the *barycentric coordinates of* $\mathbf{x}$ *with respect to* $\mathbf{uvw}$. The points $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are the *vertices* and the line segments $\mathbf{uv}$, $\mathbf{vw}$, and $\mathbf{wu}$ are the *edges* of the triangle $\mathbf{uvw}$. A

triangle consists of the points that have non-negative barycentric coordinates with respect to it; edges have, in addition, one barycentric coordinate = 0 and vertices have one coordinate = 1. Barycentric coordinates are useful in certain calculations involving triangles.

The implicit or normal form of a plane is actually used more than the parametric form (4.3). This form is quite easy to derive. Suppose that $\mathbf{x} \in \mathbb{R}^3$ is on the plane. Then $\mathbf{x}$ satisfies

$$\mathbf{x} - \mathbf{u} = \alpha_2(\mathbf{v} - \mathbf{u}) + \alpha_3(\mathbf{w} - \mathbf{u}). \tag{4.4}$$

for some α_2, α_3. To determine if a given point $\mathbf{x}$ actually satisfies (4.4) we need to solve the corresponding linear equation (4.4), which is underdetermined in $\mathbb{R}^3$. It can be completed into an equation that is always solvable using the cross product as the set $\{\mathbf{v} - \mathbf{u}, \mathbf{w} - \mathbf{u}, (\mathbf{v} - \mathbf{u}) \times (\mathbf{w} - \mathbf{u})\}$ is a basis for $\mathbb{R}^3$. Thus, if $\beta = 0$ in

$$\mathbf{x} - \mathbf{u} = \alpha_2(\mathbf{v} - \mathbf{u}) + \alpha_3(\mathbf{w} - \mathbf{u}) + \beta(\mathbf{v} - \mathbf{u}) \times (\mathbf{w} - \mathbf{u}),$$

then we know that $\mathbf{x}$ is on the plane. This occurs precisely when

$$(\mathbf{x} - \mathbf{u}) \perp (\mathbf{v} - \mathbf{u}) \times (\mathbf{w} - \mathbf{u})$$

because $(\mathbf{v} - \mathbf{u}) \times (\mathbf{w} - \mathbf{u})$ is orthogonal to $\mathbf{v} - \mathbf{u}$ and $\mathbf{w} - \mathbf{u}$. Thus, all points $\mathbf{x}$ on the plane, and only those, satisfy an equation of the form

$$\mathbf{x} \cdot \mathbf{n} = \alpha, \tag{4.5}$$

where $\mathbf{n}$ is a vector called the normal of the plane. Another sometimes useful form is based on the metric. The points $\mathbf{x}$ that satistfy $d(\mathbf{x}, \mathbf{u}) = d(\mathbf{x}, \mathbf{v})$ lie on the plane that goes through the midpoint between $\mathbf{u}$ and $\mathbf{v}$ with normal parallel to $\mathbf{v} - \mathbf{u}$.

Convexity of planes can be easily seen from (4.5). *Halfspaces* are obtained from (4.5) by replacing equality by $>, <, \leq$, or $\geq$. They are extensively used in clipping and bounding volumes are often formed from their intersections. A *discrete oriented polytope* with k faces (kDOP) is the intersection of k halfspaces. The region bounded by two planes with the same normal is given by $\alpha_1 \leq \mathbf{x} \cdot \mathbf{n} \leq \alpha_2$. These so-called *slabs* are also used in bounding volumes.

It is often useful to orient planes so that they have a front and a back side. We say that a point u is *in front of* the plane, if $\mathbf{u} \cdot \mathbf{n} \geq \alpha$, and *behind* the plane, if $\mathbf{u} \cdot \mathbf{n} \leq \alpha$. The normal $\mathbf{n}$ points towards the front of the plane. The same notion of orientation applies to triangles. The orientation of a triangle **uvw** is given by the direction of the normal vector $(\mathbf{v} - \mathbf{u}) \times (\mathbf{w} - \mathbf{u})$. Triangles **uvw**, **vwu**, and **wuv** then have the same orientation, which is opposite to the orientation of **uwv**, **wvu**, and **vuw**. When looking at the triangle from the front, the vertices are specified in counter-clockwise order. Orientation is useful, for example when objects are formed from closed triangle meshes where the triangles are oriented so that forward points outside. Then a triangle facing away from the viewer can not be visible, if the viewer is outside the object.

Polygons are quite difficult to tackle formally, so we only mention them in passing. A polygon is a planar region bounded by a simple closed curve consisting of a finite number of line segments. The segment end points are vertices of the polygon. Every polygon has a triangulation: A polygon of n vertices is the union of $n - 2$ triangles with vertices from the polygon and that only intersect at edges.

4.2.2 Intersections and distances

Most typical computations in computer graphics are distances and intersections between simple geometric objects or objects formed from them. Here we illustrate some of the computations, but this merely scrapes the surface. Möller & Haines (1999) and Eberly (2001) give this topic a more comprehensive treatment. Most of these calculations are straightforward applications of elementary linear algebra and calculus.

First, we find the distance between a point $\mathbf{u}$ and the line through the origin and $\mathbf{v}$. The point $\alpha\mathbf{v}$ closest to $\mathbf{u}$ minimizes

$$d^2(\mathbf{u}, \alpha\mathbf{v}) = \|\mathbf{u}\|^2 - 2\alpha(\mathbf{u} \cdot \mathbf{v}) + \alpha^2\|\mathbf{v}\|^2.$$

This happens when $\alpha = (\mathbf{u} \cdot \mathbf{v})\|\mathbf{v}\|^{-2}$, and $(\|\mathbf{u}\|^2 - \|\mathbf{v}\|^{-2}(\mathbf{u} \cdot \mathbf{v}))^{1/2}$ is the shortest distance. In linear algebra the point

$$\mathbf{u}' = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{v}\|^2}\mathbf{v}$$

is the *orthogonal projection* of $\mathbf{u}$ to the linear subspace spanned by $\mathbf{v}$. The remainder $\mathbf{u} - \mathbf{u}'$ is the orthogonal projection to the orthogonal complement $\mathbf{v}^\perp$ of $\mathbf{v}$. It minimizes the distance from $\mathbf{u}$ to the plane $\mathbf{x} \cdot \mathbf{v} = 0$. These operations are linear and have the matrix representations $\|\mathbf{v}\|^{-2}\mathbf{v}\mathbf{v}^T$ and $\mathbf{I} - \|\mathbf{v}\|^{-2}\mathbf{v}\mathbf{v}^T$. If $\mathbf{v}$ is one of the standard basis vectors, then the projections simply zero one or two coordinates of $\mathbf{u}$. Orthogonal projections to lines and planes are important building blocks for many algorithms. The analogous problems with planes and lines in general position can be solved by adding appropriate translations.

Next, consider finding the intersection of a line and a plane in normal form. Suppose that the line is through $\mathbf{u}_0$ and $\mathbf{u}_1$ and the plane is given in implicit form by $\pi : \mathbf{v} \cdot \mathbf{n} = \alpha$. Substituting the line's parametric form into the normal form of π we get

$$(\mathbf{u}_0 + \beta(\mathbf{u}_1 - \mathbf{u}_0)) \cdot \mathbf{n} = \alpha,$$

so that

$$\beta = \frac{\alpha - \mathbf{u}_0 \cdot \mathbf{n}}{(\mathbf{u}_1 - \mathbf{u}_0) \cdot \mathbf{n}},$$

with β undefined when $\mathbf{u}_1 - \mathbf{u}_0$ is parallel to the plane. The same result also applies to a segment $\mathbf{u}_0\mathbf{u}_1$, which intersects π at a single point if and only if $0 \leq \beta \leq 1$. This only occurs when $\mathbf{u}_0$ and $\mathbf{u}_1$ are on different sides of π. In that case, the point of intersection is

$$\mathbf{u}_0 + \frac{\alpha - \mathbf{u}_0 \cdot \mathbf{n}}{(\mathbf{u}_1 - \mathbf{u}_0) \cdot \mathbf{n}}(\mathbf{u}_1 - \mathbf{u}_0).$$

Blinn and Newell (1978) deal with computations in homogeneous coordinates. There difficulties may arise from misconceptions as the right representation of a line segment is not a linearly interpolated segment between the homogeneous points when the last coordinates differ in sign, but has to go through infinity. Correct computations result from returning to first to regular coordinates and then moving the result back to homogeneous ones.

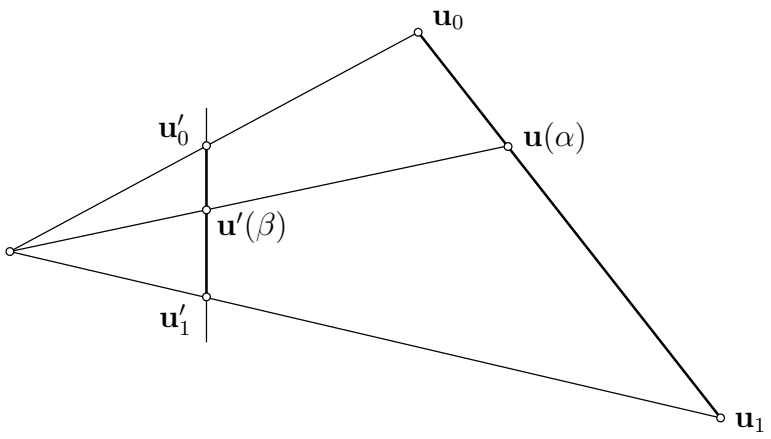


Figure 4.3. Relationship between parametric representations under perspective projection.

4.2.3 Perspective projection

One good point about triangles is that their images are still triangles under perspective projection. The correspondence of parameters in the standard parametric form is nonlinear, however. To see these facts, consider the line segment from $\mathbf{u}_0 = (x_0, z_0)$ to $\mathbf{u}_1 = (x_1, z_1)$, in parametric form

$$\mathbf{u}(\alpha) = \mathbf{u}_0 + \alpha(\mathbf{u}_1 - \mathbf{u}_0) \qquad (0 \leq \alpha \leq 1),$$

and its projected image on $z = 1$, given by

$$\mathbf{u}'(\alpha) = (x'(\alpha), 1) = (x(\alpha)/z(\alpha), 1).$$

It suffices to show that there is a one-to-one correspondence between the parameter β of the parametric form of $\mathbf{u}'_0\mathbf{u}'_1$ and α (Figure 4.3). We have

$$\begin{aligned} x'(\alpha) &= \frac{x_0 + \alpha(x_1 - x_0)}{z_0 + \alpha(z_1 - z_0)} = \frac{x_0}{z_0} + \left(\frac{\alpha z_1}{z_0 + \alpha(z_1 - z_0)}\right)\left(\frac{x_1}{z_1} - \frac{x_0}{z_0}\right) \\ &= x'_0 + f(\alpha; z_0, z_1)(x'_1 - x'_0), \end{aligned}$$

where f is the required correspondence. Evidently f maps $[0, 1]$ to itself for any $z_0, z_1 > 0$; the image of segment $\mathbf{uv}$ is the segment $\mathbf{u'v'}$. The inverse of f with is useful when mapping back from the view plane. The parameter α corresponding to the parameter β of the image line segment is given by

$$f^{-1}(\beta; z_0, z_1) = \frac{\beta z_0}{z_1 + \beta(z_0 - z_1)}. \tag{4.6}$$

The inverse mapping can also be expressed as a ratio of two linear interpolations

$$x(\beta) = \frac{\dfrac{x_0}{z_0} + \beta\left(\dfrac{x_1}{z_1} - \dfrac{x_0}{z_0}\right)}{\dfrac{1}{z_0} + \beta\left(\dfrac{1}{z_1} - \dfrac{1}{z_0}\right)}. \tag{4.7}$$

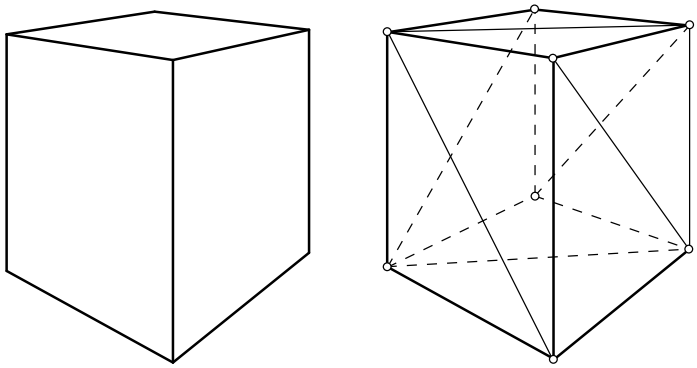


Figure 4.4. A cube and one possible triangle mesh for its representation

This result was established in (Heckbert & Moreton 1991) with a more complex derivation. It is also discussed in (Blinn 1996, ch. 17). The above correspondence is also valid for view planes at different distances. This result will be useful in rasterization.

4.3 Object and scene specification

To generate a picture of a scene, the objects in the scene must first be described. There are many possible choices for representations, but real-time computer graphics mostly deals with objects that are made out of polygons or triangles. An object's geometry is specified by giving its surface as a collection of triangles (Figure 4.4). Reasonable curved surfaces can be approximated arbitrarily well with enough triangles, so this restriction is not as severe as it may appear to be. Objects can also be logically represented as curved surfaces, such as Bézier patches, and converted to triangles before rendering. The invisible interior structures of objects are usually omitted and the appearance is specified on the surface.

The surface geometry is only part of the appearance. An object's appearance comes from the light emitted or reflected from its surfaces. Let us model this light reflection with a *shader function* f that given some parameters, possibly different at each point of a surface, can compute the light reflected from the surface in any given direction. In general, f would require the incoming light from each direction and some surface properties. Kajiya (1986) gives a general form of the required function in the form of an integral equation, known as the *rendering equation*. In his notation

$$I(x, x') = g(x, x') \left[\epsilon(x, x') + \int_S \rho(x, x', x'') I(x', x'') \, \mathrm{d}x'' \right], \tag{4.8}$$

where $I(x, x')$ is the intensity of light from x' to x, g is a geometry term, ϵ is emitted light and ρ is light scattered from x'' to x via x'. The integration is over all surface points in the scene. The rendering equation (4.8) has been mostly applied to describe global illumination algorithms as different approximations to its solution.

Using the rendering equation is infeasible in practice, so we limit the shader f to a small number of parameters. Furthermore, we require that these parameters are

specified at triangle vertices and extended by linear interpolation to other points of the triangle. If parameter values at the vertices of $\mathbf{u}_1\mathbf{u}_2\mathbf{u}_3$ are $\mathbf{x}_1, \mathbf{x}_2$, and $\mathbf{x}_3$, respectively, then at point

$$\alpha\mathbf{u}_1 + \beta\mathbf{u}_2 + \gamma\mathbf{u}_3$$

of the triangle, the color is given by

$$f(\alpha\mathbf{x}_1 + \beta\mathbf{x}_2 + \gamma\mathbf{x}_3).$$

Before about the year 2000, hardware implementations mostly had a few alternative shaders with fixed functions to choose from, but now the shaders are programmable using a special purpose *shading language*. Practical implementations place limitations on the shader function. For example, the number of parameters is limited and the shader has to be formed from a given set of functions.

Here we illustrate some possible shaders. The simplest shader sets f constant to get constant color. Generalizations of Phong's (1975) empirical model of light reflection are commonly used in realtime rendering. The model divides light reflected from a surface into ambient, diffuse, and specular components. The ambient component models the overall light level in the environment, the diffuse component light reflected evenly in all directions according to Lambert's law (matte surfaces) and the specular component imperfect mirror reflection. Material properties are defined as a mixture of these components with coefficients m_a, m_d, m_s, and specular exponent m_α (shininess). The material approaches a perfect mirror as $m_\alpha \to \infty$.

Suppose that the unit direction vector of incoming light is $\mathbf{l}$ and $\mathbf{n}$ is the unit normal of the surface. The mirror reflection direction is then $\mathbf{r} = \mathbf{l} - 2(\mathbf{l} \cdot \mathbf{n})\mathbf{n}$. According to Phong's model, the fractions f_d and f_s of specular and diffuse light reflected in direction $\mathbf{e}$ are

$$f_d = (-\mathbf{n} \cdot \mathbf{l})m_d, \qquad f_s = (\mathbf{r} \cdot \mathbf{e})^{m_\alpha} m_s,$$

where it is assumed that both the viewer and the light are in front of the surface. Thus, the observed intensity at $\mathbf{e}$ is

$$I_a m_a + I(f_d + f_s)$$

where I_a is the scene ambient light intensity and I is the intensity of the light source. A shader for Phong's model would take as parameters the position vector of the surface, the light, surface normals, and the view point, and the material properties as constants. The viewpoint is often set to the origin. The surface normal specified at triangle vertices for Phong shading is not the triangle's normal, but the normal of a surface the triangle is approximating. The surface normals given at triangle vertices will be linearly interpolated and have to be normalized before use. This also causes some artifacts because the normal is only approximated inside the triangle. Gouraud's lighting model is basically the same as Phong's, but without the specular component. *Phong shading* refers to the shader based on interpolating normals, as described above, and *Gouraud shading* evaluates the lighting model only at the triangle vertices and interpolates the intensity over the triangle. The two general approaches are called per-pixel lighting and vertex-based

lighting. Generalizations of Phong's model to colored, spot, and directional lights and inclusion of distance attenuation are straightforward. There are also more advanced light reflection models, such as that of Cook & Torrance (1982), but they are rarely used realtime graphics as other aspects are considered more important.

A simple computed function can not describe the complex textures of surfaces. Complex functions on triangles can be specified using lookup tables of the desired function or its parts in shaders. For example, texture mapping (Blinn & Newell 1976) uses an image as the table and bump mapping uses a table of surface normal directions in the lighting calculations.

4.4 A simple graphics pipeline

We now have a method to describe the geometry of objects and how they should look. The remaining task, *rendering*, is to draw an image of the scene on some display. Modern realtime rendering is hardware accelerated and based on a pipelined architecture. We will develop a simple model of the pipeline. Our treatment will be quite terse, but it differs from the norm in that more emphasis is placed on motivating the algorithms. For a more verbose discussion, see Möller & Haines (1999).

The goal is to take a scene specification in terms of triangles with the specification of viewing parameters and to produce an image defined by the shaders on a display device. Image in this context is a set of point samples evaluated on rays starting from the view point and through a regular grid (raster) on the view plane window. The grid on the view plane window corresponds to picture elements on a display device. Thus the viewing specification basically forms a correspondence between pixels and rays from the viewpoint through the view plane window (Figure 4.5). Raycasting is a straightforward solution, but it requires that the whole scene is known. The usual solution processes a triangle at a time and draws on top of previous triangles if they are behind the present one.

4.4.1 Model, world, and view space

In the following we use x, y, and z to refer to coordinates and indicate different coordinate systems with a subscript. First, the individual objects have to be specified. It is often convenient to arrange things so that every object has its own model space. The position of the object in its model space is chosen to be convenient. For example, a coordinate axis may be set to coincide with a symmetric object's axis of symmetry.

The entire scene consist of the individual objects and their spatial relationships. These relationships are described by placing the objects in the same space, called the world (or the universe). The modelling transformation takes the model from the modeling coordinates to world coordinates. The same model can be replicated in different places with different modelling transformations.

Next, the viewer's and view plane's position in the world have to be specified. A simple way to do this is to specify the viewer's position $\mathbf{u}$, forward and up directions $\mathbf{v}$ and $\mathbf{w}$ and the distance of the view plane in the forward direction, n. We are

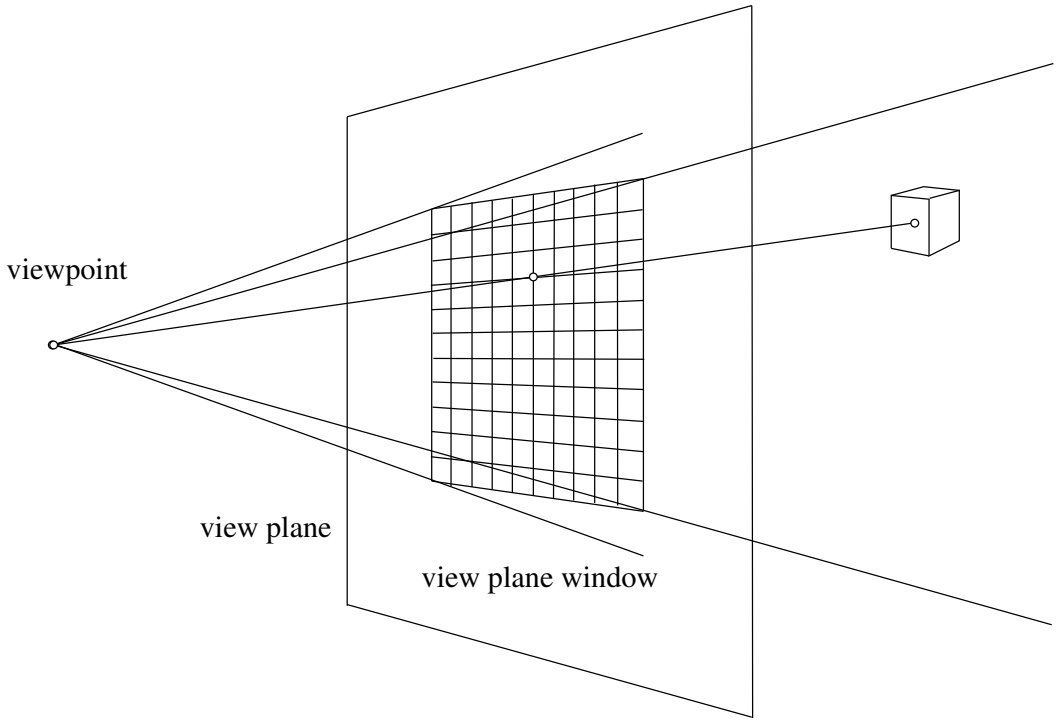


Figure 4.5. The rendering problem: point sampling through a regular grid on the view plane window.

ultimately interested in finding the surface points that fall in the viewer's field of view. This problem is easier to describe in a canonical coordinate system called *view space*. Let $\mathbf{x} = (x_w, y_w, z_w)^T$ and

$$\mathbf{R} = \begin{bmatrix} (\mathbf{w} \times -\mathbf{v}) & \mathbf{w} & -\mathbf{v} \end{bmatrix},$$

then the transformation

$$\mathbf{x}' = \mathbf{R}^T(\mathbf{x} - \mathbf{u}), \tag{4.9}$$

where $\mathbf{x}' = (x_v, y_v, z_v)^T$, takes points from world coordinates to view coordinates. Backward is mapped to z, right to x, and up to y. The sign of $\mathbf{v}$ was chosen negative to obtain the customary relation of x and y on the view plane after the mapping along with a right-handed[1] coordinate system. Quite often the model to world and world to view transformations are composed to a single transformation called the model to view transformation.

4.4.2 Screen space and clipping

Now we have the model defined in a canonical way in relation to the viewer. The next step is to find out which surface points are visible through the view plane window. Everything outside the infinite pyramid from the viewpoint through the view plane window is not. These triangles or their parts are removed in a process

[1]A coordinate frame in $\mathbb{R}^3$ is right-handed if the basis vectors satisfy $b_1 \times b_2 = b_3$ and left-handed if $b_1 \times b_2 = -b_3$. This is just a convention; usually whether a transformation preserves handedness is important.

called clipping. Suppose that the view plane window is $2w$ wide, $2h$ tall, at distance n in front of the viewer, and centered on the z-axis. A point is in the visible region, if its projection is inside the view plane window. That is, if

$$-w \leq n\frac{x_v}{-z_v} \leq w, \quad -h \leq n\frac{y_v}{-z_v} \leq h$$

or equivalently

$$\left|\frac{nx_v}{wz_v}\right| \leq 1, \quad \left|\frac{ny_v}{hz_v}\right| \leq 1.$$

Of course, we also require that visible points are in front of the viewer, $z_v < 0$. For reasons to be discussed shortly, there are also two clipping planes in the depth direction, called the near and far plane. The visible region is

$$\left|\frac{nx_v}{wz_v}\right| \leq 1, \quad \left|\frac{ny_v}{hz_v}\right| \leq 1, \quad n \leq -z_v \leq f.$$

Note that z decreases in the viewing direction. This motivates the mapping

$$x_s = \frac{n}{w}x_v, \qquad y_s = \frac{n}{h}y_v, \qquad z_s = -z \tag{4.10}$$

to so-called screen space.[2] Now visible points will have x and y coordinates in the range $[-1, 1]$ after projection.

Sometimes view plane windows are not centered with respect to the z-axis. This occurs, for example, in the CAVE environment (Cruz-Neira, Sandin, DeFanti, Kenyon & Hart 1992) where each wall functions as a window to a virtual world, and the viewer may move inside the box formed by the walls (Figure 4.6). In this case, suppose that the center of the view plane window is displaced by d from the z-axis in the x direction. Centering the view plane window using a translation gives

$$\frac{n}{-wz}x - d = \frac{n}{-wz}\left(x + \frac{wz}{n}d\right).$$

These projections can be included simply by shearing before projection.

Most triangles of a scene are usually outside visible volume, some are visible and some are only partially visible. The invisible triangles need to be removed and the partially visible triangles clipped. The borders of the visible region are formed by planes, so it suffices to consider clipping a triangle by a plane. In other words, given a plane π and a triangle **uvw**, we have to compute the portion of **uvw** in front of π. There are 4 basic cases. Two trivial cases occur, when all vertices are on one side of the plane; either the entire triangle is visible or not. In the remaining cases one or two vertices are behind the clipping plane. In this case, the result can be expressed as the union of at most 2 triangles (Figure 4.7). This type of clipper is described in (Eberly 2001, pp. 133–136).

Triangulation of the result can be deferred. Sutherland & Hodgman's (1974) re-entrant clipper that constructs the loop of edges of the polygon resulting from clipping with multiple planes. In the case of triangles the result is known to be

[2]Here z is reflected to avoid a 180° rotation in perspective projection.

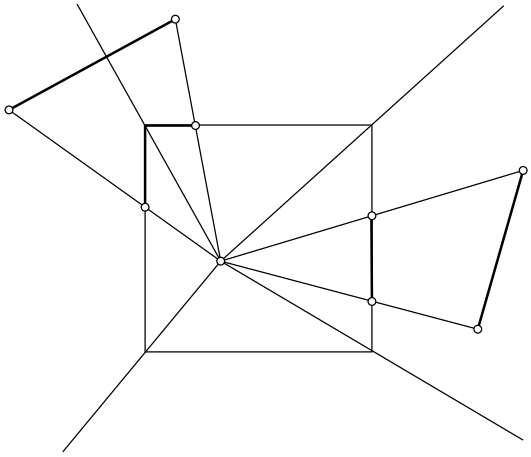


Figure 4.6. Projections in the CAVE.

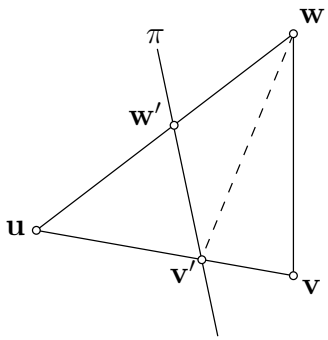


Figure 4.7. The result of clipping **uvw** with the plane π can be expressed using at most 2 triangles. In the case shown, the result is either the triangle $\mathbf{uv'w'}$ or the two triangles $\mathbf{v'vw}$ and $\mathbf{v'ww'}$.

convex so its triangulation is trivial; a simple triangle fan works. Algorithm 3 produces the entire result of clipping at each stage before clipping with the next plane; the original recurses immediately to the next clipping plane after outputting a vertex.

The required intersection points are quite easy to compute in screen space For example, if two triangle vertices are on different sides of the near clipping plane, then we have the intersection at

$$z_0 + \alpha(z_1 - z_0) = n \iff \alpha = \frac{n - z_0}{z_1 - z_0}.$$

The analogous solution works at the far plane. At the other clipping planes the sides are given by $|x|, |y| \leq |z|$ and $|x|, |y| \geq |z|$. For example, if two end points are on different sides of the plane $x = z$, then the intersection occurs when

$$x_0 + \alpha(x_1 - x_0) = z_0 + \alpha(z_1 - z_0), \iff \alpha = \frac{z_0 - x_0}{x_1 - x_0 - z_1 + z_0}.$$

Naturally all shader parameters have to be interpolated from the old vertices to the new vertices of clipped triangles.

Algorithm 3 Sutherland & Hodgman's (1974) clipping algorithm adapted to triangles. The variable n_i counts the number of vertices output and $\mathbf{u}_i(j)$ denotes the jth vertex output at stage i. Inside refers to the side on which to retain the triangle parts, and, with slight abuse of notation, the operation $\cap$ computes the point of intersection.

procedure SUTHERLANDHODGMANCLIP($\mathbf{uvw}$)
  $\mathbf{u}_0(1,2,3) \leftarrow \mathbf{u}, \mathbf{v}, \mathbf{w}$; $n_0 \leftarrow 3$.
  for clipping planes π_i, $i = 1, 2, \ldots, n$, **do**
    if $n_{i-1} < 3$ **then return** $\emptyset$ ▷ Entire triangle clipped.
    $n_i \leftarrow 0$.
    if $\mathbf{u}_{i-1}(1)$ is inside π_i **then**
      $\mathbf{u}_i(1) \leftarrow \mathbf{u}_{i-1}(1)$.
      $n_i \leftarrow n_i + 1$.
    for $j \leftarrow 2, 3, \ldots, n_{i-1}$ **do**
      if $\mathbf{u}_{i-1}(j-1)$ and $\mathbf{u}_{i-1}(j)$ are on different sides of π_i **then**
        $\mathbf{u}_i(n_i) \leftarrow \mathbf{u}_{i-1}(j-1)\mathbf{u}_{i-1}(j) \cap \pi_i$.
        $n_i \leftarrow n_i + 1$.
      if $\mathbf{u}_{i-1}(j)$ is inside π_i **then**
        $\mathbf{u}_i(n_i) \leftarrow \mathbf{u}_{i-1}(j)$.
        $n_i \leftarrow n_i + 1$.
    if $\mathbf{u}_{i-1}(1)$ and $\mathbf{u}_{i-1}(n_{i-1})$ are on different sides of π_i **then**
      $\mathbf{u}_i(n_i) \leftarrow \mathbf{u}_{i-1}(1)\mathbf{u}_{i-1}(n_{i-1}) \cap \pi_i$;
      $n_i \leftarrow n_i + 1$.
  return $\{\mathbf{u}_n(1)\mathbf{u}_n(i)\mathbf{u}_n(i+1) : i = 2, \ldots, n_n - 1\}$. ▷ Triangle fan.

4.4.3 Rasterization

The final stage of the pipeline is rasterization, where the actual picture is formed. The raster is formed from a regular grid of pixels at integer coordinates $x_d = 0, 1, \ldots, w_d - 1$ and $y_d = 0, 1, \ldots, h_d - 1$, where the width and height depend on device resolution. The input of this stage is a triangle that is known to be completely visible, but some triangles may still be in front of others. The classical z-buffer algorithm solves this problem.

The screen space has be to mapped to device coordinates to find the pixels that a triangle covers. First we perform the perspective division of x_s and y_s by z_s and then map the range $[-1, 1]$ to the range of pixels. The device coordinates are

$$x_d = w_d \frac{x_s/z_s + 1}{2} - 1/2, \qquad y_d = h_d \frac{y_s/z_s + 1}{2} - 1/2. \tag{4.11}$$

The shift by $1/2$ centers the pixel lattice on the view plane window, otherwise some pixels would lie on the edge. Rounding may cause some vertices on the clipping boundaries to round off the screen, so we should shrink the scale slightly in this case (Blinn 1996, ch. 14). We have assumed that the origin is at the bottom left corner with x increasing right and y up. Due to historical reasons, the device origin is usually at the top left corner with x right and y down.

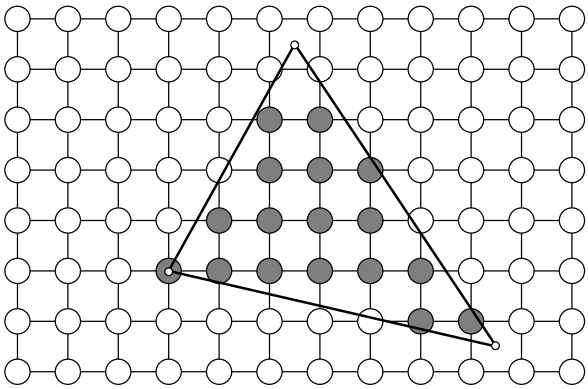


Figure 4.8. Triangle rasterization.

A point is in front of another if its z-coordinate is larger in view space. Thus, if we attach the value of z of the corresponding surface point to each pixel then this solves the problem incrementally for the whole scene: We can simply draw the surface point only if the previously drawn point is further away. Perspective interpolation of z according to (4.7) gives

$$z(\beta) = \frac{1}{\dfrac{1}{z_0} + \beta\left(\dfrac{1}{z_1} - \dfrac{1}{z_0}\right)}.$$

This shows that we can use linear interpolation of z^{-1} on segments on the device coordinates instead of perspective interpolation of z to determine visibility.[3]

There is a small problem with numerical precision as z^{-1} can be arbitrarily large and we have to map it to a finite range of values. This is the reason for the near and far clipping planes at $z = -n$ and $z = -f$. They truncate the infinite visible pyramid to a pyramid with the top cut off, known as a frustum. For convenience, we transform z^{-1} to map $[-f, -n]$ to $[0, 1]$

$$d = \left(\frac{1}{-n} - \frac{1}{z}\right)\left(\frac{1}{-n} - \frac{1}{-f}\right)^{-1} = \frac{f(z+n)}{z(f-n)}. \tag{4.12}$$

In practical implementations d can be mapped to $\lfloor 2^b d \rfloor$ for a precision of b bits. The coordinate d is known as depth.

Rasterization, or scan-conversion, of a triangle **uvw** fills all pixels inside the triangle (Figure 4.8). One strategy is to fill all pixels falling between left and right edges proceeding from top to bottom. Triangles can be clipped into parts that share edges and they can also share edges and vertices as parts of triangle meshes. In such cases we need a rule that assigns a pixel hit by an edge to only one of the triangles sharing it. One alternative is to use half-open intervals. To fill $[a, b)$, we need all integers in it. The first integer in $[a, b)$ is $\lceil a \rceil$. The number of integers on the interval is

$$\begin{cases} \lfloor b \rfloor - \lceil a \rceil, & \text{if } b \text{ integral,} \\ \lfloor b \rfloor - \lceil a \rceil + 1, & \text{otherwise.} \end{cases}$$

[3]Obviously, $z_0 < z_1$ iff $z_0^{-1} > z_1^{-1}$.

In fixed point arithmetic, one practical solution is to use $\lfloor b - \epsilon \rfloor$ instead of $\lfloor b \rfloor$, where ϵ is the smallest representable number. This removes to need to check for an integral value. In any case, the implementation must guarantee that this leaves no gaps in the discretized intervals.

First we must find the topmost vertex. The remaining vertices need to be sorted so that the triangle is in counter-clockwise order. Suppose that **uvw** is in this order with **u** topmost and that **v** is above **w**. We can divide **uvw** into two parts with a horizontal segment from **v** to **uw**. This leaves the filling of pixels between **uv** and **uw** and **vw** and **uw**. In general, the vertices are between pixels. We need to intersect the edges with the scanlines to find the pixels to fill. The intersection points are quite easy to compute, however. The segment **uv**, where **u** is topmost, intersects all horizontal lines at integral coordinates between $\lfloor u_2 \rfloor$ and $\lceil v_2 \rceil$. Intersections with scanlines lines occur at parameter values

$$\alpha_i = \frac{u_2 - \lfloor u_2 \rfloor + i}{u_2 - v_2}, \qquad (i = 0, 1, \ldots, \lfloor u_2 \rfloor - \lceil v_2 \rceil). \tag{4.13}$$

Substituting this back into the parametric form gives the points of intersection while walking down the left and right egdes of the triangles. The shader parameters divided by z as in (4.7), $1/z$, and depth are linearly interpolated between the vertices. On each scanline intersection, we walk the pixels from left to right edges with linear interpolation analogous to (4.13) between the interpolated values at edges. On each pixel we test if the interpolated depth is smaller than at that pixels and if it is, call the shader with (4.7) and fill the pixel, otherwise continue to the next pixel.

4.4.4 Summary

The pipeline presented here is only a simple conceptual outline. It fits in less than 500 lines of C code. In practice, the systems implemented by modern graphics application program interfaces (API) are much more complex. For example, the OpenGL sample implementation released by SGI in the year 2000 contains about 18000 lines of C code, and it was released before programmable shaders and the shading language were included in the specification.

In addition to the shader function, usually referred to as a fragment or pixel shader, the APIs also include programmable processing of vertex data. This can be useful for animation, for example. They generalize the notion of pixel to include additional data, similar to the depth value used in the z-buffer algorithm. For example, the stencil buffer contains a certain number of bits for each pixel. The bits can be used to count the number of times a pixel has been written, which can be used to implement shadows or to mask out portion of the raster while rendering. Translucent primitives can be drawn by alpha blending. Signal processing operations, such as blurring using a lowpass filter, and working with multiple images at the same time are possible. In fact, the computational models are so general that 3D hardware is now being used to accelerate scientific computation (Buck, Foley, Horn, Sugerman, Fatahalian, Houston & Hanrahan 2004)!

There are at present two dominant graphics APIs, Direct3D and OpenGL. They are quite similar to each other. The programs in the present thesis were imple-

mented using OpenGL because of its portability. OpenGL is treated in detail in the specification (Segal & Akeley 2003).

4.5 Application level

The graphics pipeline is just one part of an application using graphics. It guarantees to produce the correct picture for every scene, but can be quite inefficient in doing so, if the application feeding the data into the pipeline is badly written. For example, in objects specified by triangle meshes, the triangles share vertices, and the shared vertices are specified many times as parts of different triangles, but a vertex does not have to go twice through the transformation from model space to screen space. Graphics APIs usually provide the user with optimization opportunities for transferring data. In the case of shared vertices between triangles, the models can be defined using strips of triangles or triangle fans. (Möller & Haines 1999, pp. 231–240). A few of the most recently used vertices can also be cached.

The z-buffer algorithm can handle an arbitrary number of triangles correctly, but it is inefficient to solve the visibility problem at the pixel level in image precision. Drawing the triangles that will be completely occluded is not necessary. Clipping is an inefficient way to remove individual triangles outside the view frustum. A considerable amount of research has been done on accelerating visibility processing. Culling in general refers to removing objects that will not be visible. Möller & Haines (1999, pp. 191–217) review these techniques. For example, view frustum culling is a process that removes entire objects that are outside the view frustum. It requires some auxiliary data structures such as bounding volume trees or at least simple bounding volumes for objects. Bounding volumes are simple geometric objects that are used to envelop complex objects; spheres and boxes are popular. If a bounding volume is outside the view frustum, then the contained objects are also outside. Occlusion culling is a process that tries to remove objects that are behind other objects when looking from the view point.

4.6 Animation

An observer perceives motion when images are presented quickly in succession and consecutive images are sufficiently similar. Realtime computer graphics usually uses *double buffering* for animation. One finished picture is visible while another is being drawn. When the next picture is ready, the pictures are swapped. The swapping is usually done when the display starts to refresh. Figure 4.9 illustrates the process. The frequency at which new images are generated and shown has to be at least 25 Hz, preferably higher. Virtual reality systems commonly use 50–60 Hz.

Next, we discuss animation of rigid bodies by specifying their movement paths as a function time. These paths are one-parameter families of transformations that include the orientation and the position of an object. During rendering, the transformation corresponding to the appropriate time is picked from the family and applied to the object. There are of course other problems in animation, but for the purposes of this thesis, rigid body animation suffices. These techniques were used

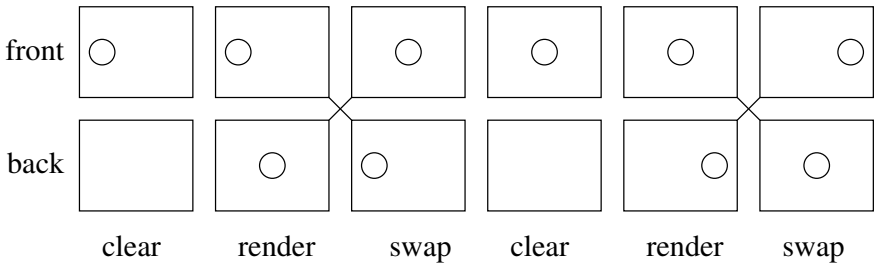


Figure 4.9. Double buffering shows one image, the front buffer, while the next is being rendered in the back buffer, and swaps the images when the rendering is ready.

to implement parametric movement, such as moving in an otherwise static scene in the virtual environments.

4.6.1 Kinematics

Kinematics, the mathematical study of motion, is usually not treated in detail in computer graphics texts. This is somewhat surprising because much of computer animation is applied kinematics. Strictly speaking kinematics is a subfield of mechanics concerning motion; forces and masses are not considered (Bottema & Roth 1990, Preface). It has applications, for example, to designing mechanisms to produce given motions in robotics. A related field, dynamics, considers also the forces involved.

The starting point of kinematics is the Euclidean displacement.[4] Consider the motion of a rigid object. The natural requirements are that during motion the shape of the object does not change and that there is a way to move the object continuously from the starting position to the final position. The first requirement is that the displacement is an isometry. However, isometries also include reflections, hence the second requirement. One formal way to include it is to require the existence of a continuous sequence of isometries from the identity transformation to the final position. The latter requirement rules out reflections as there is no way to continuously reflect an object. The Euclidean displacements form a group and it is easy to see that translations are Euclidean displacements. The factorization

$$T(\mathbf{u}) = \underbrace{T(\mathbf{u}) - T(\mathbf{0})}_{\mathbf{R}} + T(\mathbf{0})$$

of a displacement is fundamental. It factors a displacement into a translation and a rotation $\mathbf{R}$. When discussing an object's state in space, *orientation* refers to $\mathbf{R}$, and *position* to $T(\mathbf{0})$. A (1-degree of freedom) *motion* $T(\mathbf{u}, t)$ is a one-parameter family of displacements

$$T(\mathbf{u}, t) = \mathbf{R}(t)\mathbf{u} + \mathbf{v}(t),$$

where we require that T is continuous with respect to t for all $\mathbf{u}$. The standard approach to specify motion is *keyframe interpolation*. We specify $T(\mathbf{u}, t)$ at certain

[4]Euclidean displacements as defined above are also called *direct isometries* or *rigid motions*.

instants of time called *keyframes* and interpolate between them. The term is also applicable to animation of non-rigid objects, but we focus on the rigid case. Given a sequence

$$T_1, T_2, \ldots, T_n$$

of displacements the goal is produce a smooth motion that interpolates them. We can consider interpolation of position and orientation separately.

4.6.2 Animating position

Specification of $\mathbf{v}(t)$ is much simpler than that of $\mathbf{R}(t)$. It represents the movement of the origin and can be any parametric curve. We are generally interested in movements that are in some sense smooth. The usual way to ensure smoothness of $\mathbf{v}(t)$ is to require a certain number of continuous derivatives. A parametric curve is said to be of class $\mathcal{C}^i$ if at least the i first of its derivatives are continuous; a curve of class $\mathcal{C}^0$ is simply continuous. Another less stringent requirement is that of geometric continuity, which requires that the left and right derivatives have the same direction. The classes analogous to $\mathcal{C}^i$ are denoted by $\mathcal{G}^i$.

Perhaps the most natural way to specify translational movement is to give a list of locations and times when they are reached and simply interpolate between them. The following treatment is 1-dimensional for simplicity. It generalizes to more dimensions simply by replacing the appropriate parameters with vectors. Formally, the objective is to form a smooth function $f(x)$ that interpolates the points

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

that is $f(x_i) = y_i$, for $i = 1, 2, \ldots, n$. If the functions $g_1, g_2, \ldots, g_n$ satisfy $g_i(x_j) = 1$, if $i = j$, and $g_i(x_j) = 0$, if $i \neq j$, then obviously $f(x) = \sum y_i g_i(x)$ fulfills the requirements. Lagrange's interpolating polynomials

$$g_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

are well-known. The interpolating function is a polynomial of degree $n - 1$ and belongs to $\mathcal{C}^\infty$. Another even simpler solution to the interpolation problem is the piecewise linear curve

$$f(x) = y_i + \frac{x - x_i}{x_{i+1} - x_i}(y_{i+1} - y_i), \qquad (x_i \leq x \leq x_{i+1}), \tag{4.14}$$

which is in class $\mathcal{C}^0$.

Polynomial interpolation suffers from non-locality. Changing one point or adding control points affects the entire curve. The piecewise linear curve is local, but not smooth at the interpolated points. Splines are piecewise polynomial functions that can be smooth at interpolated points and have local control. The piecewise linear curve (4.14) is a spline of first order polynomials. The sharp angles at interpolated points can be smoothed by using higher order polynomials. If the function $f(x)$ is a spline with

$$f(x) = f_i(x) \qquad (x_i \leq x \leq x_{i+1}),$$

where $f_i(x)$ are segments of the spline, then $f \in \mathcal{C}^1$, if its segments satisfy

$$f_{i-1}(x_i) = f_i(x_i), \qquad f'_{i-1}(x_i) = f'_i(x_i).$$

Thus each segment must satisfy

$$f_i(x_i) = y_i, \quad f'_i(x_i) = y'_i, \quad f_i(x_{i+1}) = y_{i+1}, \quad f'_i(x_{i+1}) = y'_{i+1}$$

at the end points. The simplest polynomials capable of this are cubics. We introduce a new parameter $t = (x - x_i)/(x_{i+1} - x_i)$ and adjust the conditions on derivatives accordingly to simplify the situation. To construct a cubic curve with specified end points and derivatives at end points, set

$$f(t) = a_3t^3 + a_2t^2 + a_1t + a_0$$

with $f(0) = y_0, f'(0) = y'_0, f(1) = y_1, f'(1) = y'_1$. This gives a system of linear equations

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} y_0 \\ y'_0 \\ y_1 \\ y'_1 \end{bmatrix}$$

that has the solution

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y'_0 \\ y_1 \\ y'_1 \end{bmatrix}.$$

The function f can now be written in terms of the constraints

$$f(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y'_0 \\ y_1 \\ y'_1 \end{bmatrix}$$

or simply as

$$f(t) = (1 - 3t^2 + 2t^3)y_0 + (t - 2t^2 + t^3)y'_0 + (3t^2 - 2t^3)y_1 + (t^3 - t^2)y'_1.$$

The coefficient polynomials of the parameters are called the Hermite basis and the resulting splines are called Hermite curves, in honor of Charles Hermite. Values of the derivative at the points to be interpolated are still to be determined. Catmull-Rom splines set $y'_i = (y_{i+1} - y_{i-1})/2$. Kochanek and Bartels (1984) present cubic splines with slightly more control and that are easier to specify. They add 3 parameters to each interpolated point: tension τ, continuity γ, and bias β. These curves, also known as TCB-splines after the parameters, were used in implementing the virtual reality stimuli in this thesis.

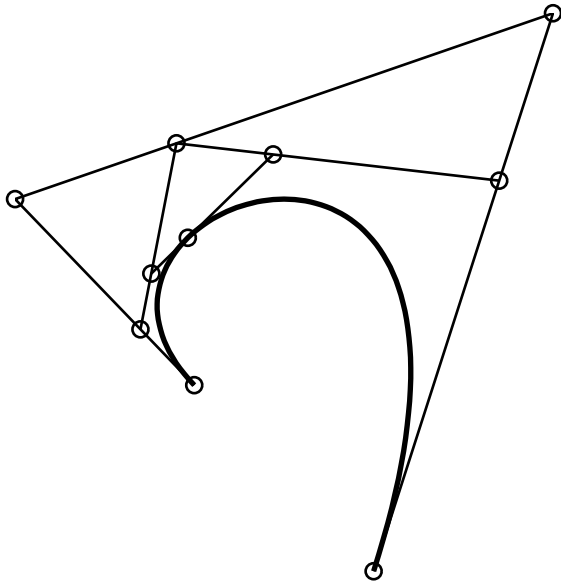


Figure 4.10. De Casteljau's construction of a cubic Bézier curve.

Bézier curves should be mentioned because of their importance in computer graphics. They are defined by iterated linear interpolation of consecutive control points with a single parameter. For example, a quadratic Bézier curve is defined by

$$\begin{aligned} f(t) &= (1-t)((1-t)y_0 + ty_1) + t((1-t)y_1 + ty_2), \\ &= (1-t)^2 y_0 + 2t(1-t)y_1 + t^2 y_2. \end{aligned}$$

where y_0 and y_2 are the end points and y_1 is a control point. This method of evaluation is called de Casteljau's construction (Figure 4.10). For $n+1$ control points, $y_0, y_1, \ldots, y_n$, the curve can be expressed as

$$f(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} y_i,$$

where coefficient polynomials are the Bernstein basis polynomials. One interesting property about these curves is that they are contained in the convex hull of the control points. This is apparent from the evaluation as iterated convex combinations in de Casteljau's construction.

The instaneous velocity of a point moving along a curve $f(t)$ is $\|f'(t)\|$ and its integral is arc-length, the distance travelled by a point moving along $f(t)$. The velocity is not constant in the case of cubic splines, which makes specification of smooth movement difficult. It is natural to think that an object follows a path with a certain velocity. Velocity can be decoupled from the movement path by *arc-length reparameterisation* of the path, where the parameter t is exchanged to a parameter s that gives the distance along the path from its beginning. The process computes s as a function of t by numerical integration, as there is no analytical solution for cubic curves, and then inverts $s(t)$ to find t for a given s during animation (Watt & Watt 1992, pp. 346–350).

4.6.3 Animating orientation

Rotations are more difficult and also more interesting than translations. We already wrote $\mathbf{R}$ for rotations to signify linearity; it is not difficult to see this. When an object rotates, it is generally considered that one point does not move. A rotation R about the origin is an Euclidean displacement with the additional property that it has the origin is a fixed point. They form a subgroup of displacements. Rotations preserve the inner product.[5] Thus, the image of an orthonormal basis of $\mathbb{R}^3$ is also an orthonormal basis for $\mathbb{R}^3$. It follows that R is linear.

Rotation matrices have the property that $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}$, because the columns are an orthonormal basis. Thus, $\mathbf{R}^{-1} = \mathbf{R}^T$. Now, since

$$1 = \det \mathbf{I} = \det \mathbf{R}^T\mathbf{R} = (\det \mathbf{R})^2$$

we have $\det \mathbf{R} = \pm 1$. However, $\det \mathbf{R} = 1$, because the determinant is continuous and $\det \mathbf{I} = 1$. Thus, the group of rotations is isomorphic to orthogonal matrices with $\det = 1$, the so-called *special orthogonal group* $SO(n)$.

Some way to define the "curve" connecting two orientations is needed to interpolate them. A simple method to do this is to parameterize rotations and interpolate the parameters. The elements of rotation matrices are not useful as such, because the matrix is required to remain orthogonal during the interpolation. To come up with a parameterization of rotations we start from $\mathbb{R}^2$. There, every unit vector is of the form $(\cos\theta, \sin\theta)$ for some θ and the other unit vector in an orthonormal basis must be $\pm(-\sin\theta, \cos\theta)$. Only the positive case is possible, because for a rotation matrix $\mathbf{R}$ we have $\det \mathbf{R} = 1$. All rotations in $\mathbb{R}^2$ are of the form

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

This gives a parameterization in the planar case. We can rotate any vector to a coordinate axis. For example, for $\mathbf{u} = (u_1, u_2)$ we can find a rotation such that

$$\mathbf{R}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \pm \begin{bmatrix} \|\mathbf{u}\| \\ 0 \end{bmatrix}.$$

Taking

$$\theta = \begin{cases} -\arctan(u_2/u_1), & u_1 > 0, \\ \pi - \arctan(u_2/u_1), & u_1 < 0, \\ -\pi/2, & u_1 = 0, u_2 \geq 0, \\ \pi/2, & u_1 = 0, u_2 < 0 \end{cases}$$

gives the positive case. We can use the parameterization of planar rotations to build a parameterization of rotations in $\mathbb{R}^3$ from rotations on coordinate planes. In

[5] As $d(\mathbf{u}, 0) = d(R(\mathbf{u}), 0)$, the norm is preserved, consider $d^2(\mathbf{u}, \mathbf{v}) = d^2(R(\mathbf{u}), R(\mathbf{v}))$.

$\mathbb{R}^3$ we have at least the rotations

$$\mathbf{R}_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \quad \mathbf{R}_2(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix},$$
$$\mathbf{R}_3(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

analogous to the planar case; the subscript gives the invariant coordinate. These are called Givens rotations. If $\mathbf{R}$ is a rotation in $\mathbb{R}^3$, we can choose θ and ψ in

$$\mathbf{R} = \mathbf{R}_3(\psi)\mathbf{R}_2(\theta)\underbrace{\mathbf{R}_2(-\theta)\mathbf{R}_3(-\psi)\mathbf{R}}_{\mathbf{R}'}.$$

so that the third column of $\mathbf{R}'$ is $\mathbf{e}_3$. Now $\mathbf{R}'$ is also a rotation as a composition of rotations. The third row vector of $\mathbf{R}'$ is also $\mathbf{e}_3$, because it has unit length, and $\mathbf{R}'_{33} = 1$. We must have $\mathbf{R}' = \mathbf{R}_3(\phi)$ for some ϕ. Thus, any rotation in $\mathbb{R}^3$ is representable as

$$\mathbf{R} = \mathbf{R}_3(\psi)\mathbf{R}_2(\theta)\mathbf{R}_3(\phi)$$

for a suitable choice of the so-called Euler angles ϕ, θ, and ψ. This parameterization is not quite unique, as there are always at least two valid choices of θ and ψ in the above construction. This form is apparently conventional. With a slight change in the above derivation we can show that there is also a similar representation of the form $\mathbf{R} = \mathbf{R}_3\mathbf{R}_1\mathbf{R}_2$, where the angles are called yaw, pitch, and roll,[6] and indeed a representation in any factorization into three Givens rotations where consecutive ones are around different axes. Sometimes the angles occurring in any such factorization are called Euler angles. We can now, in principle, linearly interpolate the Euler angles to interpolate between orientations.

Rotations in the plane can be represented by considering vectors as complex numbers. The numbers $e^{i\theta}$ correspond to planar rotations. There is a similar representation of rotations in space, namely the quaternions. Their invention by Sir William Rowan Hamilton in 1843 is considered a great moment in mathematics as the first noncommutative algebra was introduced (Eves 1990, pp. 504–508). Shoemake (1985) introduced them to computer graphics. In short, quaternions generalize complex numbers by introducing two extra imaginary units with the rules

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1.$$

It follows from this formulation that $\mathbf{ij} = \mathbf{k}$, $\mathbf{jk} = \mathbf{i}$, $\mathbf{ki} = \mathbf{j}$, $\mathbf{ji} = -\mathbf{k}$, $\mathbf{kj} = -\mathbf{i}$, and $\mathbf{ik} = -\mathbf{j}$. The conjugate $\mathbf{q}^*$ and the norm $|\mathbf{q}|$ of a quaternion $\mathbf{q} = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$ are defined

$$\mathbf{q}^* = a - b\mathbf{i} - c\mathbf{j} - d\mathbf{k}, \quad |\mathbf{q}|^2 = a^2 + b^2 + c^2 + d^2.$$

Embedding $\mathbb{R}^3$ by $\mathbf{u} \mapsto u_1\mathbf{i} + u_2\mathbf{j} + u_3\mathbf{k}$ into quaternions, the mapping $\mathbf{u} \mapsto \mathbf{quq}^*$ is a rotation, whenever $\mathbf{q}$ is a unit quaternion. The proof can be found in (Eberly 2001,

[6] Assuming the standard setup of axes. Yaw, pitch, and roll are subjectively defined. Yaw rotates horizontally, pitch vertically, and roll around the line of sight.

pp. 13–15). Quaternions are not quite isomorphic to $SO(3)$ as both $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation. The unit quaternions form a 4-dimensional sphere and the notion of linear interpolation has a simple analogue called spherical linear interpolation, slerp, on the sphere. Constant angular velocity interpolation between two unit vectors $\mathbf{u}_0$ and $\mathbf{u}_1$ along a great circle gives

$$\mathrm{slerp}(\mathbf{u}_0, \mathbf{u}_1, t) = \frac{\sin(1-t)\theta}{\sin\theta}\mathbf{u}_0 + \frac{\sin t\theta}{\sin\theta}\mathbf{u}_1,$$

where θ is the angle between $\mathbf{u}_0$ and $\mathbf{u}_1$. Shoemake (1985) replaces linear interpolation in de Casteljau's construction by spherical linear interpolation to construct smooth curves on the quaternion unit sphere. Here we have omitted the derivation of the Bèzier control points. There is a geometric construction of Catmull-Rom splines that expresses the spline value in terms of linear interpolations of 4 adjacent interpolated points. It is applied to quaternion interpolation in (Schlag 1991). This is the method used in this thesis to interpolate orientations.

4.7 Virtual reality

Virtual reality refers to an artificial immersive environment. In this thesis, the notion is limited to stereoscopic graphics generated based on measured head orientation, but often the environments are interactive. Perceptual factors play a prominent role in creating virtual environments (Kalawsky 1993, ch. 3).

Binocular disparity caused by the slightly different images in left and right eyes is one of the factors in depth perception. The closer an object is to the viewer, the greater the differences are in the images. Binocular disparity can be simulated by drawing two images from slightly different view points and showing one to the left eye and the other to the right eye (Figure 4.11). There are two basic forms of display for this purpose: Either there are two completely separate displays or two images are overlaid and the different images are filtered out for the left and right eyes. The former method is used in head-mounted displays and the latter in projected displays. Showing two different overlaid images can be accomplished by polarized light or with shutter glasses. Shutter glasses shut out one eye and alternate rapidly between eyes. This action is synchronized with the display so that the display shows one image to the left eye and another to the right eye. Polarized light tends to lose intensity, and the result can be like wearing sunglasses in a dark room. Shutter glasses suffer from flicker, and head mounted displays are cumbersome and have limited resolutions.

Looking around in the virtual environment requires head movements to be tracked, and interaction with the environment requires specialized input devices. Kalawsky (1993, ch. 4–6) surveys the hardware and software solutions for VR. It is easy to integrate the sensor data into rendering from an implementor's point of view: For example, head-orientation sensors basically give $\mathbf{R}$ in (4.9).

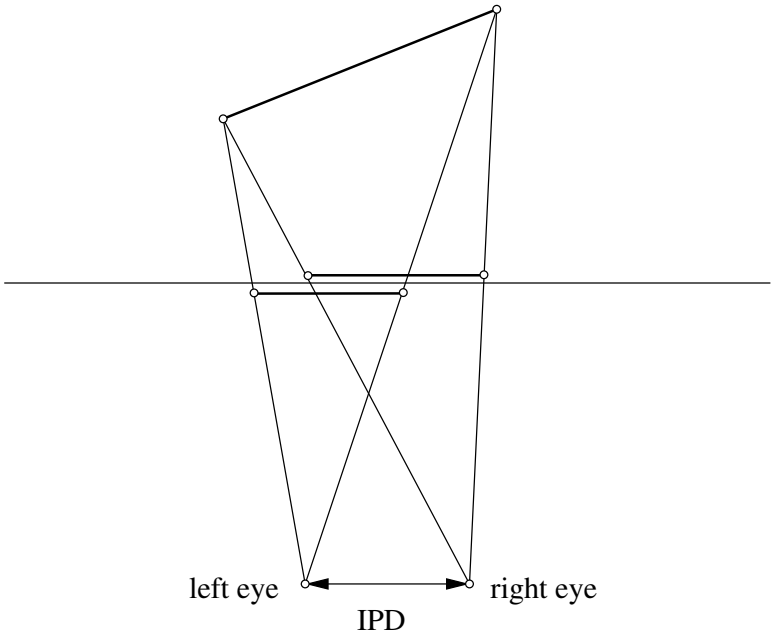


Figure 4.11. Stereoscopic projection. The interpupillary distance (IPD) separates the two centers of projection and the eyes see different images.

Chapter 5

Implementation and applications

During the research presented in Publications I–V, we conducted many experiments with virtual reality and posturography, and our measurement system underwent iterative improvements. In the first version, one computer showed VR graphics on a head-mounted display (HMD), another computer collected the measurements, and the two were synchronized manually. Later, the system evolved to an integrated package designed for laboratory use by laboratory assistants. This chapter begins with a detailed description of the most advanced version and then presents the applications with various versions along with comments on the system's evolution over time.

5.1 Implementation

The measurement system, as used in Publications IV and V, is deployed it in three laboratories: the Virtual Reality Laboratory at the University of Tampere, the Hearing Center of Tampere University Hospital (HC), and at the Finnish Institute of Occupational Health (FIOH) in Helsinki. The laboratories have different hardware configurations: The HC and FIOH installations include the moving platform, and the FIOH installation uses a stereoscopic projection display implemented with polarized light. Figure 5.1 shows the structure of the system.

The main components of the measurement system are a force platform, an HMD (Virtual Research V8) or another VR display device, and two personal computers. One of the computers (client) runs the operator's user interface, and the other computer (server) generates the virtual environment; it is equipped with a 3D graphics accelerator (nVidia GeForce series) for this purpose. The computers communicate over an Ethernet network. The server receives measurements of head orientation from a head orientation tracker (InterSense InterTraxx or IS-300 InertiaCube) mounted on the HMD, and uses them when rendering the graphics. The tracker only measures head orientation, which suffices for looking around in the virtual environment by rotating the head, but not for larger movements.

Force transducers (Hottinger Baldwin PWSM-100kg) beneath the force platform measure forces. These measurements go through an amplifier, and an analog-digital converter (ADC, Hewlett-Packard HP-34970A or Data Translation 9800 series) subsequently samples them with a sampling rate of 50 Hz and a resolution of 16 bits.

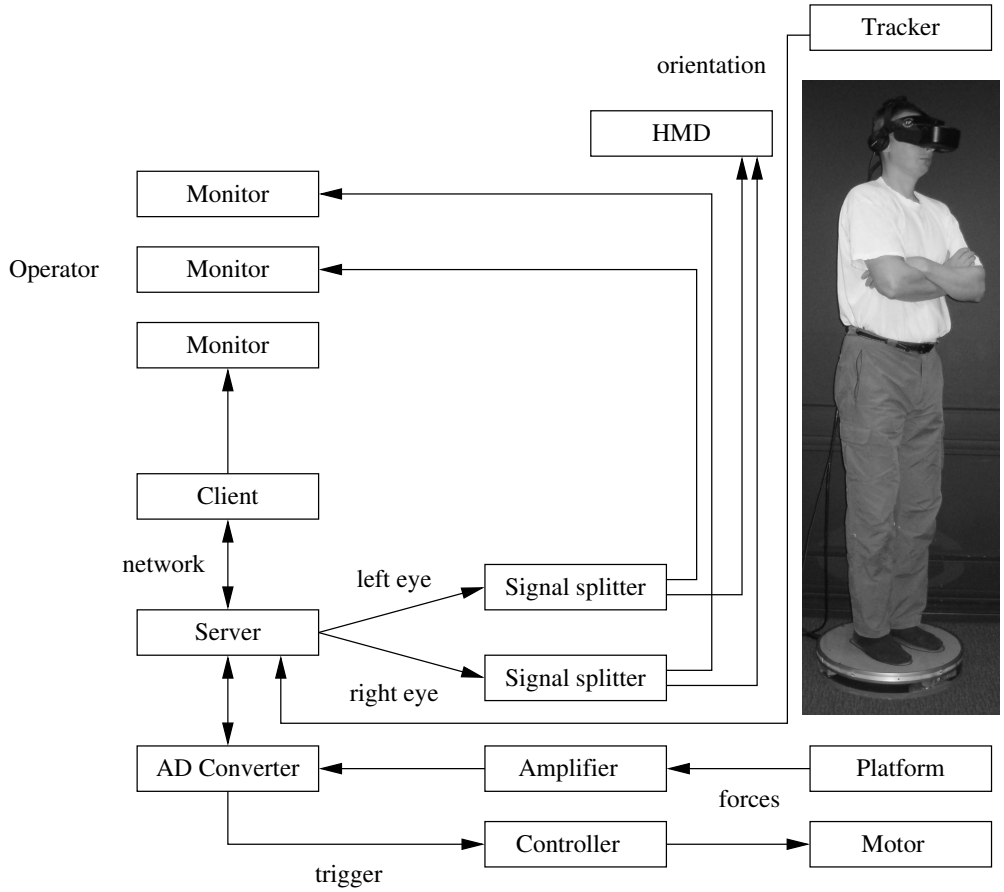


Figure 5.1. VR posturography system: components and interconnections.

The server receives the sampled measurements from the ADC over an USB or RS-232 serial connection. It also uses the ADC to signal a controller (API Motion DS-3042i) to start a preprogrammed movement sequence, when the programmable tilting platform is in use. A motor-driven screw-slider system accomplishes the tilting. The design of the mechanism is such that the movement of the platform is essentially unaffected by a person standing on it. The platform is quite accurate in the static case when a weight is placed on it. For example, when we carried out one calibration of a static platform with a 60 kg calibrated weight, the standard deviation of measurements was about 0.0005 V on each of the three channels. Most of the noise is from the environment, as the platform was in an office building and was not shielded from mechanical vibration. After calibration using Algorithm 1, the root mean square (RMS) error of weight was 0.010 kg on the unfiltered calibration measurements. The tilting platform has some more noise, because the motors keep the platform in place. In a calibration with a 40 kg weight, the standard deviation of measurements was about 0.0013 V and the RMS error of weight was 0.056 kg.

The operator uses the client computer to control the server. The interface is simple: It can only start and stop trials and save the measurements. There are safeguards against saving over previous measurements and forgetting to save them. After each measurement, a display of the measured stabilograms and some basic parameters is shown, and the operator can verify that the data acquisition works

properly. She can also see the virtual environment on monitors during a trial; the feature is implemented with signal splitters. We use MATLAB to analyze the measurements after taking them.

The software implementation is modular and mostly portable. We developed it mainly under Linux and used the GNU toolchain to build the executables. MinGW was used to build the software under Microsoft Windows. Our laboratories run it on Microsoft Windows XP, but the software also works on Unix-like platforms such as Linux. Cross-platform libraries ensured portability; the Simple DirectMedia Layer (SDL), OpenGL, and GTKmm were used. SDL provides portable graphics initialization and event handling, OpenGL is for 3D rendering, and GTKmm is a cross-platform GUI toolkit that we used to implement the operator's user interface. The implementation in the C++ programming language takes about 20000 lines of code. Of course, much more code was written and thrown away during the development, because the requirements changed from one experiment to another.

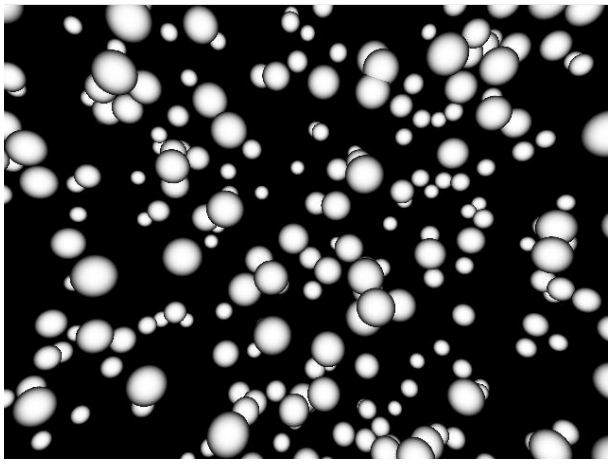
We saw the addition of new VR stimuli and measurement devices as the most likely changes to the software, and implemented a plugin architecture for this purpose. The virtual environments and the software modules for measurement devices are implemented as dynamic link libraries (DLL) loaded at run-time, and the server software implements a simple API for them. Figures 5.2 and 5.3 contain a few of our virtual environments—some of them are not used in published research. The client computer controls the server via a simple text protocol over TCP/IP; these programming interfaces are documented in (Stevens 1993, Stevens 1998, Stevens 1999), for example. We used libraries provided by the equipment manufacturers to control the measurement devices, with exception of the HP-34970A that used a simple text protocol over a serial connection.

The software was written so that the measurements and stimulus would be as synchronized as possible. This was accomplished by a simple protocol: All components first prepare to start their operation as quickly as possible. For example, the stimulus module loads the bitmaps used as textures into memory and generates the geometric objects to be displayed, and the AD converter data acquisition settings are set. The signal to start the measurement is given after all components are ready. The measurement devices begin collecting data and the stimulus module begins to render the graphics. The underlying operating systems, Linux and Windows XP, are not realtime operating systems, and some unexpected events may cause the stimulus to miss a rendering deadline. To cope with this, the rendering conceptually samples a continuously running stimulus based on absolute time. Luckily, the synchronization is not critical, because the measured movements are relatively slow and all the parameters used in the included studies are averages in some sense.

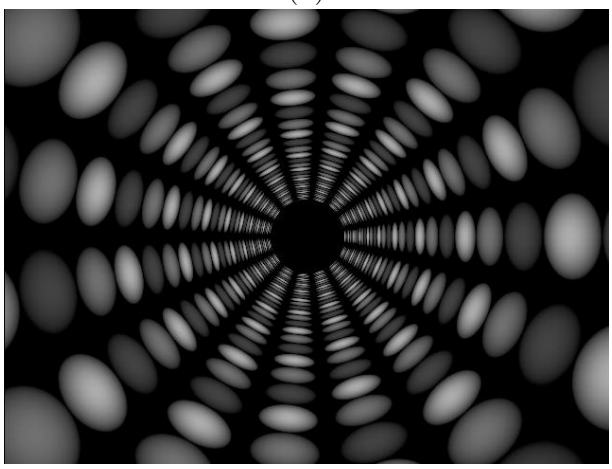
5.2 Applications

5.2.1 Pilot study

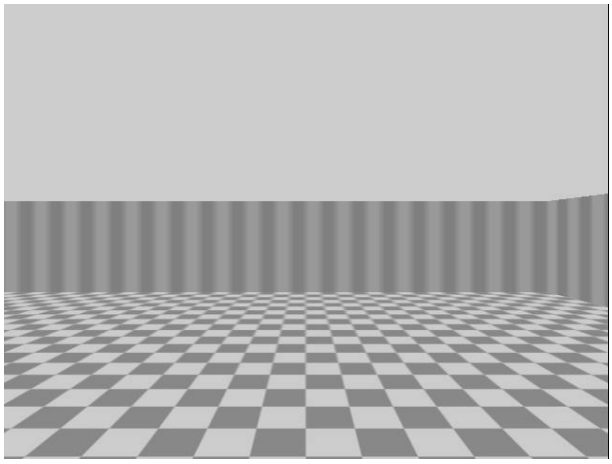
Our first study with virtual reality and postural control was a small study with only 3 test subjects. It concerned the effects of alcohol and virtual reality stimulation on balance. The study was reported in (Tossavainen, Juhola, Aalto, Toppila, Pyykkö,



(a)

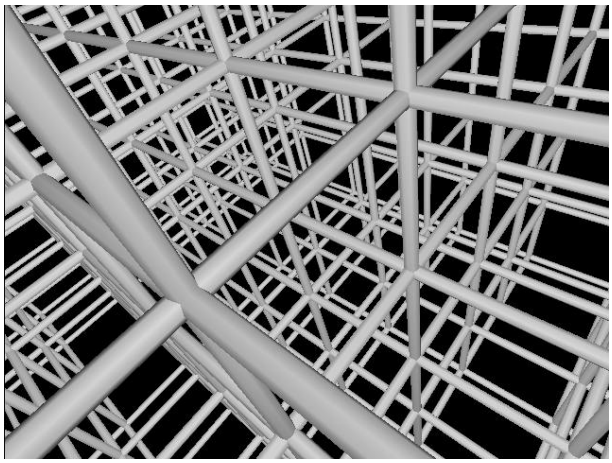


(b)

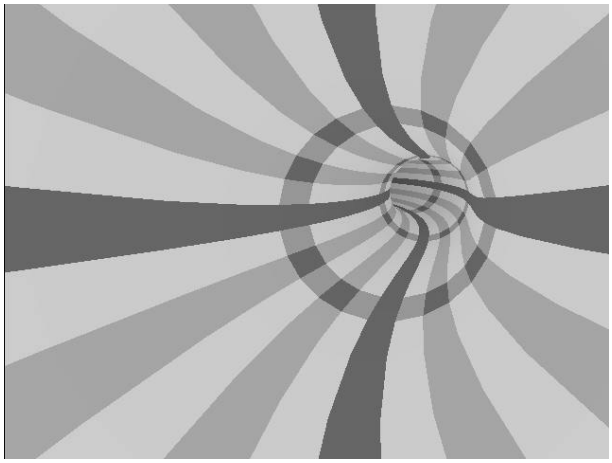


(c)

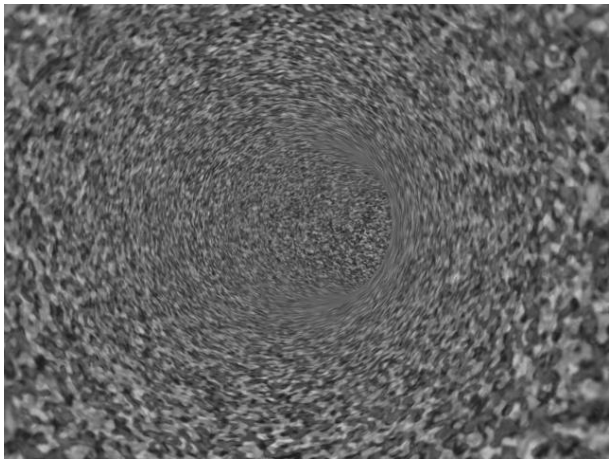
Figure 5.2. Virtual reality stimuli: (a) dots, (b) cylinder, and (c) moving and tilting room.



(a)



(b)



(c)

Figure 5.3. More VR stimuli: (a) lattice, (b) tunnel, and (c) textured tunnel with a random texture.

Honkavaara, Laurikkala & Laakso 2001). My role was to implement the VR software for this study.

In the test setup, one computer produced the VR environments, another computer collected the measurements, and the two computers were synchronized manually. The Sense8 WorldToolkit (WTK) library was used to implement the virtual environments. An Intergraph workstation equipped with an Intense3D Wildcat graphics accelerator rendered the graphics. It had 3 monitors: one displayed the user interface and the other two displayed the VR graphics. During the software development, our laboratory at the University of Tampere did not have a force platform or the HMD, and the development workstation only had one monitor. We performed the experiment at the Finnish Institute of Occupational Health in Vantaa. The experiments in Publication I use essentially the same hardware setup.

5.2.2 Affecting balance (I)

In the study presented in Publication I, we aimed to show that VR stimulation does have an effect on balance. We measured 30 healthy subjects (26 male, 4 female, aged 21-35 years) for 60 seconds while exposed to a static stimulus, consisting of an immobile scene with two blocks and a floor, and a moving tunnel stimulus while standing in a standard measurement posture on the force platform. The tunnel stimulus is similar that depicted in Figure 5.3 (b) with a different surface texture. It follows a parametric curve consisting of 4 sinewaves that are not harmonically related, 2 horizontal and 2 vertical, and constant velocity forward. The stimuli were administered using HMD, and the amount of swaying was assessed using sway path. The result was that there is a clear difference between the measurements: More swaying occurs during the tunnel stimulus. The results for sway path are mean 192.1 cm, std. dev. 55.3 cm, range 113.2–305.3 cm for the static stimulus and mean 258.1 cm, std. dev. 93.8 cm, range 126.8–484.1 cm for the tunnel stimulus. Three test subjects either needed support to stay upright during the tunnel stimulus or adjusted the HMD during the measurements. These cases were removed from the analysis. On the average, the sway path is 1.34 ± 0.28 (mean $\pm$ std. dev.) times longer during the tunnel stimulus. The Wilcoxon signed ranks test indicates that the difference in sway paths is statistically significant ($p = 1.23 \times 10^{-5}$).

5.2.3 Inducing different effects on balance (II)

Publication II demonstrates how virtual environments can cause different effects on balance. We measured 22 test subjects (20 male, 2 female, age 22–45) in six virtual environments: oscillating dots, dots rotating around 3 different axes, a rotating cylinder, and a tunnel. In addition, the subjects were measured with eyes open and closed and while wearing HMD without a visual stimulus. Dots is depicted in Figure 5.2 (a), cylinder is depicted in Figure 5.2 (b), and tunnel is depicted in Figure 5.3 (b). The oscillating dots rotates the environment $\pm 25^\circ$ for 15 seconds around the different axes at a frequency of 0.2 Hz. It pauses for 5 seconds between the oscillations. The rotating stimuli start to rotate with constant angular velocity, accelerating for 15 seconds and decelerate similarly to a halt in the next 15 seconds.

There is a pause of 10 seconds between the two rotations. The peak angular velocity of rotation just before deceleration is 160°/s in cylinder and 120°/s in rotating dots. The tunnel movement is similar to the tunnel stimulus in Publication I.

We evaluated swaying using COP MV (called sway path in Publication II). The visual stimulation increased MV by 8–83% on the average compared to the case of wearing the HMD with no stimulus. In particular, both tunnel and cylinder increase MV by more than 50%, and this increase was statistically significant by the Wilcoxon signed ranks test ($p \leq 0.013$ with Bonferroni correction). In addition to MV, we analyzed the leaning caused by the rotating stimuli. The mean position of COP is displaced during the rotation 0.54–0.86 cm from the mean position during the first 10 seconds when the stimulus was immobile. The direction of leaning is generally in the direction of rotation. For example, the subjects tend to lean left during counter-clockwise rotation around the line of sight. This can be interpreted as a reaction to illusory self-motion in the opposite direction. The Wilcoxon signed ranks test indicates that the leaning is significant in all cases.

We made a number of changes in the system for this study. The HP data acquisition device was controllable using a simple text-based protocol over a serial connection, so we implemented its control in the stimulation program to improve synchronization. This enabled the use of measures other than averages in the analysis, which was impossible earlier because of manual synchronization. We also ported the virtual environments from WTK to OpenGL for efficiency.

5.2.4 Laboratory bus installation

FIOH conducted a study on the effects of antioxidants on noise-induced temporary hearing loss and balance (Toppila, Pyykkö, Starck, Tossavainen, Nyman, Juhola & Oksa 2002). The study was supposed to use a discotheque as the noise source, and the measurement system was installed in a field laboratory bus. The level of noise turned out to be too low in the discotheque, so the exposure had to be conducted in the premises of FIOH. Nevertheless, the measurements were carried out in the bus as the laboratory was already installed in it. This shows that VR measurements can be used in the field, at least in principle. The VR equipment is fragile and has to be handled with care. A programmable motor-driven tilting platform was added to the hardware setup, and we made the required changes to the measurement program. I helped in setting up the measurement system in the bus and my VR software was used for the balance measurements.

5.2.5 CAVE experiments (III)

The Tampere Virtual Reality Center (VRC) at Tampere University of Technology built a CAVE (Cruz-Neira et al. 1992, Cruz-Neira, Sandin & DeFanti 1993) and the University of Tampere had the opportunity to use it. We only had the HMD, and its limitations—low resolution, narrow field of view, and weight—were apparent. The CAVE seemed well-suited for visual stimulation, so we arranged an experiment to compare it to the HMD using the same virtual environments on the same test subjects with both displays. Figure 5.4 contains photographs taken during the

experiment.

The study, reported in Publication III, included 20 healthy test subjects (17 male, 3 female, age 22–46). The test setup was otherwise similar to Publication II, but the stimuli only included dots rotating around the mediolateral axis, cylinder, and tunnel. In summary, dots and cylinder are more effective with the CAVE and tunnel is more effective with the HMD. We used Wilcoxon signed ranks test for statistical evaluation, and found that all stimuli cause a significant increase of swaying compared to the baseline measurement in quiet standing with eyes open. The differences between the displays are significant in the case of dots and tunnel ($p = 0.0002$ and $p = 0.001$, respectively), and also in cylinder ($p = 0.048$). In particular, dots is radically more effective with the CAVE. Dots with CAVE caused 3 test subjects to lose balance and 2 of them also lost balance during tunnel with CAVE. Everyone was able to complete all of the other tests.

Differences in field of view, resolution, contrast, and lighting conditions affect these results. With hindsight, we should have paid more attention to matching brightness and contrast between the two displays, especially for the tunnel stimulus. The HMD is brighter due to the shutter glasses used in CAVE, and also the black CAVE floor was not completely matte. Dots and cylinder scenes have a lot of black so this may affect them less than tunnel.

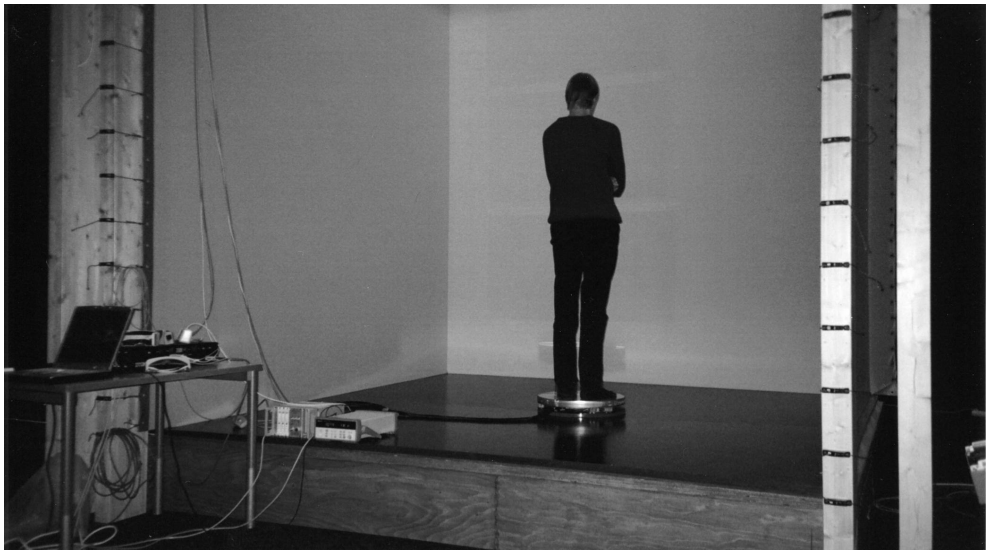
We had to port the virtual environments to CaveLib for display in the CAVE, so we extracted them into stand-alone executables that were implemented with plain OpenGL. These were easy to port to CaveLib, and it only took a few hours of programming in the VRC laboratory. We also extracted the data acquisition part into a stand-alone application and used a laptop computer to collect the data. Synchronization had to be done manually during the experiment.

5.2.6 Effects of solvent exposure on balance

The Finnish Institute of Occupational Health (FIOH) studied the effects of solvent exposure on balance using the VR measurement system (Toppila, Forsman, Pyykkö, Starck, Tossavainen, Oksa & Uitti 2006). They included 264 male subjects in the study and found that styrene is an additional risk factor in impaired postural stability, with the effects occuring already in younger workers. The VR posturography system used in the study is identical to the laboratory bus installation. I helped to set up the laboratory at FIOH.

5.2.7 Detecting balance disorders (IV, V)

We had extracted many parts of the system to independent components during our previous research. The Intergraph workstation had aged, and an inexpensive upgrade path was needed. Common graphics accelerators had exceeded it in performance, for the game industry drove their rapid development. The accelerators began to support two monitors, which was a requirement for our application. When we started our research, no common graphics card supported hardware-accelerated rendering on two synchronized monitors. Now this could be done, so we replaced the workstation with two standard desktop computers and equipped one of them with



(a)

(b)

Figure 5.4. CAVE experiments: (a) before stimulation, (b) during dots.

an nVidia 3D graphics accelerator. We also replaced the HP-34970A on RS-232 with a Data Translation DT-9800 series ADC connected with USB.

Different versions of the system included a projection display and a programmable moving platform. The software obviously needed to be more configurable, so that we could make changes faster in response to research needs. It was rewritten from the ground up, and we put more effort on the user interface, as the new measurement system was intended for use by laboratory assistants.

In Publication IV, we used the new measurement system to implement a series of tests. The series includes measurements in quiet standing with eyes open and eyes closed, the cylinder stimulus, a tilting platform stimulus without VR stimulation, and the tunnel stimulus with the tilting platform. It is designed to characterize a test subject's postural control comprehensively. We had implemented more advanced environments already, but cylinder and tunnel were kept, because we had used them in earlier studies. First the eyes open and eyes closed tests are administered. After these tests, the test subject puts on a safety harness to prevent falls, and the cylinder, the tilting platform, and the tunnel tests are performed.

As an application, we considered discriminating 33 controls (all male, age 24–46, mean 33, std. dev. 5) from 88 Ménière patients (23 male, 54 female, age 38–82, mean 60, std. dev. 10). We used MV as in the previous studies, and also introduced a new parameter called vertical force power fraction (VFPF) to catch the test subjects who were hanging or leaning on the safety harness. The parameter is the fraction of signal power below 1 Hz of the instantaneous weight signal after DC-removal computed from a standard periodogram. Its purpose is to detect (relatively) slow drifting in weight. VFPF proved to be more effective than MV for discrimination. Both of the parameters had significant differences between the groups, however. To evaluate the discriminative power of the test set, we applied univariate cutoff value classifiers, Fisher's linear discriminant (LD), k nearest neighbor (kNN) with automatic choice of k, and Adaboost with decision stumps to classify the cases into the respective groups and used leave-one-out cross-validation (LOOCV) to estimate the classification accuracy. We also used feature subset selection with LD and kNN. This test included only the cases with complete data: 33 control subjects and 55 Ménière cases. Some of the patients were unfit or unable to complete the entire series of tests. We found that classification slightly more accurate than 80% could be expected on the basis of the measurements. The baseline classifier, majority rule, that classifies every case to Ménière has an accuracy of 62.5%.

In Publication V, we take the same measurements and try to apply pattern recognition methods to describe how the measurements of controls and Ménière patients differ. As stated earlier, there is no established way to analyze stabilograms, and brute-force methods may give hints on where to start looking. We computed 156 different parameters from the test series and used a feature subset selection algorithm to select a set of 5 features that seemed to be good at describing the differences. The fitness of the feature subset was defined by a .632+-bootstrap error estimate of a simple 1NN classifier that standardizes the variables and uses the Euclidean metric. The accuracy of the subset selection process was estimated with LOOCV. The classification accuracy seems to improve slightly with the addition of more parameters, but there are too many correlated parameters with respect to the

number of measurements to choose the best subset with certainty. The differences of estimated discriminative power between parameter subsets may be just random variation.

During the work for Publication V, we were still experimenting with the analysis of the instantaneous weight. Consequently, the VFPF1 parameter is a different version from the one described in the text: Instead of subtracting the mean, it subtracts the best straight line fit from the signal before computing the periodogram. The two parameters are quite close. Also, the results in Publication IV are reported in percents of signal power, and in Publication V, they are reported as fraction of signal power. This explains the small differences in the VFPF results between the two publications.

5.2.8 Psychophysiological stimulation

In (Alatalo, Juhola, Surakka & Tossavainen 2005), we applied the measurement system in a psychophysiological experiment on approach-withdrawal reactions. We used a virtual environment, where a person runs directly toward the test subject, and changes direction just before collision to pass him by on his left or on his right. The facial expression of the runner varies between happy, neural, and angry. The event takes place in a city scene that is much more realistic than the environments used in our postural control studies. For example, it includes realistic buildings, animation, lighting, and dynamic shadows. The test included 20 healthy test subjects, who were exposed repeatedly to the stimulus with the runner's facial expression and passing sides randomized. We used force platform measurements to evaluate the response and found statistically significant evasive actions visible in the stabilograms before the impending collision. The study also shows that VR balance measurements have applications outside postural control research.

The stimulus was implemented as a module for my VR stimulation and measurement software. I helped with the integration work and with the balance measurements and their analysis.

Chapter 6

Discussion

We used essentially the same hardware to implement the test setups described in Chapter 5 and in Publications I–V. The setups used multiple visual stimuli during the course of the experiment, and later we also included a tilting force platform for mechanical perturbations. The stimuli caused quantifiable effects on balance, and we could design them to induce different effects, such as leaning in the rotating cylinder and tracking movements in the tunnel. Healthy subjects and patients with Ménière's disease had significantly differering responses to stimulation.

It was straightforward to implement the virtual environments used in Publications I–V, and we deliberately kept them simple. Complex environments would have made the results more difficult to interpret. The hardware also placed limitations on them, especially during the experiments in Publications I–III. Programming is still required for implementing new environments or modifying the existing ones. Parametric environments are a simple solution. For example, the tunnel can follow any parametric curve, so the curve could be generated in another application, e.g. MATLAB, and read from a file. The same applies to the path of movement in static environments. Importing virtual environments from 3D modelling programs would add even more flexibility. Stimuli with complex environments are probably easiest to implement with open source game engines or scene graph libraries such as Ogre3D and OpenSceneGraph.

There are some improvements that could be made to improve the usability of the system in a laboratory environment. First, the measurement logic could be improved. The present version only starts and stops the measurement and saves the measured data, but some test setups change the stimulus based on the previous measurements. Making the sequence scriptable based on previous test outcomes is a general solution. Second, measurements involving many tests and test subjects complicate the management of collected data and the integration of data from multiple sources. The identification of the measured data is done manually with filenames and directories in the present version. The solution is sufficient for now, but the issue will have to be dealt with in the future. This will probably require some changes in the architecture of the measurement system; a rewrite of the non-realtime portions, the user interface and remote control of the server, in a more dynamic programming language would be beneficial. The stimuli and the control of measurement devices should be decoupled from the server into stand-

alone executables to make the implementation of new stimuli easier, but this will introduce new synchronization problems and startup delays between measurements.

While the virtual environments worked for visual stimulation, they can still be improved. Some of the new environments that were not used in the included studies try to take perceptual issues into account. For example, the new tunnel can have arbitrary textures on the tunnel surface, and the lattice takes advantage of the parallax effect to improve movement perception. There is much to do in this area; the theories of motion perception in perceptual psychology are a good place to start.

Our HMD is quite cumbersome, and test subjects may have difficulties making the required user-dependent adjustments. The field of view is quite narrow, only 60° measured diagonally, and it rules out much of peripheral vision. A projection display might be better as it does not encumber the viewer with heavy equipment or need subjective adjustments. However, it may be difficult to eliminate the floor or other visible objects from the test subject's field of view with one, and the displays take a lot of space. We were concerned for the safety of test subjects, especially when using with the tilting platform, and added a safety harness to prevent falls. This caused some problems, as some of the subjects tended use the harness for extra support. We can probably improve its design.

The novel VFPF parameter introduced in Publication IV and the pattern recognition methods applied in Publication V seem promising. The feature subset selection methods need a lot of data, but they are valuable tools when faced with data that is difficult to interpret. In Chapter 3, we made novel improvements to algorithms for estimating COM movements from COP movements. Algorithm 2 is close to optimal within the stated assumptions. Given its simplicity, it is tempting to introduce new parameters based on the estimated COM movement. For example, patterns in COM accelerations may correspond to muscle activations. We did not have the chance to use VR stimulation for system identification, but it is possible. For example, the room stimulus in Figure 5.2 (c), can be used to show simple linear and angular movements. We also recorded the head orientations during stimulation; their analysis is still open.

In summary, we investigated the applicability of virtual reality as a stimulation method in postural control research. From the applications presented in Chapter 5 and in Publications I–V, and the results obtained in them, we can conlude that virtual reality is a versatile and effective stimulation method when applied to dynamic posturography.

Appendix A

Personal Contributions

I have worked in a multidisciplinary group during the studies presented in thesis. Given the breadth of the topic, this thesis would not exist without collaboration between scientists from different fields. However, in a thesis consisting of individual publications with multiple authors, I have to try to separate my individual contributions. They are as follows:

I. I implemented the virtual environments, designed the test setup, and conducted the measurements using a measurement program written by Esko Toppila. Martti Juhola and I wrote the article.

II. I designed the test setup, implemented the measurement software, conducted and analyzed the measurements, and wrote the article.

III. I ported the virtual enviroments to CaveLib with much assistance from the staff of Tampere Virtual Reality Center (VRC). VRC staff also assisted during the experiments with the CAVE. I performed the HMD experiments, analyzed the measurements, and wrote the article. Martti Juhola provided assistance in logistic arrangements.

IV. The test setup was designed by Esko Toppila and Ilmari Pyykkö. I wrote measurement software for VR tests and Esko Toppila's program was used for measurements without VR. Kalevi Nieminen designed and built the tilting force platform, and Esko Toppila programmed its movements. Laboratory staff at the Hearing Center of Tampere University Hospital collected the measurements. I analyzed the data and implemented all the algorithms used in the analysis. The VFPF parameter and platform calibration are my inventions. I wrote most of the article with contributions from Ilmari Pyykkö on medical aspects and suggestions from my other colleagues.

V. Publication V is completely my own work based on balance measurements collected in the laboratory in the Hearing Center.

Bibliography

Alatalo, T., Juhola, M., Surakka, V. & Tossavainen, T. (2005), Creation of a virtual reality stimulation for psychophysiological measurement. Submitted to Virtual Reality.

Baratto, L., Morasso, P. G., Re, C. & Spada, G. (2002), 'A new look at posturographic analysis in the clinical context: sway-density vs. other parameterisation techniques', *Motor Control* **6**(3), 248–273.

Bles, W. & Roos, J. W. P. (1991), 'The tilting room and posturography', *Acta Oto-Rhino-Laryngologica Belgica* **45**, 387–391.

Blinn, J. (1996), *Jim Blinn's Corner: A Trip Down the Graphics Pipeline*, Morgan Kaufmann, San Francisco.

Blinn, J. F. & Newell, M. E. (1976), 'Texture and Reflection in Computer Generated Images', *Communications of the ACM* **19**(10), 542–547.

Blinn, J. F. & Newell, M. E. (1978), Clipping using homogeneous coordinates, *in* 'SIGGRAPH '78: Proceedings of the 5th annual conference on computer graphics and interactive techniques', ACM Press, pp. 245–251.

Bottema, O. & Roth, B. (1990), *Theoretical Kinematics*, Dover.

Brandt, T., Paulus, W. & Straube, A. (1986), Vision and posture, *in* W. Bles & T. Brandt, eds, 'Disorders of Posture and Gait', Elsevier Science Publishers B.V., Amsterdam, pp. 157–175.

Bräuer, D. & Seidel, H. (1978), 'The autoregressive time series modelling of stabilograms', *Acta Biologica et Medica Germanica* **37**, 1221–1227.

Bräuer, D. & Seidel, H. (1979), 'Time series analysis of postural sway', *Agressologie* **20B**, 111–112.

Bräuer, D. & Seidel, H. (1980), 'The autoregressive structure of postural sway', *Agressologie* **21E**, 101–104.

Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M. & Hanrahan, P. (2004), 'Brook for GPUs: stream computing on graphics hardware', *ACM Transactions on Graphics* **23**(3), 777–786.

Carola, R., Harley, J. P. & Noback, C. R. (1990), *Human Anatomy & Physiology*, McGraw-Hill, New York.

Caron, O., Faure, B. & Brenière, Y. (1997), 'Estimating the centre of gravity of the body on the basis of the centre of pressure in standing posture', *Journal of Biomechanics* **30**(11/12), 1169–1171.

Carpenter, M. G., Frank, J. S., Winter, D. A. & Peysar, G. W. (2001), 'Sampling duration effects on centre of pressure summary measures', *Gait & Posture* **13**, 35–40.

Chiari, L., Cappello, A., Lenzi, D. & Croce, U. D. (2000), 'An improved technique for the extraction of stochastic parameters from stabilograms', *Gait & Posture* **12**, 225–234.

Chiari, L., Rocchi, L. & Cappello, A. (2002), 'Stabilometric parameters are affected by anthropometry and foot placement', *Gait & Posture* **17**, 666–677.

Chow, C. C. & Collins, J. J. (1995), 'Pinned polymer model of posture control', *Physical Review E* **52**(1), 907–912.

Collins, J. J. & De Luca, C. J. (1993), 'Open-loop and closed-loop control of posture: A random-walk analysis of center-of-pressure trajectories', *Experimental Brain Research* **95**, 308–318.

Collins, J. J. & De Luca, C. J. (1994), 'Random walking during quiet standing', *Physical Review Letters* **73**(5), 764–912.

Collins, J. J. & De Luca, C. J. (1995*a*), 'The effects of spaceflight on open-loop and closed-loop postural control mechanisms: human neurovestibular studies on sls-2', *Experimental Brain Research* **107**, 145–150.

Collins, J. J. & De Luca, C. J. (1995*b*), 'The effects of visual input on open-loop and closed-loop postural control mechanisms', *Experimental Brain Research* **103**, 151–163.

Collins, J. J. & De Luca, C. J. (1995*c*), 'Upright, correlated random walks: A statistical-biomechanics approach to the human postural control system', *Chaos* **5**(1), 57–63.

Cook, R. L. & Torrance, K. E. (1982), 'A Reflectance Model for Computer Graphics', *ACM Transactions on Graphics* **1**(1), 7–24.

Cruz-Neira, C., Sandin, D. J. & DeFanti, T. A. (1993), Surround-screen projection-based virtual reality: the design and implementation of the CAVE, *in* 'Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques', ACM Press, pp. 135–142.

Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V. & Hart, J. C. (1992), 'The CAVE: audio visual experience automatic virtual environment', *Communications of the ACM* **35**(6), 64–72.

Eberly, D. H. (2001), *3D Game Engine Design*, Morgan Kaufmann, San Diego.

Eves, H. (1990), *An Introduction to the History of Mathematics*, 6th edn, Saunders College Publishing, Philadelphia, Pennsylvania.

Ferdjallah, M., Harris, G. F. & Wertsch, J. J. (1999), 'Instantaneous postural stability characterization using time-frequency analysis', *Gait & Posture* **10**, 129–134.

Foley, J. D., van Dam, A., Feiner, S. K. & Hughes, J. F. (1989), *Computer Graphics: Principles and Practice*, Addison-Wesley, Reading.

Fransson, P.-A., Hafström, A., Karlberg, M., Magnusson, M., Tjäder, A. & Johansson, R. (2003), 'Postural control adaptation during galvanic vestibular and vibratory proprioceptive stimulation', *IEEE Transactions on Biomedical Engineering* **50**(12), 1310–1319.

Godement, R. (1968), *Algebra*, Kershaw Publishing Company, London.

Goldman, R. (2002), 'On the algebraic and geometric foundations of computer graphics', *ACM Transactions on Graphics* **21**(1), 52–86.

Goldman, R. N. (1985), 'Illicit expressions in vector algebra', *ACM Transactions on Graphics* **4**(3), 223–243.

Goldstein, E. B. (2002), *Sensation and Perception*, 6th edn, Wadsworth, Pacific Grove, CA.

Goral, C. M., Torrance, K. E., Greenberg, D. P. & Battaile, B. (1984), Modeling the interaction of light between diffuse surfaces, *in* 'SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques', ACM Press, New York, NY, USA, pp. 213–222.

Grzegorzewski, B. & Kowalczyk, A. (2001), 'First-order statistics of human stabilogram', *Human Movement Science* **20**, 853–866.

Gurfinkel, E. V. (1973), 'Physical foundations of the stabilography', *Agressologie* **14**(C), 9–14.

Heckbert, P. S. & Moreton, H. P. (1991), Interpolation for polygon texture mapping and shading, *in* D. F. Rogers & R. A. Earnshaw, eds, 'State of the Art in Computer Graphics: Visualization and Modelling', Springer-Verlag, New York, pp. 101–111.

Hufschmidt, A., Dichgans, J., Mauritz, K.-H. & Hufschmidt, M. (1980), 'Some methods and parameters of body sway quantification and their neurological applications', *Archiv für Psychiatrie und Nervenkrankheiten* **228**, 135–150.

Jacobson, J., Redfern, M. S., Furman, J. M., Whitney, S. L., Sparto, P. J., Wilson, J. B. & Hodges, L. F. (2001), Balance NAVE: a virtual reality facility for research and rehabilitation of balance disorders, *in* 'Proceedings of the ACM

Symposium on Virtual Reality Software and Technology', ACM Press, pp. 103–109.

Johansson, R., Magnusson, M. & Åkesson, M. (1988), 'Identification of human postural dynamics', *IEEE Transactions on Biomedical Engineering* **35**(10), 858–869.

Johansson, R., Magnusson, M., Fransson, P.-A. & Karlberg, M. (2001), 'Multistimulus multi-response posturography', *Mathematical Biosciences* **174**, 41–59.

Kajiya, J. T. (1986), The rendering equation, *in* 'SIGGRAPH '86: Proceedings of the 10th annual conference on computer graphics and interactive techniques', ACM Press, pp. 143–150.

Kalat, J. W. (1984), *Biological Psychology*, 2nd edn, Wadsworth Publishing Company, Belmont, CA.

Kalawsky, R. S. (1993), *The Science of Virtual Reality and Virtual Environments*, Addison-Wesley, Reading.

Karlsson, A. & Frykberg, G. (2000), 'Correlations between force plate measures for assessment of balance', *Clinical Biomechanics* **15**, 365–369.

Keshner, E. & Kenyon, R. (2000), 'The influence of an immersive virtual environment on the segmental organization of postural stabilizing responses', *Journal of Vestibular Research* **10**, 207–219.

Kim, N. G., Yoo, C. K. & Im, J. J. (1999), 'A new rehabilitation training system for postural balance control using virtual reality', *IEEE Transactions on Rehabilitation Engineering* **7**(4), 482–485.

King, D. L. & Zatsiorsky, V. M. (1997), 'Extracting gravity line displacement from stabilographic recordings', *Gait & Posture* **6**, 27–38.

Kochanek, D. H. U. & Bartels, R. H. (1984), Interpolating splines with local tension, continuity, and bias control, *in* 'Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques', ACM Press, pp. 33–41.

Kramer, P. D., Roberts, D. C., Shelhamer, M. & Zee, D. S. (1998), 'A versatile stereoscopic visual display system for vestibular and oculomotor research', *Journal of Vestibular Research* **8**(5), 363–379.

Kuczyński, M. (1999), 'The second order autoregressive model in the evaluation of postural stability', *Gait & Posture* **9**, 50–56.

Kuno, S., Kawakita, T., Kawakami, O., Miyake, Y. & Watanabe, S. (1999), 'Postural adjustment response to depth direction moving patterns produced by virtual reality graphics', *Japanese Journal of Physiology* **49**, 417–424.

Lafond, D., Duarte, M. & Prince, F. (2004), 'Comparison of three methods to estimate the center of mass during balance assessment', *Journal of Biomechanics* **37**, 1421–1426.

Le Clair, K. & Riach, C. (1996), 'Postural stability measures: what to measure and for how long', *Clinical Biomechanics* **11**(3), 176–178.

Lee, H.-Y., Cherng, R.-J. & Lin, C.-H. (2004), 'Development of a virtual reality environment for somatosensory and perceptual stimulation in the balance assessment of children', *Computers in Biology and Medicine* **34**(8), 719–733.

Lenzi, D., Cappello, A. & Chiari, L. (2003), 'Influence of body segment parameters and modeling assumptions on the estimate of center of mass trajectory', *Journal of Biomechanics* **36**, 1335–1341.

Levin, O. & Mizrahi, J. (1996), 'An iterative model for estimation of the trajectory of center of gravity from bilateral reactive force measurements in standing sway', *Gait & Posture* **4**, 89–99.

Maki, B. E. (1986), 'Selection of pertubation parameters for identification of the posture-control system', *Medical & Biological Engineering & Computing* **24**, 561–568.

Maki, B. E. & Fernie, G. R. (1988), A system identification approach to balance testing, *in* O. Pompeiano & J. H. J. Allum, eds, 'Progress in Brain Research', Vol. 76, Elsevier Science Publishers B.V., chapter 26, pp. 297–306.

Mitchell, S. L., Collins, J. J., Luca, C. J. D., Burrows, A. & Lipsitz, L. A. (1995), 'Open-loop and closed-loop postural control mechanisms in parkinson's disease: increased mediolateral activity during quiet standing', *Neuroscience letters* **197**, 133–139.

Möller, T. & Haines, E. (1999), *Real-Time Rendering*, A.K. Peters, Natick, MA.

Morasso, P. G., Spada, G. & Capra, R. (1999), 'Computing the COM from the COP in postural sway movements', *Human Movement Science* **18**, 759–767.

Nashner, L. M. (1971), 'A model describing vestibular detection of body sway motion', *Acta Otolaryngologica* **72**, 429–436.

Nashner, L. M. (1985), Strategies for organization of human posture, *in* M. Igarashi & F. O. Black, eds, 'Vestibular and Visual Control on Posture and Locomotor Equilibrium', Karger, Basel, Switzerland, pp. 1–8.

Önell, A. (2000), 'The vertical ground reaction force for analysis of balance?', *Gait & Posture* **12**, 7–13.

Peterka, R. J. (2000), 'Postural control interpretation of stabilogram diffusion analysis', *Biological Cybernetics* **82**, 335–343.

Phong, B.-T. (1975), 'Illumination for computer generated pictures', *Communications of the ACM* **18**(6), 311–317.

Proakis, J. G. & Manolakis, D. G. (1996), *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd edn, Prentice Hall, Upper Saddle River, New Jersey.

Redfern, M. S., Yardley, L. & Bronstein, A. M. (2001), 'Visual influences on balance', *Journal of Anxiety Disorders* **15**, 81–94.

Riley, M. A., Balasubramaniam, R. & Turvey, M. T. (1999), 'Recurrence quantification analysis of postural fluctuations', *Gait & Posture* **9**, 65–78.

Rocchi, L., Chiari, L. & Cappello, A. (2004), 'Feature selection of stabilometric parameters based on principal component analysis', *Medical & Biological Engineering & Computing* **42**, 71–79.

Roy, S. H., Ladin, Z. & De Luca, C. J. (1987), Experimental evidence for a random process model of postural sway, *in* 'The IEEE Ninth Annual Conference of the Engineering in Medicine and Biology Society, Boston, MA', p. 759.

Schlag, J. (1991), Using geometric constructions to interpolate orientation with quaternions, *in* J. Arvo, ed., 'Graphics Gems', Vol. II, Academic Press, San Diego, pp. 377–380.

Schmid, M., Conforto, S., Camomilla, V., Cappozzo, A. & D'Alessio, T. (2002), 'The sensitivity of posturographic parameters to acquisition settings', *Medical Engineering & Physics* **24**, 623–631.

Schumann, T., Redfern, M. S., Furman, J. M., El-Jaroudi, A. & Chaparro, L. F. (1995), 'Time-frequency analysis of postural sway', *Journal of Biomechanics* **28**(5), 603–607.

Segal, M. & Akeley, K. (2003), *The OpenGL® Graphics System: A Specification (Version 1.5)*, Silicon Graphics. Retrieved January 7, 2006, from http://www.opengl.org.

Seidel, H., Bräuer, D., Bastek, R. & Issel, I. (1978), 'On the quantitative characterization of human body sway in experiments with long-term performance', *Acta Biologica et Medica Germanica* **37**, 1551–1561.

Shimba, T. (1984), 'An estimation of center of gravity from force platform data', *Journal of Biomechanics* **17**(1), 53–60.

Shoemake, K. (1985), Animating rotation with quaternion curves, *in* 'Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques', ACM Press, pp. 245–254.

Stevens, W. R. (1993), *Advanced Programming in the UNIX® Environment*, Addison Wesley Longman Inc., Reading, Massachusetts.

Stevens, W. R. (1998), *UNIX Network Programming*, Vol. 1, 2nd edn, Prentice-Hall, Upper Saddle River, New Jersey.

Stevens, W. R. (1999), *UNIX Network Programming*, Vol. 2, 2nd edn, Prentice-Hall, Upper Saddle River, New Jersey.

Sutherland, I. E. & Hodgman, G. W. (1974), 'Reentrant polygon clipping', *Communications of the ACM* **17**(1), 32–42.

Toppila, E., Forsman, P., Pyykkö, I., Starck, J., Tossavainen, T., Oksa, P. & Uitti, J. (2006), 'Effect of styrene on postural stability among reinforced plastic plant workers in Finland', *Journal of Occupational and Environmental Medicine* **48**(2), 6 pages. To appear.

Toppila, E., Pyykkö, I., Starck, J., Tossavainen, T., Nyman, P., Juhola, M. & Oksa, P. (2002), Protection of inner ear against acute environmental noise with antioxidants, *in* 'XXII Barany Society Meeting'. Abstract.

Tossavainen, T., Juhola, M., Aalto, H., Toppila, E., Pyykkö, I., Honkavaara, P., Laurikkala, J. & Laakso, J. (2001), 'Postural control as assessed with virtual reality', *Acta Otolaryngologica Supplement* **545**, 53–56.

van Asten, W. N. J. C., Gielen, C. C. A. M. & van der Gon, J. J. D. (1988), 'Postural adjustments induced by simulated motion of differently structured environments', *Experimental Brain Research* **73**, 371–383.

Watt, A. (2000), *3D Computer Graphics*, 3rd edn, Addison-Wesley, Harlow, England.

Watt, A. & Watt, M., eds (1992), *Advanced Animation and Rendering Techniques*, Addison-Wesley, Harlow, England.

Whitted, T. (1980), 'An improved illumination model for shaded display', *Communications of the ACM* **23**(6), 343–349.

Widder, D. W. (1989), *Advanced Calculus*, 2nd edn, Dover, Mineola, New York.

Winter, D. A. (1990), *Biomechanics and Motor Control of Human Movement*, 2nd edn, Wiley Interscience, New York.

Yamada, N. (1995), 'Chaotic swaying of the upright posture', *Human Movement Science* **14**, 711–726.

Zatsiorsky, V. M. & King, D. L. (1998), 'An algorithm for determining the gravity line location from posturographic recordings', *Journal of Biomechanics* **31**, 161–164.