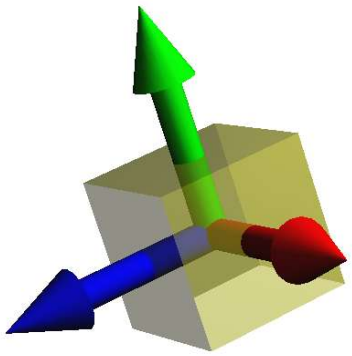
Using linear filters for real-time smoothing of rotational data in a virtual reality application



Rob C.P. van den Bogaard

2 August 2004

Master of Science Thesis for Artificial Intelligence
supervisor: dr. ir. L. Dorst
Intelligent Autonomous Systems
Faculty of Science
Universiteit van Amsterdam

I would like to thank my supervisor Leo Dorst, Robert van Liere, Eus van Someren, my girlfriend Saskia, parents, brothers, friends and colleagues from Amstel Instituut for their support and patience!

Abstract

At CWI (Centrum voor Wiskunde en Informatica) a desk-top Virtual Reality apparatus, the Personal Space Station, was built which enables its users to interact with a computer generated 3D scene by manipulating real-world objects. A mirror reflects the generated scene in such a way that virtual objects seem to coincide with their real-world counterparts, in personal space, the natural space in which humans for instance connect two Lego bricks together or place their cups of coffee. The Personal Space Station (PSS) measures where the real objects are and what their pose is in order to be able to generate a matching virtual scene. Unfortunately the measurements are not free of noise: Small, fast oscillations in rotational motion estimates disturb the virtual reality experience.

To counteract the noise, rotation data can be processed by computationally cheap linear filters suitable for on-line use, although rotations form a cyclic non-linear space. Quaternions are a compact and convenient representation of rotations. Their linear tangent spaces are easily entered and left through the quaternion logarithm and exponent, which is exploited by a method found in the literature ([Lee2002]). It linearizes the filtering task by application of a finite impulse response (FIR) filter in rotational tangent space.

In this report an automated procedure is described to tune a FIR filter to the application using captured motion data from the PSS. The resulting filter is extended with prediction based on velocity extrapolation and quantitative methods are presented to evaluate filter performance during the actual use of the PSS. A comparison is made between performances of the FIR filter and a predictive Double Exponential Smoothing (DES) and Kalman filter, both acting linearly on quaternion components.

The FIR filter, apart from being the computationally cheapest method, performs best when the rotational signal is heavily oversampled. DES and Kalman perform comparably and better than FIR at lower lags, which is useful in applications with less oversampling. The DES algorithm is extremely memory friendly and has only two free parameters, whereas the Kalman filter requires much more to be tuned. The standard Kalman filter used is relatively computation power-friendly while performing at the same level as DES, although in other work the (extended) Kalman filter is reported to be extremely expensive compared with DES of the same performance. The experiments also show that errors introduced by disregarding the non-linear nature of rotations turn out to be of no significance for the PSS and similar applications if quaternions are employed.

Contents

Chapter 1

Introduction

This report describes a solution for a problem encountered during the implementation of a Virtual Reality apparatus that presents the user a 3D depth cued virtual scene. One or more objects in this scene mimic the motion of their corresponding "input device", an object in the "real" world (typically a cube or wand marked with spots). The user can directly manipulate those devices by holding them in his or her hands. For the experience in Virtual Reality to be comfortable and convincing the resulting motion of the manipulated objects in the virtual scene should be like a "shadow" of the corresponding real world input devices, preferably a shadow that is as tight as possible.

Due to measurement and reconstruction uncertainties the part of the system that tracks the motion of the input devices suffers from errors that to the user of the Virtual Reality environment have the appearance of a "*tremor*", a relatively fast oscillation that is added as rotational noise to motion. We would like to reduce this tremor by filtering captured rotational motion data, however the non-linear nature of rotation space suggests that simply filtering rotational data by traditional means of convolution with a filter kernel or other linear methods may not yield the expected results.

In section 1.1 the "Personal Space Station" virtual reality apparatus is introduced, the application in which the filtering algorithm is to be implemented. Section 1.2 sketches the context of the motion tracking errors in a problem statement. In section 1.3 the PSS is described in a more functional manner to show from where in the apparatus the identified problem is going to be approached and in section 1.4 related work in the field of rotational data filtering is discussed. An overview of the rest of the report is provided by section 1.5.

1.1 The Personal Space Station

The "*Personal Space Station*" (PSS) [Mulder2002] is developed as a low-cost *near-field virtual reality apparatus with graspable reach-in user interface* being developed by the National Research Institute for Mathematics and Computer Science in the Netherlands (CWI). The PSS consists of a frame on top of which a regular display device (such as a CRT monitor) is placed with the screen side down. A mirror is mounted underneath it in order to reflect displayed images into the eyes of the user who is seated in the front of the PSS. At the rear side of the frame two infra-red sensitive cameras are placed several decimeters apart aiming at the space below the mirror. The user can reach into this interaction space to manipulate input devices (objects such as a optically

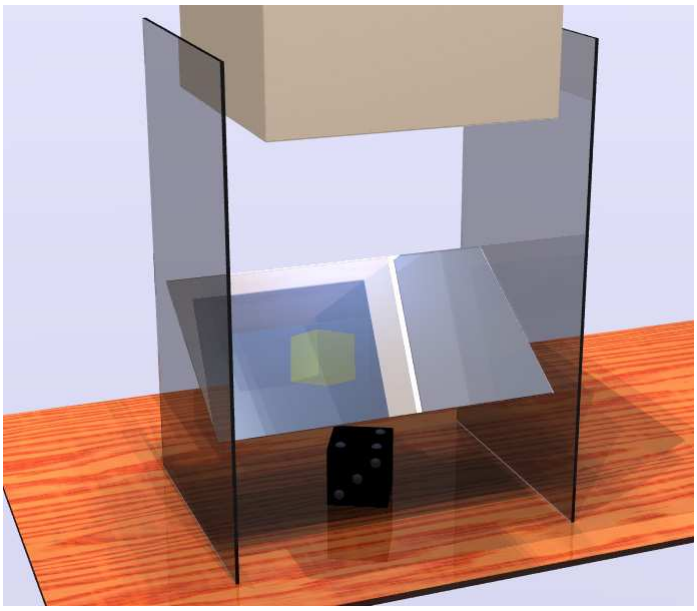


Figure 1.1: *The image of the monitor is reflected by a tilted mirror in order to let the virtual scene coincide with the (real world) scene in which the user of the Personal Space Station manipulates input devices.*

marked cube or pen). An ultra-sound emitter is oriented toward the user, who wears virtual reality look-through shutter glasses. To determine the position and orientation of the user's head, the glasses have sound sensors attached that pick up the sound signal from the emitter.

Virtual reality applications fake sensoric stimuli and results of actuator actions. The user of a VR application experiences the presence of objects in a time, at a place, scale, or position along an other dimension (e.g. color) that would be different without intervention of the application. In that perspective a telescope could also be considered a kind of virtual reality device presenting an altered *visual space*, although it only fakes the scale or distance of objects and actuation is limited to panning, tilting (spherical motion) and zooming. A view like this ultimately recognizes the actual "virtualness" of the reality presented by our busy eyes. However (not to over-complicate things), what most people refer to when talking about virtual reality are applications that present a visual *virtual scene* in which objects are presented in at least three spatial dimensions, evoking an ever increasing experience of virtual presence as technology advances.

A *near-field* virtual reality console presents a virtual scene to its user (the *operator*) in which the interaction with objects within physical reach of the operator is simulated. Operators can engage in the virtual scene by holding one or more input devices in their hands and moving them around in *interaction space*. Because the (spatial) configuration of the input devices in interaction space is to be tracked by the console, another space is of importance: *tracking space*, made up from all object configurations that can actually be tracked by the sensing facilities of the console.

The design of the PSS is aimed at letting the spaces coincide with *Personal Space*,

the natural space in which humans tend to handle and examine objects, operate devices and make gestures. This space is located in front of the operator with the center approximately at half an arm's length. For human beings this is a comfortable space to interact in, because they can assume a stable posture during actions. When sitting behind a desk, elbows or upper arms can be supported by the desk and the head can remain relaxed: To focus on the center of actions, the head and eyes can adopt approximately the same orientations as one automatically adopts casually walking down the streets or staring at one's desk during the four-o-clock collapse...

When the user reaches into the PSS, under the mirror, to manipulate the input devices in personal space, his/her view onto the virtual scene is not blocked by the devices itself or the user's hands and arms: *Visual space* is effectively altered to reflect the virtual scene, because the user experiences the effect of holding a rigid object in one's hand at the same position and in the same orientation as the virtual objects appear to be by looking into the mirror.

To enhance the realism of the virtual visual space the user wears shutter glasses which present the image shown in the mirror to the left and right eye of the user in an alternating manner. Whenever the 3D renderer of the PSS receives a signal from the glasses that the user's view is switched it will generate the virtual scene in the correct projection for the current eye. In this way the eyes are given depth cues improving 3D realism.

The glasses also measure the time the ultra-sound emitted from the back of the PSS takes to travel to three sound sensors attached to the glasses in a triangular constellation. With this information the PSS can infer the position and orientation of the user's head relative to the frame with which the personal space is aligned. Adaptation of the projection parameters by the renderer ensures that the experienced configuration of virtual visual space relative to real virtual space is invariant under head translation and rotation. In other words, the user can look around objects: the objects seem to remain in their positions and orientations relative to the real world.

1.2 Problem statement

To determine the configuration of the input devices in interaction space and the location and aim of the user's head, properties of the world are measured that bear consequences of changes in the parameters that are to be tracked (position, orientation and identity of input devices and the way the user is looking in the mirror). Cameras measure the amount and distribution of light projected on a rectangular area of a plane at a specific position, with a specific orientation. The PSS uses two cameras to exploit the combined geometric constraints on the projections and derives in this way the distances to points on a tracked object relative to a perception frame (for instance coinciding with the focal point of one of the cameras). With the use of depth information from three or more object points the orientation of the object is inferred.

Ultra-sonic distance sensors measure the distance to an emitter somewhat more directly than a constellation of cameras can, but the described principle for determining the orientation of an object also holds for this sensing method. In 3D space the configuration of a plane is completely determined by specifying three points in a triangular non-collinear (all points lie in a plane, but not on one line) arrangement, which is the reason why three points are enough. Theoretically speaking additional points would be spurious, if coplanar with the other three, and otherwise produce a volume instead of a plane. In practical applications though, measured point positions will be disturbed

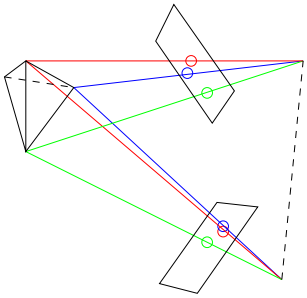


Figure 1.2: *Using stereoscopy to determine the pose of an observed object.*

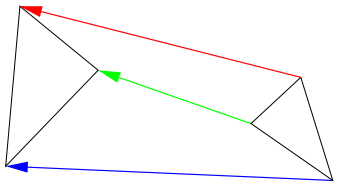


Figure 1.3: *Using ultra-sound to determine the pose of an observed object. The sound sensors are able to distinguish sounds from different emitters, because each emitter transmits sound with a unique frequency or other modulation.*

by noise which introduces uncertainty about the exact configuration of a plane. The redundancy of extra measurements can help to minimize this.

Monitored properties of the "outside" world will have to be inferred from a complicated chain (or directed graph) of measurements, uncertainties on those measurements, modeled filters, uncertainty induced by the models used to build the filters and calculation errors because of the discontinuous nature of the computation device. Uncertainties are recognized in errors and incompleteness and arise in the data during their progression through transformations, almost always increasing although decreases are possible by combining different uncertainties of multiple sources or extra measurements.

Because of this chain of derivations it is not surprising that a straight forward application of all theories involved is either very complicated to set up. To determine the nature of the operational uncertainty of an application one can try to calculate the complete graph through or analyse an intersection of the data streams (watching the data at a point in the graph) that satisfyingly represents the operation of the application.

The most important "intersection" for the PSS itself are the measured spatial configurations of the input devices and the user's head. The most important intersection for the complete "application graph" though, which includes the user viewing the virtual scene and manipulating input parameters of the PSS, is the extent to which the user experiences the virtual scene to be tightly fixed to and embedded into the real environment. This observation justifies a rather crude and qualitative evaluation of the user's experience to start with.

Using the first version of the PSS application users would notice that virtual objects

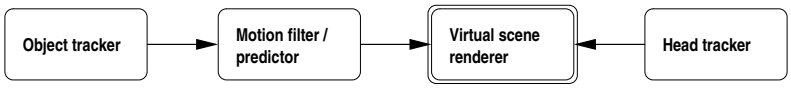


Figure 1.4: *The virtual scene renderer using input from the object tracker and the head tracker to determine the projection parameters to use when displaying the scene. The orientation filter processes the data from the object tracker before it is delivered to the renderer.*

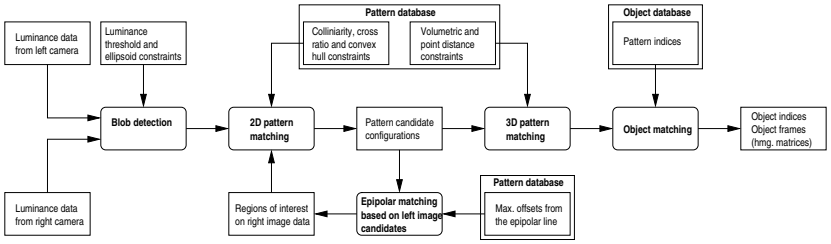


Figure 1.5: *Overview of the algorithm which estimates the 3D locations and poses of observed input devices from infra-red luminance data acquired by a stereoscopic arrangement of cameras.*

float at approximately the same position the input device is held, but that the orientations of tracked virtual objects seem to oscillate around the orientation which the user expects the object to have. In addition the PSS sometimes loses track of an input device, because it is held in a part of tracking space that yields configuration uncertainties that are too big to be decisive or its appearance to the cameras is too ambiguous to cope with. The discussed solution only addresses the erroneous oscillation of tracked object orientation, based on an analysis of the orientation signal coming from the PSS module that determines the spatial configurations of the input device objects.

1.3 Modules of the Personal Space Station

The PSS roughly consists of an input device and head tracker, and a virtual scene renderer. Figure 1.4 shows these main modules with their data flow relations and indicates where the orientation filtering module described in this report fits in. Figure 1.5 shows the sub-modules of the input device tracker which feeds the filter with motion data. Two infra-red cameras are connected to a PC and provide a stereographic infra-red image of *tracking space*. Retro-reflective markers on the input devices facilitate image segmentation which separates the scene background from the pixels whose luminance is increased by the reflection of infra-red light in the markers. From these image pairs (one for the left, one for the right camera) the centers of the markers are inferred in 2D image coordinates. A search on the 2D information determines which 3D objects correspond to the detected coordinates ([Liere2002]). Stereoscopy associates the 2D origins and orientations of the lines and/or faces found in the left and right image with 3D frames in tracking space. These frames are combined to form the 3D reference frames of the objects corresponding to the input devices moved around by the user.

1.4 Related Work

Orientation filtering techniques are inspired by problems in several areas such as computer graphics, robotics and human-computer interfaces, each of which has its own specific requirements and different methods.

A substantial part of the literature on the filtering of rotational data deals with interpolating orientations in the context of animation. In this context synthesized or captured motion key frames have to be interpolated to generate smooth trajectories that are natural to the eye. If the animator defines key frames by hand, interpolation distance is typically greater than when frames are acquired using a motion tracking system. Techniques like spline-interpolation or Bezier-curves (described in [Shoemake1985]) can help to produce smooth rotational motion paths from sparse data, but require fine-tuning by the human operator (adjusting control points, deciding how much information the fitting is allowed to introduce to the curves). Spline interpolation minimizes curve strain energy over the entire data set, which is not very useful for applications using only a part of the data at one time, but useful for extrapolating motion. Bezier curves have the advantage that they do not need the entire data set at once (only a few data points are needed), but they do not take continuity constraints into account at the end-points of the data used to construct a single Bezier curve.

For methods using this techniques quaternions are suited well, because they are a compact yet meaningful representation of orientations which eliminates the major problem experienced with Euler angles or equivalents (gimbal lock). However, some of the methods apply the curve fitting technique of choice directly on the 4D quaternion coefficients. In this way the interpolation curves can diverge from the surface on the hypersphere which represents orientations. To keep them on the sphere the resulting vectors are simply *renormalized*. This compresses the curve near the two end-points and inflates the curve in between, which means that, unintentionally, the velocity of the motion changes faster near the key frames than it does mid-way. Note that there always exist three orientations which are in fact correct interpolations using the renormalization technique (see figure 1.6): the trivial interpolations coinciding with the endpoint orientations $q(0)$ and $q(1)$, plus the interpolation exactly mid-way $q(0.5)$. Renormalization of interpolated data preserves rotation axes, but the interpolation does not evenly distribute the rotations as intended.

To avoid the problems of renormalization efforts are made to use spherical linear interpolation (*slerp*, see [Shoemake1985]) with the aforementioned methods:

$$\begin{gathered} \mathbf{q}_t = \mathbf{q}_0 e^{t \cdot \log\left(\mathbf{q}_0^{-1}\mathbf{q}_1\right)} = \mathbf{q}_0 \left(\mathbf{q}_0^{-1}\mathbf{q}_1\right)^t \\ t = 0 \rightarrow \mathbf{q}_t = \mathbf{q}_0 \cdot 1 \\ \vdots \\ t = 1 \rightarrow \mathbf{q}_t = 1 \cdot \mathbf{q}_1 \end{gathered} \tag{1.1}$$

Slerp does take the curved nature of orientation space into account and can be used with both unit quaternion and matrix representation of orientations. It exploits the properties of the exponential map, which is defined for both representations. Taking the inverse of the exponent, the logarithm, of a rotation from one orientation to another (a *relative orientation*) yields a 3-dimensional vector from the first to the second orientation. Scaling the vector with a factor between 0 and 1 interpolates between the orientations along the shortest geodesic path (for complex numbers the shortest path on the unit circle, and for quaternions the shortest path on the unit quaternion hypersphere). The shortest geodesic is a good measure, because its length corresponds in a one-to-one

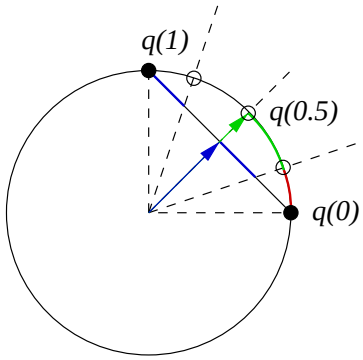


Figure 1.6: *Renormalization of non-unit quaternions distorts the distances between successive interpolated orientations. This picture shows that renormalization of linear, non-spherical interpolation (indicated with blue lines) causes the motion to accelerate in the first half of the travel from $q(0)$ to $q(1)$ and to slow down again in the second half. Spherical linear interpolation (slerp) interpolates along the arc, the curved path between rotations (the red and green lines in the figure).*

manner with the angle over which the first orientation has to be rotated to coincide with the second (in the plane through the origin shared by the two orientations).

A recent paper from Lee and Shin [Lee2002] describes a filtering method, inspired by spherical linear interpolation and the signal processing concept of finite impulse response linear filters. The method operates in orientation tangent space which is approximately linear and ensures that the output signal stays in the same rotation-space on the unit hypersphere.

In [Laviola2003] a predictive double exponential smoothing (PDES) filter was compared to a predictive Extended Kalman filter and found to yield approximately the same accuracy using much less computation power. The PDES filter operates in the euclidean space of quaternion components and therefore renormalization is required to keep the data on the same rotation-hypersphere. The performance of the double exponential smoothing depends on a smoothing parameter to be set manually.

In [Azuma2002] a Kalman filter is applied on the quaternion components and the angular rate of change of a tracked orientation. Renormalisation is used to keep the quaternions on the unit hyper-sphere, which suffers from the earlier mentioned problem of distorting the rate of change between to data points. Kalman filtering has some strong advantages over the other methods described, because it is a filtering technique that combines model based smoothing (expectations are combined with actual measurements) with (linear) optimal minimization of errors in the expectations the model provides through on-line adaptation of stochastic confidence in model and measurements.

Because it follows an iterative and automated procedure the Kalman filter is especially suited for tasks that have to be performed without user intervention as opposed to the (animator guided) techniques generally used in motion picture animation which are aimed at synthesizing motion not necessarily tied to a specific motion model and techniques which require tuning parameters not explicitly related to a theoretical model. The difficulty in employing a Kalman filter lies in finding a system model that is ac-

curately enough (what can be said about the evolution of things when there is no extra information from sensors) and that fits into the Kalman equations. Besides that, the propagation of errors within the context of the system and measurement model has to be incorporated in the filter to function properly. While the Kalman filter is fairly forgiving concerning other a priori information not exactly known (e.g. the actual a priori noise distribution in the system and measurement models) the ratio of system and measurement model noise distribution magnitudes and the structure of the models are crucial to the use of the filter.

1.5 Road map

In chapter 2 of this report we will look at the properties and representation of rotational data, because it can occur to be an obstacle for convenient filtering practice and the reader may not be familiar with the representation of rotations[1] used further on. A method suggested by Lee and Shin (in [Lee2002]) which linearly filters rotations in their own space using the correspondence between angular and linear displacements is explored in chapter 3. In chapter 4 this method is used in the design of an actual filter and refinements are discussed that could counteract drawbacks of the filter. Chapter 5 presents experimental data to evaluate filter parameters and compare it with other methods. Chapter 6 summarizes the conclusions drawn throughout the report and lists recommendations for the use of presented orientation filters in the Personal Space Station.

[1] Quaternions were not addressed in Artificial Intelligence/Autonomous Systems geometry classes.

Chapter 2

Rotations and orientations

2.1 Rotational motion tracking

Near-field virtual and augmented reality applications such as the PSS measure the moving around and turning of *input devices* (tracked objects) whose location and pose influence the way virtual objects are presented to the user. To retrieve the location and pose of an object, several sensing techniques can be employed. Some applications combine positional information of points on tracked objects to estimate the motion parameters (e.g. using magnetic markers), others derive them instead (or additionally) from sensors in or on the object itself, using for instance accelerometers (e.g. in [Azuma2002]) and gyroscopes, earth magnetic field sensors or bending energy sensors. The Personal Space Station uses depth information from stereographic camera images to determine the location of multiple points per tracked object. Because the objects are required to be *rigid bodies*, which means they are solid and cannot be deformed (squeezed etc.), relative distances of points on the object are fixed. Changes in the 3D constellation of the points are thus completely determined by changes in location and pose of the object.

The result of any rigid body motion can be described as a combination of a *translation* and a *rotation* of the body. Translational and rotational parts of the motion signal are often processed separately, because the filtering techniques applied to translational data are not directly applicable to rotational signals. Rotations differ from translations because of the "looping" nature of rotations (e.g. the result of a 5 degree turn is normally indiscernible from the result of a 360+5 degree turn) and because combining rotations is non-linear (you cannot simply add them). Moreover, selecting a suitable data representation for rotations is not a trivial matter.

The first issue can be dealt with by ensuring that the rotational signal is conveniently constrained: Actual rotations can involve multiple revolutions of tracked objects to achieve the same observed orientation as could be reached in less than one revolution. It is common to forget about the multiplicity of the revolutions when tracking objects, not for the least because it complicates tracking algorithms. However, for the estimation of rotational velocity for instance, one should be careful not to neglect the possibility of multiple revolutions (because the number of revolutions an object performs within the sample interval should influence the measured velocity). Fortunately, when we are looking at orientation differences, if the rotational signal to be tracked does in practice have an upper bound on velocity of change and the interval between measurements is

small enough to accurately capture the maximum change within one revolution, taking multiple revolutions into account is not necessary. Because human operators of the PSS will use their hands to manipulate the input devices, it will be hard (and probably an indication of system malfunctioning) to encounter a complete revolution within the typical sampling interval of 1/25 sec.

This report is focused on the last two rotation issues: non-linearity and representation. We show that a (slight) change of perspective on the signal and the filter enables us to apply the same linear filtering technique used for translations to rotations. A data representation is chosen which also facilitates working with rotations.

The terms *orientation* and *rotation* will be used to distinguish static (more or less absolute) situations from dynamic (relative) events comparable with the distinction between *position* and *translation*. The orientation of an object can be characterized by the rotation that transforms points defined in the local object frame into points relative to a reference frame (often the "world"-frame) after the origins are made to coincide by translation. Thus a rotation is regarded as an *operator* and an orientation as a *value*: a rotation maps orientations to other orientations. The following sections will therefore generally speak of *rotations*, in 3-dimensional space, bearing in mind that orientations are the result of rotations.

When discussing motion signals (varying over time) it is often convenient to speak of *absolute* and *relative* orientations to discern between orientations described as (single) rotations relative to a fixed reference pose (the world-frame) and orientations described relative to the previous measured orientation.

2.2 The nature of rotations

A *rotation* turns an object about a *rotation axis* by a certain *angle* without deforming (sizing, shearing, mirroring or distorting the object's shape by other means) or displacing it: Rotation maps are invertible orientation-, angle- and norm-preserving orthogonal transformations. Elements of the group $SO(3)$ (the set of rotation matrices) share these properties with rotational operators (rotation maps) having 3 degrees of freedom and acting on points in 3-dimensional Euclidean space. $SO(n)$ is the orientation-preserving subgroup of the special orthogonal group, a subgroup of the norm-preserving general orthogonal group $GO(3)$, which in turn is a subgroup of general linear group $GL(n)$ characterized by the set of invertible square matrices. The general linear group is a member of the *affine* group to which translations also belong. Transformations within this group ensure that points that lie on one line, still lie on a straight line after the transform. Rigid body motion, or *Euclidean motion*, is an affine transformation which consists of a rotational and translational part.

Special orthogonal groups contain elements that are reversible under multiplication and preserve orientations, angles and norms when used as an operator and can thus be used to describe n-dimensional rotations. That rotation maps are reversible means that whichever rotation is performed, it will always be possible to undo (reverse) it by applying the *inverse* of the rotation. Angles and norms are preserved, which means that all angles in an object will be the same after a rotation (e.g. the "squareness" of a cube does not change) and that the object retains its size. An example of an angle-preserving mapping can be found in the painting "Prentententoonstelling" (see figure 2.1) by M.C. Escher. It shows what could happen if the requirement for rotation maps to be norm-preserving is dropped. That these relaxed constraints include mappings which only vaguely resemble rotations locally can be seen in figure 2.3, stressing that a truly

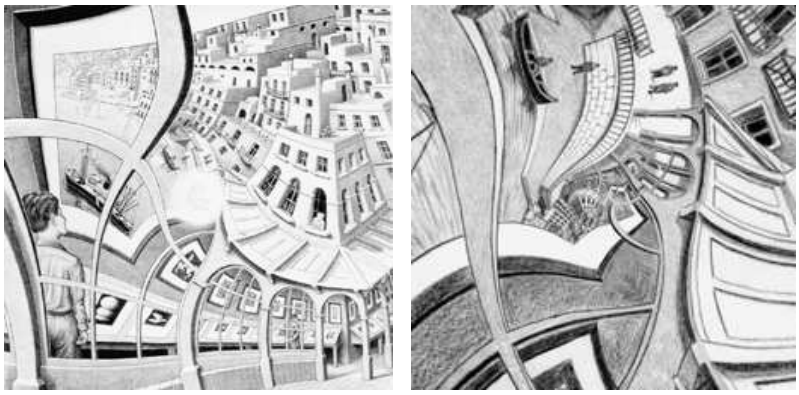


Figure 2.1: *The self-referring picture "Prententententoonstelling" (left) made by M.C. Escher in 1956 seems to have undergone an angle-, but not norm-preserving mapping. Researchers from Universiteit Leiden [DeSmit2003] discovered that by application of a rotated and scaled exponential mapping on a rectified version of the image as if it were the plane of complex numbers, a picture (right, detail) can be produced that very closely resembles Escher's distortion. Moreover, they managed to "mathemagically" fill in the center spot which Escher probably intentionally left blank.*

useful representation of rotations must include *all* constraining properties essential to rotation maps.

Rotations are considered to concatenate in a *multiplicative* manner, $R1 * R2$ is the rotation of $R1$ applied to rotation $R2$, which is the primary reason to stay away from applying methods for processing linear (e.g. translational) data to a rotational signal. When rotations are seen as the Lie group $SO(3)$ however, some "hidden" linearity in them can be identified using the exponential map, which maps *Lie algebras* to Lie groups. A Lie algebra can be thought of as the *tangent space* to the identity of its corresponding Lie group. Shifting from a Lie group to a Lie algebra means a reduction of multiplicativeness toward *additiveness*, because of the exponential relation between them. To help intuition with a lower dimensional example we can look at mapping elements from the tangent space $i\theta$ (where $\theta \in \mathbb{R}$) onto the unit circle $\mathbb{S}^1$ (figure 2.2 illustrates the meaning of the $\mathbb{S}^n$ notation) in the plane of the complex numbers $\mathbb{C}$ by the exponential function e^x. Starting from identity $(1,0)$ the mapped values neatly follow the curved perimeter of the circle. Adding a value to the variable θ in tangent space, means moving on the perimeter for a distance equal to the value, measured along the arc (see equation (2.1) and figure 2.4).

$$e^{i\theta}e^{i\varphi} = e^{i(\theta+\varphi)} \tag{2.1}$$

That the circle has something to do with the exponent can also be seen in equations (2.2), the MacLaurin-series (or Taylor-expansions about 0) of the exponent, sine and cosine functions. The summed absolute values of the sine and cosine coefficients form the expansion of the exponent function.

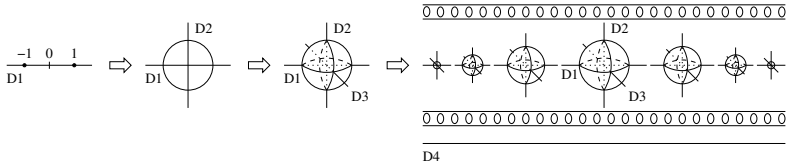


Figure 2.2: *From left to the right each arrow adds a dimension perpendicular to the dimensions of the previous sub-figure. The figures show unit spheres of increasing dimensionality: A unit sphere consists of all points at distance 1 to the origin and is mathematically denoted as* $\mathbb{S}^d$*, where* d *represents the number of dimensions* on *the surface of the sphere embedded in a* $d+1$ *dimensional space. The picture shows* $\mathbb{S}^0$*,* $\mathbb{S}^1$*,* $\mathbb{S}^2$ *and* $\mathbb{S}^3$*. Because it is not so easy to depict a 4 dimensional (hyper)sphere it is represented here as a film strip from -1 to 1 in the 4th dimension, to illustrate the same principle as building a 3D sphere from circular slices. Note that the distance-1-constraint is still respected. If the fourth dimension of a 4D sphere would be time and we would encounter it in our universe, we would see a small sphere (seemingly 3D, but it is only a 3D slice of the object) suddenly emerging out of the blue, growing bigger and bigger to eventually collapse again and disappear in the puff of logic.*

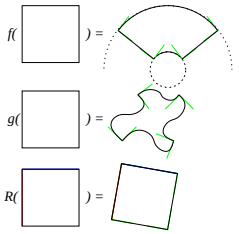


Figure 2.3: *Mappings* f *and* g *that do not preserve norms, and preserve angles locally, graphically compared to the rotation map* R*.*

$$
\begin{aligned}
e^x &= \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \tfrac{x^2}{2!} + \tfrac{x^3}{3!} + \tfrac{x^4}{4!} + \tfrac{x^5}{5!} + \tfrac{x^6}{6!} + \tfrac{x^7}{7!} + \dots \\
\sin(x) &= \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} = x - \tfrac{x^3}{3!} + \tfrac{x^5}{5!} - \tfrac{x^7}{7!} + \dots \\
\cos(x) &= \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} = 1 - \tfrac{x^2}{2!} + \tfrac{x^4}{4!} - \tfrac{x^6}{6!} + \dots
\end{aligned}
\tag{2.2}
$$

In a similar way Lie Algebras to $SO(3)$ resemble R^3although not as precise as is the case for $\mathbb{S}^1$: for $SO(3)$ the non-linearity does not entirely disappear when shifting to Lie algebra which is expressed by $\exp(u)\exp(v) = \exp(u+v+w)$ where $w = 0$ for $\mathbb{S}^1$ and $w \neq 0$ for $SO(3)$. The first (and largest) term of w is half the vector cross product of u and v. This means that for increasing magnitudes of u and v the error in assuming tangent space linearity can be expected to increase in proportion (because $(au) \times (bv) = ab(u \times v)$, with scalars a and b).[1]

[1] For a nice web resource on Lie groups and other mathematics, see [MathWorld- Lie Group].

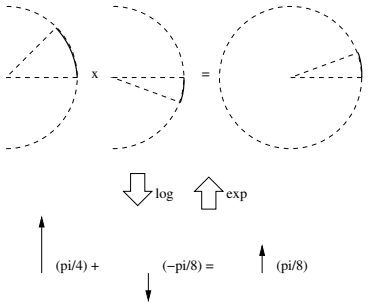


Figure 2.4: *Illustrating the equation* $e^{i\theta}e^{i\varphi} = e^{i(\theta+\varphi)}$.

2.3 The representation of rotations

3D-Rotations can be represented in several ways. Probably the most popular representation, often encountered in the fields of computer aided animation and aviation, are *Euler angles* [MathWorld- Euler Angles]: Any 3D-rotation can be described as consecutive (concatenated) rotations about three predefined (orthogonal) axes, relative to the object's orientation. This is a compact representation because 3 values describe a 3 degrees of freedom rotation, no variable is "wasted". One of the problems of Euler angles is, however, that there are many different conventions about the axes and the order in which the rotations about those axes have to be performed, which introduces a lot of opportunities for errors to sneak into calculations. Moreover, Euler angles suffer from a problem called *gimbal lock*: a rotation about one axis might align it with one of the other axes, which reduces the degrees of rotational freedom- information is lost (see figure 2.5). This is painfully illustrated by the navigational problems of space crafts using gyroscopes (consisting of three *gimbals*, two of which can align their rotation axes causing a *gimbal lock*) to determine changes in their attitude (pose). In an anecdote found at [NASA- Gimbal Lock] a lunar astronaut eventually jokes: "How about sending me a fourth gimbal for Christmas..." Surprisingly enough it turned out that adding a fourth (motor driven) redundant gimbal actually was a solution for this "attitude problem".

Simply adding an axis and angle did solve the mechanical problem, but it does not yield a convenient representation to calculate with. If one is not dependent on mechanical constructions like gyroscopes, the fourth variable could be used in a more sensible manner. A major theoretical issue of using Euler angles is that it is hard to do calculations with Euler angles on their own, because of the lack of a solid "Euler angle" arithmetic. Euler angles cannot simply be multiplied or added together: results would be only of use in the "neighborhood" of the rotations they was calculated from, because it would not take into account the strange interactions between the axes. Representing rotations using rotation matrices (for instance by incorporating the Euler angles into them) instead does introduce usable arithmetic to calculate with rotations. On the other hand, a rotation in matrix form uses 9 parameters for 3 degrees of freedom, which suffers from a lot of distracting interdependencies between the matrix elements.

Probably inspired by the use of Euler's formula $e^{ix} = \cos(x) + i\sin(x)$[2] on com-

[2]Can't help to mention the simply beautiful identity $e^{i\pi} + 1 = 0$, as found on the MathWorld pages [MathWorld- Euler Formula], which connects the fundamental numbers 0, 1, e, π, and i...

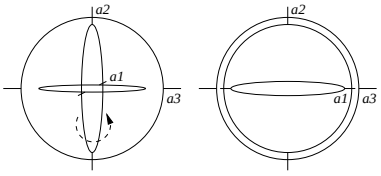


Figure 2.5: Gimbal lock *occurs when two rotation axes happen to coincide. This abstract representation of a gyroscope (with three* gimbals*, the rings in the picture) shows why this is a problem. Starting with the left picture, imagine that the middle gimbal is turning about axis* a2*. Whenever it aligns with the outer gimbal (see the picture to the right), axis* a1 *has become useless. The gyroscope cannot detect rotations about the axis perpendicular to* a2 *and* a3 *anymore. The problem actually increases gradually as axes are more coinciding:* Mechanically*, because forces have to "work harder" to get the gimbals rotating, and* computationally *because the representation of a pose gets more uncertain (due to numerical errors in small angle contributions to the resultant orientation data) as axes get increasingly ill-aligned.*

plex numbers for describing rotations in a plane, Hamilton (an Irish mathematician who lived from 1805-1865) developed the theory of *quaternions*, for which Euler's formula holds in the form $e^{Ix} = \cos(x) + I\sin(x)$, where I is the compound quaternion imaginary basis i, j, k (in contrast with imaginary basis i for complex numbers). Quaternions can compactly describe 3D rotations, using the representation of a single rotation about a single axis, effectively evaporating the concerns about the choice of axes and their order. Besides that, quaternion arithmetic with its quaternion product does take into account the interactions between rotations about single axes when they are concatenated.

2.4 Quaternions

Quaternions are numbers that have 4 components- a real part and an imaginary part consisting of 3 mutually orthogonal imaginary units i, j and k. Whereas quaternions can be regarded as 4-dimensional vectors, the complex number-like notation (2.3) may be preferred, because this representation takes explicitly into account the special imaginary arithmetic defined on it.

$$\mathbf{q} = w + xi + yj + zk \quad w,x,y,z \in \mathbb{R} \tag{2.3}$$

$$\begin{aligned} i^2 = j^2 = k^2 = ijk = -1, \\ ij = k, \qquad ji = -k, \\ jk = i, \qquad kj = -i, \\ ki = j, \qquad ik = -j. \end{aligned} \tag{2.4}$$

An alternative representation separates quaternions into its real part and a 3-dimensional imaginary vector. The description of the multiplication rules (2.4) in vector arithmetic (2.6) aims at the geometric interpretation of quaternions.

$$\mathbf{q} = w + xi + yj + zk = \left(w, \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right) = (w, \mathbf{u}) \quad w \in \mathbb{R}, \mathbf{u} \in \mathbb{R}^3 \tag{2.5}$$

$$\begin{aligned}\mathbf{q}_3 = \mathbf{q}_1\mathbf{q}_2 &= (w_1w_2 - \mathbf{u}_1\cdot\mathbf{u}_2, (w_1\mathbf{u}_2 + w_2\mathbf{u}_1 + \mathbf{u}_1\times\mathbf{u}_2)) \\ &= (1,i,j,k)\begin{pmatrix} w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2 \\ w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2 \\ w_1y_2 - x_1z_2 + y_1w_2 + z_1x_2 \\ w_1z_2 + x_1y_2 - y_1x_2 + z_1w_2 \end{pmatrix}\end{aligned} \tag{2.6}$$

These rules also define a *multiplicative identity* quaternion (2.7), the quaternion which does not change the value of an other quaternion when multiplied by it and vice versa. The *inverse* of a quaternion (2.8) is the quaternion which can cancel a quaternion out by a multiplication, resulting in the identity $\mathbf{q}\mathbf{q}^{-1} = \mathbf{q}/\mathbf{q} = 1$. Quaternions with length one (*unit quaternions*) have the pleasant property that their inverses are equal to their *conjugates* (2.9), which can simplify formulas and computations.[3]

$$\begin{gathered} 1 = (1,(0,0,0)) \\ (1,(0,0,0))\,(w,(x,y,z)) = (w,(x,y,z)) \\ (w,(x,y,z))\,(1,(0,0,0)) = (w,(x,y,z)) \end{gathered} \tag{2.7}$$

$$\begin{gathered} 1/\mathbf{q} = \mathbf{q}^{-1} = (w,(-x,-y,-z))\,/\left(w^2+x^2+y^2+z^2\right) \\ \mathbf{q}\mathbf{q}^{-1} = 1 \end{gathered} \tag{2.8}$$

$$\begin{gathered} \mathbf{q}' = (w,(-x,-y,-z)) \\ |\mathbf{q}| = \sqrt{w^2+x^2+y^2+z^2} \\ |\mathbf{q}| = 1 \rightarrow \mathbf{q}' = \mathbf{q}^{-1} \end{gathered} \tag{2.9}$$

Regarding the definition with imaginary axes and the formulas above quaternions seem to show a close similarity with complex numbers which can represent one-dimensional rotations in a two-dimensional plane. However, by comparing the rules for quaternion and complex number multiplication (2.10) it becomes clear that though similarities exist, there are also important differences.

$$\mathbf{z}_1 = a+bi, \quad \mathbf{z}_2 = c+di, \quad \mathbf{z}_3 = \mathbf{z}_1\mathbf{z}_2 = (ac-bd)+(ad+bc)\,i \tag{2.10}$$

(ab)	$b=1$	$b=i$	$b=j$	$b=k$
$a=1$	1	i	j	k
$a=i$	i	-1	k	$-j$
$a=j$	j	$-k$	-1	i
$a=k$	k	j	$-i$	-1

Table 2.1: *Tabular view on the quaternion multiplication rules. The complex number product involves only the four top left cells. The quaternion product extends the interdependency of the axes.*

Quaternion multiplication introduces a cross product term, or "mingling of the axes", which renders the product non-commutative, as opposed to the complex-product. This corresponds with the fact that arbitrary rotations are commutative in 2D, but not in 3D (the order of applying rotations influences the result). Whereas the coordinate

[3] The inverse involves both multiplication and division, where the conjugate only switches the sign of the (imaginary) vector part of a quaternion, a very cheap operation in terms of computing power.

axes of a translation are completely independent from each other, coordinate axes do depend on each other during (successive) rotations, in the typical way reflected by the quaternion product rules. For instance a rotation in 3D of 180 degrees about an x-axis $(1,0,0)$ followed by a rotation of 180 degrees about an y-axis $(0,1,0)$, represented by the quaternions $(0,(1,0,0))$ and $(0,(0,1,0))$ respectively, can be described as a single rotation of 180 degrees about the z-axis, coinciding with the result of the quaternion multiplication $(0,(1,0,0))(0,(0,1,0)) = (0,(0,0,1))$ (see table 2.1). In 2D, rotations can only be performed about a single axis (the one perpendicular to the plane in which the rotation is performed), which explains why rotation in 2D is commutative in spite of the fact that general rotation is not.

Quaternions are used to represent three-dimensional rotations in the sense that when $\mathbf{n}$ is the axis (of unit length) and θ is the angle of any rotation map $R_{\mathbf{n},\theta}$ acting on vector $\mathbf{v}$, the quaternion $\mathbf{q}_{\mathbf{n},\theta}$ that describes the rotation yielding $\mathbf{v}_{rot}$ is constructed as follows:

$$\begin{array}{c} R_{\mathbf{n},\theta} \in SO(3) \\ \mathbf{v}_{rot} = R_{\mathbf{n},\theta}(\mathbf{v}) \end{array} \quad \begin{array}{c} \left(\theta \in \mathbb{R}, \mathbf{n} \in \mathbb{S}^2\right) \\ \leftrightsquigarrow \end{array} \quad \begin{array}{c} \mathbf{q}_{\mathbf{n},\theta} = \left(\cos\frac{\theta}{2}, \mathbf{n}\sin\frac{\theta}{2}\right) \in \mathbb{S}^3 \\ \mathbf{p} = (0,\mathbf{v}) \\ (0,\mathbf{v}_{rot}) = \mathbf{q}_{\mathbf{n},\theta}\mathbf{p}\mathbf{q}_{\mathbf{n},\theta}^{-1} = \mathbf{q}_{\mathbf{n},\theta}\mathbf{p}\mathbf{q}'_{\mathbf{n},\theta} \end{array} \tag{2.11}$$

This formula states that only quaternions that have unit length correspond with a proper rotation, because $\mathbb{S}^3$ is the 4D unit hypersphere which consists of all four-dimensional vectors that have length one. Furthermore the formula shows a two-to-one correspondence, called the *anti-podal equivalence*, between $SO(3)$ and $\mathbb{S}^3$: One can always find two points on the unit-quaternion sphere that represent the same rotation, $\mathbf{q}$ and $-\mathbf{q}$, because sign changes are canceled out by the double multiplication.

Quaternions of arbitrary length (greater than zero of course) can be used as representations for rotations; magnitude evens out in the formula. However, it is still more convenient to use unit-quaternions, because a fixed length eliminates the redundant scale factor (all quaternions that "point in the same direction" regardless of their length, correspond to the same rotation) and using length one keeps the fit between $SO(3)$ and quaternions tight and elegant (because $\mathbb{S}^3$ is a "basic" manifold and $\mathbb{R}^3$with the cross product is a Lie algebra to it, isomorphic with the algebra to $SO(3)$).

That $(0,\mathbf{v}_{rot}) = \mathbf{q}_{\mathbf{n},\theta}(0,\mathbf{v})\mathbf{q}_{\mathbf{n},\theta}^{-1}$ performs a rotation can be proven by noting that quaternion multiplication preserves norms, scalar multiplication commutes and a continuous path from identity to every possible action exists, which excludes reflections.

To show how the quaternion product works geometrically [Berkeley- CS184] we first note that a vector which is to be rotated about $\mathbf{n}$ can be decomposed into $\mathbf{v} = \mathbf{v}_{\parallel} + \mathbf{v}_{\perp}$, separating the part $\mathbf{v}_{\parallel}$ parallel and the part $\mathbf{v}_{\perp}$ perpendicular to axis of rotation $\mathbf{n}$. Using a third vector $\mathbf{v}_{\times} = \mathbf{n} \times \mathbf{v} = \mathbf{n} \times \mathbf{v}_{\perp}$, perpendicular to both $\mathbf{n}$ and $\mathbf{v}$, the rotated version $\mathbf{v}_{rot}$ of $\mathbf{v}$ can be expressed in a familiar-looking sine-cosine form (resembling the construction of a circle):

$$\begin{array}{c} e^{\hat{\mathbf{n}}\theta} = \mathbf{1} + \hat{\mathbf{n}}\sin(\theta) + \hat{\mathbf{n}}^2(1-\cos\theta) \\ \hat{\mathbf{n}} = \begin{bmatrix} 0 & -\mathbf{n}_3 & \mathbf{n}_2 \\ \mathbf{n}_3 & 0 & -\mathbf{n}_1 \\ -\mathbf{n}_2 & \mathbf{n}_1 & 0 \end{bmatrix} \end{array} \tag{2.12}$$

$$\mathbf{v}_{rot} = \mathbf{v}_{\parallel} + \mathbf{v}_{\perp}\cos\theta + \mathbf{v}_{\times}\sin\theta \tag{2.13}$$

The projection of the vector on the plane perpendicular to the axis of rotation is rotated in that same plane over the rotation angle. The plane intersects the rotation axis

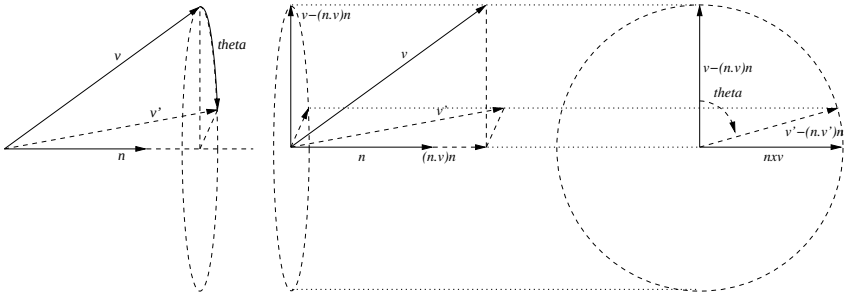


Figure 2.6: *Quaternion rotation: Vector* $\mathbf{v}'$ *is vector* $\mathbf{v}$ *rotated by angle theta about axis* $\mathbf{n}$*. The part of* $\mathbf{v}$ *parallel to* $\mathbf{n}$ *is* $\mathbf{v}_{\parallel} = (\mathbf{n}\cdot\mathbf{v})\,\mathbf{n}$*, the part perpendicular to* $\mathbf{n}$ *is* $\mathbf{v}_{\perp} = \mathbf{v} - \mathbf{v}_{\parallel} = \mathbf{v} - (\mathbf{n}\cdot\mathbf{v})\,\mathbf{n}$*.*

at the origin, so in order to get the right rotated vector the rotated projection must be shifted over $\mathbf{v}_{\parallel}$, the component of the original vector parallel to the axis of rotation.

One characteristic feature of rotations is that all points that lie on the axis of rotation remain unchanged: In quaternion terms this means that $\mathbf{q}\left(0,\mathbf{v}_{\parallel}\right)\mathbf{q}^{-1} = \left(0,\mathbf{v}_{\parallel}\right)$ which is the case if $\left(0,\mathbf{v}_{\parallel}\right)$ would commute with $\mathbf{q}^{-1}$, because then $\mathbf{q}\left(0,\mathbf{v}_{\parallel}\right)\mathbf{q}^{-1} = \mathbf{q}\mathbf{q}^{-1}\left(0,\mathbf{v}_{\parallel}\right) = \left(0,\mathbf{v}_{\parallel}\right)$. The cross product ingredient in quaternion multiplication is the cause of non-commutativeness, so $\mathbf{v}_{\parallel}\times\mathbf{n} = \mathbf{n}\times\mathbf{v}_{\parallel}$ should be true. It is indeed, because $\mathbf{v}_{\parallel}$ and $\mathbf{n}$ are parallel to each other and applying the cross product in either way would always result in 0.

$$\mathbf{q}(\left(0,\mathbf{v}_{\parallel}\right) + (0,\mathbf{v}_{\perp}))\mathbf{q}^{-1} = \mathbf{q}\left(0,\mathbf{v}_{\parallel}\right)\mathbf{q}^{-1} + \mathbf{q}\,(0,\mathbf{v}_{\perp})\,\mathbf{q}^{-1} \tag{2.14}$$

Then if the rotation can be described by (2.14) and $\mathbf{q}\left(0,\mathbf{v}_{\parallel}\right)\mathbf{q}^{-1} = \left(0,\mathbf{v}_{\parallel}\right)$ it remains to be explained that $\mathbf{q}\,(0,\mathbf{v}_{\perp})\,\mathbf{q}^{-1}$ is equal to $(0,(\mathbf{v}_{\perp}\cos\theta + \mathbf{v}_{\times}\sin\theta))$. With the help of goniometry identities and properties of dot- and cross-product it can be shown that this is the case:

$$\begin{aligned}
\mathbf{q}(0,\mathbf{v}_{\perp})\mathbf{q}^{-1} &= \left(\left(\cos\tfrac{\theta}{2},\mathbf{n}\sin\tfrac{\theta}{2}\right)(0,\mathbf{v}_{\perp})\right)\mathbf{q}^{-1}\\
&= \left(-(\mathbf{n}\cdot\mathbf{v}_{\perp})\sin\tfrac{\theta}{2},\mathbf{v}_{\perp}\cos\tfrac{\theta}{2} + (\mathbf{n}\times\mathbf{v}_{\perp})\sin\tfrac{\theta}{2}\right)\mathbf{q}'\\
&= \left(0,\mathbf{v}_{\perp}\cos\tfrac{\theta}{2} + \mathbf{v}_{\times}\sin\tfrac{\theta}{2}\right)\left(\cos\tfrac{\theta}{2},-\mathbf{n}\sin\tfrac{\theta}{2}\right)\\
&= \left(-\mathbf{n}\cdot\left(\mathbf{v}_{\perp}\cos\tfrac{\theta}{2} + \mathbf{v}_{\times}\sin\tfrac{\theta}{2}\right)\sin\tfrac{\theta}{2},\cos\tfrac{\theta}{2}\left(\mathbf{v}_{\perp}\cos\tfrac{\theta}{2} + \mathbf{v}_{\times}\sin\tfrac{\theta}{2}\right)\right.\\
&\qquad \left. + \left(\left(\mathbf{v}_{\perp}\cos\tfrac{\theta}{2} + \mathbf{v}_{\times}\sin\tfrac{\theta}{2}\right)\times\left(-\mathbf{n}\sin\tfrac{\theta}{2}\right)\right)\right)\\
&= \left(\sin\tfrac{\theta}{2}\cos\tfrac{\theta}{2}(\mathbf{v}_{\perp}\cdot\mathbf{n}) + \sin^2\tfrac{\theta}{2}(\mathbf{v}_{\times}\cdot\mathbf{n}),\mathbf{v}_{\perp}\cos^2\tfrac{\theta}{2} + \mathbf{v}_{\times}\cos\tfrac{\theta}{2}\sin\tfrac{\theta}{2}\right.\\
&\qquad \left. - \left(\left(\mathbf{v}_{\perp}\cos\tfrac{\theta\mathbf{q}_i}{2}\right)\times\mathbf{n} + \left(\mathbf{v}_{\times}\sin\tfrac{\theta}{2}\right)\times\mathbf{n}\right)\sin\tfrac{\theta}{2}\right)\\
&= \left(0,\mathbf{v}_{\perp}\cos^2\tfrac{\theta}{2} + \mathbf{v}_{\times}\cos\tfrac{\theta}{2}\sin\tfrac{\theta}{2} - \mathbf{v}_{\perp}\times\mathbf{n}\cos\tfrac{\theta}{2}\sin\tfrac{\theta}{2} - \mathbf{v}_{\times}\times\mathbf{n}\sin^2\tfrac{\theta}{2}\right)\\
&= \left(0,\mathbf{v}_{\perp}\cos^2\tfrac{\theta}{2} + \mathbf{v}_{\times}\cos\tfrac{\theta}{2}\sin\tfrac{\theta}{2} + \mathbf{v}_{\times}\cos\tfrac{\theta}{2}\sin\tfrac{\theta}{2} - \mathbf{v}_{\perp}\sin^2\tfrac{\theta}{2}\right)\\
&= \left(0,\mathbf{v}_{\perp}\left(\cos^2\tfrac{\theta}{2} - \sin^2\tfrac{\theta}{2}\right) + 2\mathbf{v}_{\times}\left(\cos\tfrac{\theta}{2}\sin\tfrac{\theta}{2}\right)\right)\\
&= (0,(\mathbf{v}_{\perp}\cos\theta + \mathbf{v}_{\times}\sin\theta))
\end{aligned}$$

We conclude this overview of quaternion arithmetic with the quaternion exponential map, defined and used in a way similar to its complex number counterpart (2.15):

$$e^{\mathbf{q}} = e^{w+I\mathbf{v}} = e^{(w,\mathbf{v})} = e^{w}\left(\cos\|\mathbf{v}\|,\frac{\mathbf{v}}{\|\mathbf{v}\|}\sin\|\mathbf{v}\|\right) \tag{2.15}$$

From a signal processing point of view ([Dooijes1999, section 3.3]) $f(t) = xe^{i\Omega t} = x(\cos(\Omega t) + i\sin(\Omega t))$ describes a periodic signal f with amplitude x and angular velocity Ω (corresponding with a frequency of $\Omega/2\pi$ Hz provided that t is measured in seconds). If such a signal has unit amplitude then the logarithm of the signal value $\tilde{f}(t)$ yields the angular displacement (distance) of the signal relative to where the signal "started" ($t = 0$), at the complex identity $e^{i0} = 1 + 0i$. If the angular displacements are measured relatively[4] from one value to the next instead from identity, they equal the instantaneous angular velocity (rate of change) of the signal, under the assumption that velocity remains constant during the time interval between two measurements.

When represented by an ordered set of quaternions, the values of an orientation signal lie on the 4D unit hyper-sphere, because we use unit quaternions to represent the orientation values, just as the values of $\tilde{f}(t)$ are bound to the unit circle: our rotation signal also has "unit amplitude". The logarithms of these unit quaternions produce their corresponding angular displacements from the identity quaternion $(1,(0,0,0))$. Note that because of the higher order entanglement of the quaternion axes $\log(\mathbf{q}(t+dt)) - \log(\mathbf{q}(t)) \neq \log(\mathbf{q}(t+dt)/\mathbf{q}(t))$. As mentioned in section 2.2 taking the logarithm means moving from the Lie-group manifold to its tangent space, its Lie-algebra. This is compatible with the notion of signal velocity when applied to relative orientations. If the changing pose of an object is represented by quaternions $\{\mathbf{q}_0, \mathbf{q}_1, \ldots \mathbf{q}_i, \ldots \mathbf{q}_n\}$ that rotate from a fixed reference pose to the orientation of the object at increasing time instants $t(i)$, then the instantaneous magnitude and direction of the object's average angular velocity between time instant $t(i)$ and $t(i+1)$ is equal to (mind the half angle correspondence) twice the logarithm of the rotation from $\mathbf{q}_i$ to $\mathbf{q}_{i+1}$, $\log\left(\mathbf{q}_i^{-1}\mathbf{q}_{i+1}\right)$, the magnitude and direction of the tangent to the sphere emerging from $\mathbf{q}_i$ and pointing towards $\mathbf{q}_{i+1}$.

[4] $\left(\log\left(\tilde{f}(t+dt)\right) - \log\left(\tilde{f}(t)\right) = \log\left(\tilde{f}(t+dt)/\tilde{f}(t)\right)\right)$

Chapter 3

Linear filtering of orientation data

3.1 Key ideas

The orientation filtering scheme proposed by Lee and Shin in [Lee2002] is based on three key ideas. The first, *linearization*, is to identify an additive property of the orientation signal that can be associated with a linear signal, which in turn can be filtered with a Finite Impulse Response (FIR) linear filter. The association itself is achieved by the second idea: *offset filtering*. Instead of filtering absolute values, the filter *gain* on data *relative to the sample in the center of the filter mask* is computed. The third idea is to *rewrite filter masks* in a way that makes offset filtering computationally more efficient.

3.1.1 Linearization

For a (maximally) meaningful result of applying a linear filter, the nature of the filtered signal should be additive. A rotational signal however, does not behave additively, but multiplicatively. This can be seen by examining the relation between successive signal values: As opposed to the linear motion (translational) profile (3.1) of an object, the (discrete) evolution of its orientation $\mathbf{q}_i$ is not described by sequentially adding orientation differences but by iterative multiplication of the orientation value with relative orientations, using $\mathbf{q}_0$ as a starting point (3.2).

$$\begin{aligned} \mathbf{p}_i &= \mathbf{p}_0 + (\mathbf{p}_1 - \mathbf{p}_0) + \cdots + (\mathbf{p}_i - \mathbf{p}_{i-1}) \\ &= \mathbf{p}_0 + \sum_{j=0}^{i-1} (\mathbf{p}_{j+1} - \mathbf{p}_j) \end{aligned} \tag{3.1}$$

$$\begin{aligned} \mathbf{q}_i &= \mathbf{q}_0 \left(\mathbf{q}_0^{-1}\mathbf{q}_1\right) \left(\mathbf{q}_1^{-1}\mathbf{q}_2\right) \cdots \left(\mathbf{q}_{i-1}^{-1}\mathbf{q}_i\right) \\ &= \mathbf{q}_0 \prod_{j=0}^{i-1} \left(\mathbf{q}_j^{-1}\mathbf{q}_{j+1}\right) \end{aligned} \tag{3.2}$$

From the explanations in section 2.2 and 2.4 it can be concluded that there is actually an additive property “hiding” in the rotational signal: its tangent. The logarithm

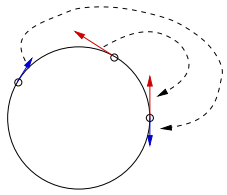


Figure 3.1: *An approximation of the instantaneous velocity* ω_j *requires the extraction of the tangent* $\log\left(\mathbf{q}_j^{-1}\mathbf{q}_{j+1}\right)$ *between two successive orientations* $\mathbf{q}_j$ *and* $\mathbf{q}_{j+1}$.

of a rotation yields a vector in a nearly linear tangent space attached to rotational (multiplicative) identity indicating *angular displacement*, the distance and direction of the travel in rotation space corresponding with the rotation. This property can be exploited to express both translational and rotational signals in terms of displacement vectors with linear behaviour:

$$\begin{aligned} w_j &= -\mathbf{p}_j + \mathbf{p}_{j+1} \\ \mathbf{p}_i &= \mathbf{p}_0 + \sum_{j=0}^{i-1} w_j \\ \omega_j &= \log\left(\mathbf{q}_j^{-1}\mathbf{q}_{j+1}\right) \\ \mathbf{q}_i &= \mathbf{q}_0 \prod_{j=0}^{i-1} e^{\omega_j} \end{aligned} \tag{3.3}$$

The magnitude of the displacement vector ω_j corresponds to the *slerp-* or *spherical quaternion distance* travelled in one time step. If samples are taken equidistant in time (the sampling rate is constant), ω_j can be interpreted as discrete velocity vector. The anti-podal rotational equivalence of unit quaternions causes the velocity measured in quaternion space to be twice as slow as observed in the rotation it represents: a rotation over an angle θ corresponds with traveling over a distance of $\frac{1}{2}\theta$ along the surface of the quaternion sphere.

One of the ideas of the orientation filtering scheme is to couple the multiplicative angular signal of relative orientations with the corresponding additive linear motion profile using the linearity of the signal tangents. Displacement ω_j can be interpreted as a linear vector, as if it were the linear displacement of additive motion profile $\mathbf{p}$, for which $\omega_j = \mathbf{p}_{j+1} - \mathbf{p}_j$. This leads to a transform (3.4) between the multiplicative signal $\mathbf{q}$ to additive signal $\mathbf{p}$, but note that $e^{\mathbf{p_i}} \neq \mathbf{q}_i$ because of the not entirely linear nature of the algebra to the quaternion unit sphere.

$$\begin{aligned} \mathbf{p}_i &= \mathbf{p}_0 + \sum_{j=0}^{i-1} \log\left(\mathbf{q}_i^{-1}\mathbf{q}_{i+1}\right) &&= \mathbf{p}_0 + \sum_{j=0}^{i-1} \omega_j \\ \mathbf{q}_i &= \mathbf{q}_0 \prod_{j=0}^{i-1} e^{\left(\mathbf{p}_{j+1} - \mathbf{p}_j\right)} &&= \mathbf{q}_0 \prod_{j=0}^{i-1} e^{\omega_j} \end{aligned} \tag{3.4}$$

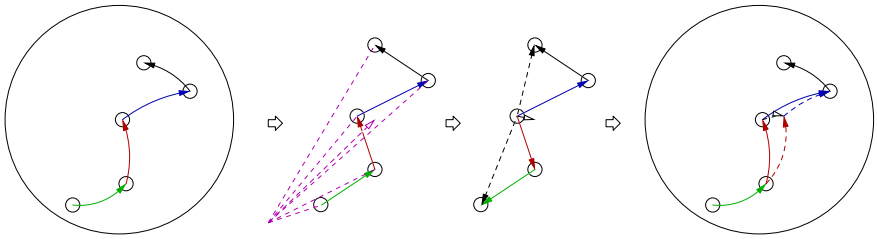


Figure 3.2: *The Lee/Shin scheme first maps relative rotations represented in unit-quaternion space ($\mathbb{S}^3$) into the vector space of $\mathbb{R}^3$. In this space the filter mask is applied. The convolution mask is constructed to produce the (linear) filter gain on the data sample in the middle of the filtering window. The gain is in turn mapped back to $\mathbb{S}^3$. Filtering relative orientations ensures coordinate-invariance and reduces the risk of encountering singularities in the inverse exponential mapping onto $\mathbb{S}^3$.*

3.1.2 Offset filtering

When the signal $\mathbf{p}$ is filtered by a linear filter F signal values $\mathbf{p}_i$ are displaced by *filter gain* $F(\mathbf{p}_i)-\mathbf{p}_i$ (3.5). Just like linear displacement, this gain can be transformed to a rotational gain which can be applied to the rotational signal, yielding a rotational filter (3.6) based on F: a filtered orientation is the original measurement rotated over an angular distance equal to the filter gain of F. Note that the filter gain is computed from the direct neighbourhood of the sample in the center of the filter mask and that it is the (local) gain only which couples the linear and angular signals, which respects the locality of rotational tangent space.

$$\begin{aligned} F(\mathbf{p}_i) &= a_{-k}\mathbf{p}_{i-k}+\cdots+a_0\mathbf{p}_i+\cdots+a_k\mathbf{p}_{i+k} \\ &= \sum_{m=-k}^{m=k} a_m\mathbf{p}_{i+m} \\ &= \mathbf{p}_i+(F(\mathbf{p}_i)-\mathbf{p}_i) \\ F(\mathbf{p}_i)-\mathbf{p}_i &= \left(\left(\sum_{m=-k}^{m=k} a_m\mathbf{p}_{i+m}\right)-\mathbf{p}_i\right) \end{aligned} \tag{3.5}$$

$$H(\mathbf{q}_i) = \mathbf{q}_i e^{(F(\mathbf{p}_i)-\mathbf{p}_i)} \tag{3.6}$$

If we demand that the filter coefficients $a_{-k}\ldots a_k$ sum to one, which makes filter F an *affine-invariant* filter, we can easily rewrite the filter to act on values relative to the value of the center sample instead of to the origin, so that both input values and filter gain appear in the same space originating in the center of the mask:

$$\sum_{m=-k}^{m=k} a_m = 1$$

$$\begin{aligned} F(\mathbf{p}_i) &= \mathbf{p}_i + \left(\left(\sum_{m=-k}^{m=k} a_m \mathbf{p}_{i+m} \right) - \mathbf{p}_i \right) \\ &= \mathbf{p}_i + \left(\left(\sum_{m=-k}^{m=k} a_m \mathbf{p}_{i+m} \right) - \left(\sum_{m=-k}^{m=k} a_m \mathbf{p}_i \right) \right) \\ &= \mathbf{p}_i + \left(\sum_{m=-k}^{m=k} a_m \left(\mathbf{p}_{i+m} - \mathbf{p}_i \right) \right) \end{aligned} \tag{3.7}$$

$$H(\mathbf{q}_i) = \mathbf{q}_i \exp \left(\sum_{m=-k}^{m=k} a_m \left(\mathbf{p}_{i+m} - \mathbf{p}_i \right) \right) \tag{3.8}$$

3.1.3 An efficient offset filtering mask

Because $\mathbf{p}_{i+1} - \mathbf{p}_i = \log\left(\mathbf{q}_i^{-1}\mathbf{q}_{i+1}\right)$ but in general $\mathbf{p}_{i+m} - \mathbf{p}_i$ is *not equal* to $\log\left(\mathbf{q}_i^{-1}\mathbf{q}_{i+m}\right)$, expressing equation (3.8) in angular terms involves the summation of small displacements:

$$\begin{aligned} \mathbf{p}_{i+m} - \mathbf{p}_i &= \begin{cases} \sum_{j=i}^{i+m-1} \mathbf{p}_{j+1} - \mathbf{p}_j, & m \geq 1 \\ 0, & m = 0 \\ \sum_{j=i-m}^{i-1} \mathbf{p}_{j+1} - \mathbf{p}_j, & m \leq -1 \end{cases} \\ &= \begin{cases} \sum_{j=i}^{i+m-1} \log\left(\mathbf{q}_j^{-1}\mathbf{q}_{j+1}\right), & m \geq 1 \\ 0, & m = 0 \\ \sum_{j=i-m}^{i-1} \log\left(\mathbf{q}_j^{-1}\mathbf{q}_{j+1}\right), & m \leq -1 \end{cases} \end{aligned} \tag{3.9}$$

The filter kernel $[a_{-k}, \ldots a_0, \ldots a_k]$ can be rewritten in such a way that it incorporates these sums:

$$b_m = \begin{cases} \sum_{j=m+1}^{k} a_j, & 0 \leq m \leq k-1 \\ \sum_{j=-k}^{m} -a_j, & -k \leq m < 0 \end{cases} \tag{3.10}$$

With mask $[b_{-k}, \ldots b_0, \ldots b_{k-1}]$, derived from $[a_{-k}, \ldots a_0, \ldots a_k]$, an efficient filter can be formulated for use in situations where only relative data is available or is only trusted to behave linearly in tangent space:

$$\begin{aligned} F(\mathbf{p}_i) &= \mathbf{p}_i + \left(\sum_{m=-k}^{k-1} b_m w_{i+m} \right) \\ w_j &= -\mathbf{p}_i + \mathbf{p}_{i+1} \end{aligned} \tag{3.11}$$

$$\begin{aligned} H(\mathbf{q}_i) &= \mathbf{q}_i \exp \left(\sum_{m=-k}^{k-1} b_m \omega_{i+m} \right) \\ \omega_j &= \log\left(\mathbf{q}_j^{-1}\mathbf{q}_{j+1}\right) \end{aligned} \tag{3.12}$$

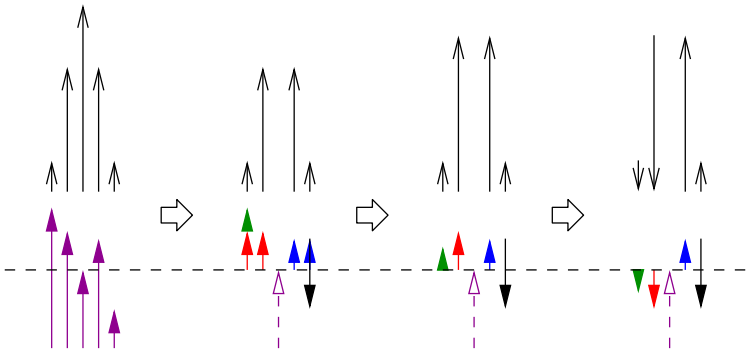


Figure 3.3: *This diagram explains how a convolution kernel is to be modified so that it can be applied to relative instead of absolute data. It illustrates that the sums of displacements from equation (3.9) can be incorporated into the filter kernel. The lower sets of arrows represent input data, the upper sets smoothing convolution kernels (as an example). The first (left-most) sets of arrows can be regarded as convolution kernel* $[a_{-2}, a_{-1}, a_0, a_1, a_2]$ *and data values* $\mathbf{p}_1..\mathbf{p}_5$. *The second as* $[a_{-2}, a_{-1}, a_1, a_2]$ *and* $[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4, \mathbf{p}_5] - \mathbf{p}_3$. *The third as* $[a_{-2}, a_{-1} + a_{-2}, a_1 + a_2, a_2]$ *and* $[\mathbf{p}_1 - \mathbf{p}_2, \mathbf{p}_2 - \mathbf{p}_3, \mathbf{p}_4 - \mathbf{p}_3, \mathbf{p}_5 - \mathbf{p}_4]$. *The last (rightmost) sets of arrows represent* $[-a_{-2}, -(a_{-1} + a_{-2}), a_1 + a_2, a_2]$ *and* $[\mathbf{p}_2 - \mathbf{p}_1, \mathbf{p}_3 - \mathbf{p}_2, \mathbf{p}_4 - \mathbf{p}_3, \mathbf{p}_5 - \mathbf{p}_4]$. *The left-most convolution yields absolute results (i.e. in the frame of the arrows in the lower set), whereas the other convolutions generate a filter gain on the center sample (the dashed arrows). The right-most operation only uses input data that is represented relative to directly previous data.*

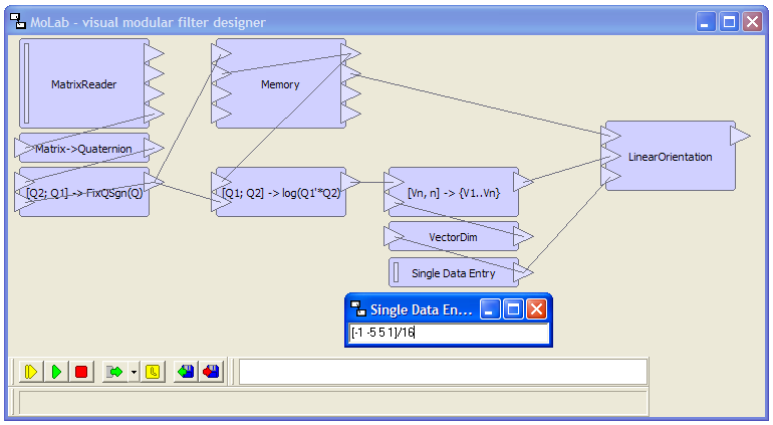


Figure 3.4: *The MoLab visual algorithm builder.*

3.2 Implementation

The orientation filter was implemented in three application environments: in MatLab, a Borland Delphi application and the PSS itself. MatLab functions were used to implement quaternion functions and the filter algorithm, to plot filter and signal properties and to evaluate the filter quantitatively. In a Borland Delphi application, called "MoLab" the components of the orientation filter were implemented in a modular fashion using the Object Pascal programming language (see figure 3.4). The graphical user interface in MoLab represents the components as pluggable (sub-)filters with typed data inputs and outputs. It was used to get familiar with quaternions and the filter operations outside the MatLab environment. The source code to the subset of MoLab filters relevant for on-line filtering was translated into C-code and integrated into the PSS architecture (by Robert van Liere). Additionally an OpenGL application, called "MoVis" was built and used to read in motion data files and visualize the data streams in real-time by moving and rotating cubes, offering the opportunity to monitor the filter's visual performance in real-time on a standard PC (independent from the PSS see figure 3.5).

The general scheme of filtering incoming samples is as follows:

- Initialization
  - Initialize the convolution kernel, allocate and clear a quaternion and a displacement history, which are first-in first-out shift registers (FIFO buffers). Displacement history is required to span the width of the used convolution kernel, whereas quaternion history has to be one step longer than half the width (the quaternion in the center of filter support is used in the calculation).
- Calculation step
  - A sample is received from the PSS optical object tracker in the form of a homogeneous matrix or is extracted by MoLab from a file containing captured motion data (created by the PSS).

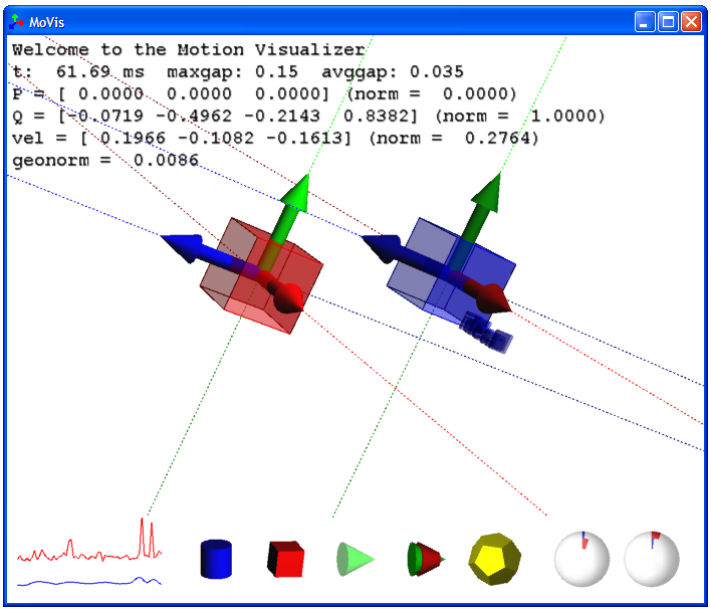


Figure 3.5: *The MoVis motion visualization tool.*

- The top left 3x3 elements of this matrix carry the rotational information, from which a corresponding unit- quaternion is computed.
- Because of the anti-podal equivalence of quaternions, a new quaternion value is negated if its angular distance to the previous value is more than $\pi/2$ along the unit-quaternion sphere (assuming the difference between consecutive orientations to be less than $2(\pi/2)$ radians, i.e. 180 degrees). The value is appended to the quaternion history. This check can be performed using the quaternion logarithm, but as it's just a check against an upper bound, it is probably more efficient to measure the raw Euclidean distance between quaternions (regarding them as 4D linear vectors) suffices - it has to stay below $\sqrt{2}$ (or only the sum of squared quaternion components below 2).
- The angular displacement between the last two quaternions in history is computed and appended to displacement history.
- The convolution kernel is applied to displacement history, which yields a *gain* on the oldest orientation in quaternion history.
- The exponent of this displacement generates a small rotation relative to the quaternion in the center of filter support, with which it is multiplied.
- The resulting orientation is reported back to the PSS architecture or an OpenGL renderer by converting it into matrix-form.

3.3 Discussion

It would be nice if a linear -for instance smoothing- filter could be directly applied to the unit-quaternion data. A solution of this kind would already have no problems with gimbal lock. However, the renormalization problem mentioned in section 1.4, besides incorrectly speeding up and down rotations, brings forth a rather troublesome singularity when the filtered quaternions are symmetrically distributed over the sphere: The more symmetric a constellation of quaternions, the more they cancel each other out; which leads to increasing numerical errors and eventually a zero signal, which does not have a direction, causing the normalization to fail. Although Lee and Shin use this as an argument to justify the choice for the more intricate method, it should be noted that if quaternions are kept tightly together (within angular distance $\pi/2$, compensating for anti-podal equivalence) the degenerate case mentioned does not occur. Detecting whether two quaternions are too far apart can be done without knowledge about the exponential map nor slerp. Moreover, the trade-off between sampling frequency and numerical stability is becoming less and less awkward as capture speed gets cheaper, which means that sampling with current technology is dense enough not to break the implied angular speed limits.

We have seen that quaternions can be linearized by taking their logarithms. Then if we cannot filter the raw quaternion coordinates, we could transform quaternions directly into logarithmic space and apply our linear filter on the resulting values. This eliminates the need of renormalisation, because working in log-space means staying on the unit sphere; so the zero-singularity is gone too. Despite these advantages, an issue remains: such filters would not be coordinate-invariant because the sum of quaternion logarithms is not equal to the logarithm of the (product) concatenation of those quaternions (see section 3.1.3).

The orientation filter proposed by Lee and Shin has some desirable features: It is flexible, computationally efficient and coordinate- and time-invariant (because filter support is finite). The filter applies a convolution mask on the (transformed rotational) input data, which is a widely used linear filtering technique. It is flexible because by changing the mask the behaviour of the convolution can adopt a wide range of functionality, among others smoothing and sharpening. Convolution involves computation of a linear weighted sum, it is thus efficiently and easily implemented using additions and multiplications (with mask size N, N multiplications and N additions are required). The filter operates on (transformed) relative rotations, measured from the body frame rather than from the world frame. Because of this the filter is coordinate-invariant; the response of the filter does not depend on where the world-frame is positioned and how it is oriented. The filter is also time-invariant, which means that it does not matter (it does not change the result) whether a (motion) signal appears earlier or later in time.

The Lee and Shin FIR filter computes a filter gain relative to that sample based on angular differences between successive samples. The exponent of the filter gain is then applied to the original orientation, which in effect “displaces” the orientation, staying on the surface of the unit quaternion sphere. The filtered values only depend on samples in the neighbourhood (under the mask) of the current sample and not on all previous samples. To be able to apply a convolution mask on rotational input data, Lee and Shin exploit the one-to-one correspondence between linear and angular velocity, coupling linear filter gain $F(pi) - p$ and angular filter gain $H(q_i)/q_i$ using the exponential map. In this way the non-commutativity of the orientations is taken into account.

Chapter 4

Design of the orientation filter

4.1 Signal properties

Without even looking at actual data from the motion tracker one can already define some points of focus regarding motion signals when designing a comfortable filter for specific virtual reality applications. For instance the visual examination of molecule models (a typical task for the PSS) requires the motion tracker to deliver a stable signal which preferably does not oscillate when the tracked input device is held still. However, signal disturbances easily noticed by the user during "calm" motions may not be noticeable at all when quickly rotating the object to an other viewing pose: the accuracy of a filter is more important during the "slow" phases or "plateaus" of the signal, than during the phases of quick change.

Keeping the signal stable though (throwing away higher frequency components of motion) may give rise to uncomfortable side-effects in the human operator, when virtual objects but slowly follow the input motion due to the great size of the filter support (the signal neighbourhood the filter includes in its calculations, which could be regarded as the filter's receptive field). For interactive motion to be accurate and comfortable the perceived feedback-loop from the operator to the VR-apparatus and back to the operator again has to be as short as possible.

4.1.1 Properties of human motion

In [VanSomeren1996] frequency domain information is given regarding motion patterns of the wrist in a human subject executing various tasks: finger tapping, diadochokinesis (rotating the wrist about the axis along the lower arm), pin-placing (a targeting task), tooth brushing, writing and walking. Based on acceleration data the authors associate frequencies in the range of 0.5 to 11 Hz with motion actually resulting from muscle force. Furthermore Van Someren et al. identify a low frequency signal component in the neighbourhood of 0.25 Hz as gravitational artefact distorting actigraph[1] data for use in human research (e.g. research on sleep, circadian rhythms and aging). For the PSS the actual orientation of input objects (often physically connected to a human wrist) is to be measured instead of just the result of muscle activity, whether originating from translational or rotational forces. Furthermore the PSS has to accurately

[1] Actigraphy: the long-term assessment of wrist movements by means of a small solid-state recorder and accelerometers.

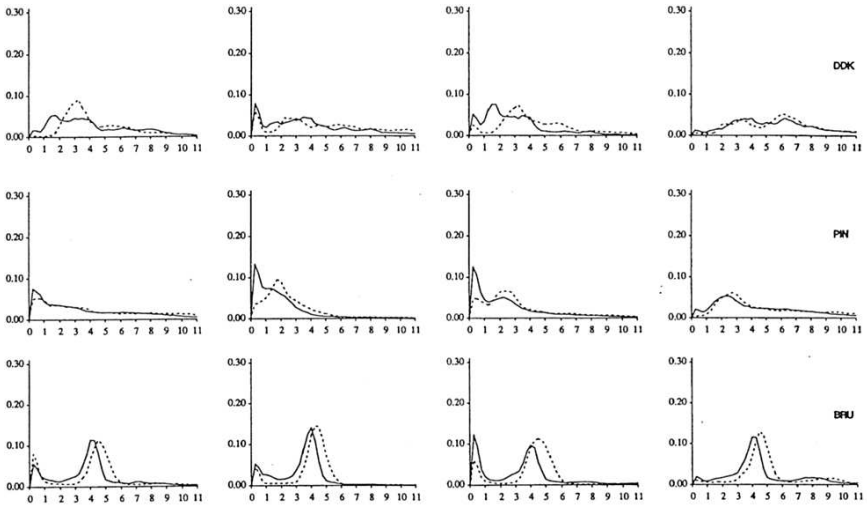


Figure 4.1: *Frequency spectra of actigraph data (adapted from [VanSomeren1996]) for three different tasks: diadochokinesis (DDK), pin-placing (PIN) and tooth brushing (BRU). The first three columns represent acceleration data from three orthogonal axes transverse on the wrist (x), from hand to arm (y) and perpendicular to the wrist (z) respectively. The last column contains the spectra of the* instantaneous acceleration vectors *($A(t) = \sqrt{x(t)^2 + y(t)^2 + z(t)^2}$).*

reproduce intended motion and not necessarily contributions such as from tremors in the motor system of the user, which show up in the higher frequency part of the signal. In [VanSomeren1996] it is noted that in earlier (human) research normal motion of the wrist is found to predominantly consist of the very low frequencies about 0.25Hz associated with gravitational artefact. This supports a focus on the lower frequencies though, if looking for intended motion resulting from both muscular and gravitational forces.

Whereas the acceleration of motion was measured and not velocity, or pose (as the Personal Space Station does), the frequency constraints are likely to be applicable to these other modalities. In the ideal case of a sinusoidal acceleration, its corresponding velocity signal (obtained by integration) has the same frequency distribution[2] which in turn holds for the double integration yielding the pose signal (seen along its curved geometry) used in the PSS. The sinusoidicity assumption is plausible for human motion, because the human body can be mechanically described by a system of weights and springs.

Concluding, an orientation filter for the PSS should let through as much of frequencies lower than 11Hz as possible. There is a trade-off between also letting through errors in the signal that contribute to frequencies within this interval and covering less of the higher frequencies involved with intended motion. In figure 4.1 can be seen that the main components of characteristic motion rarely have frequencies higher than 8Hz.

[2] Up to a gradual amplitude attenuation toward the higher frequencies: The Fourier transform of a derivative: $\frac{d^n f(t)}{dt^n} \leftrightarrow (i\Omega)^n F(\Omega)$

4.1.2 Properties of the motion tracker

Captured motion data has passed through several submodules in the PSS before it is actually used to generate the 3D virtual environment shown to the user. The total delay between the physical motion and the corresponding response of that particular motion in the virtual scene is therefore composed of a number of more or less independent delays propagated from the individual modules (see figure 4.2). For the evaluation of orientation filtering schemes we group the contributions to the total experienced delay into four parts:

- A: The time needed to derive orientation data for a specific input device from camera images;
- B: The calculation time needed to apply the filter on the presented orientation data;
- C: The signal delay introduced by the filtering method (apart from calculation time);
- D: The time needed afterwards to render and display the virtual scene.

In its current configuration the PSS is housed in two consumer PC's. One PC contains the motion tracker, which is network connected to a second PC taking care of rendering and displaying the 3D scene. Because the tracker and filtering component are executed synchronously and serially, the resulting orientation sampling rate depends on the joint delays in parts A and B. Delays in part D are marginal, because for the sampling rate only the time needed to queue motion data for transmission to the renderer PC counts. Delay variation in part A causes the sampling rate to range from approximately 25 to 40 Hz. This is because the time it takes to detect the reflective markers and to match marker patterns with reference input device models is not constant between models and over configuration space.

Our orientation filter contributes to parts B and C of this scheme: For the calculation of the filter output the algorithm uses per time step one dot product, one quaternion inverse, two quaternion products, one quaternion exponent and one quaternion logarithm. The execution time of the algorithm is directly proportional with the width of the filter kernel (see table 4.1). Memory requirements are also in the order of filter width and remain constant throughout operation. From this can be concluded that with current consumer grade computation power and capture rates the delay introduced in part B, computation of the filter output, is negligible.

Part C on the other hand, generates far more significant time lag. For a causal (physically feasible) filter, the output signal will always be delayed with respect to the input signal: the filter has to "collect" data samples from both history and future of a signal value to perform it's task correctly. As it cannot look into the future it has to delay and hold samples, thus shifting the present toward history. For instance for a five point (fourth order) filter this delay is two samples. Then the actual delay time or time lag is two times the sampling interval.

4.1.3 Properties of human perception

During on-line tests with orientation filtering it turned out that for the PSS a delay of as few as two sample points is already noticeable; it lessens the feeling of directness of the connection between input devices and virtual objects they control. As the tracker was

	dot	qinv	qprod	qexp	qlog	qfilter: (dot+qinv+2qprod+qexp+qlog)
sign	0	3	0	0	0	3
add	*w*	4	6	3	0	*w*+19
sub	0	0	6	0	0	12
mul	*w*	4	16	4	3	*w*+43
div	0	1	0	1	1	3
mod	0	0	0	0	1	1
trig	0	0	0	1	2	3
bra	0	0	0	1	1	2

Table 4.1: *Operations needed for the execution of the FIR quaternion filter algorithm ("qfilter" in this table) per time step (sample), where "dot" stands for the dot-product needed to apply the filter kernel to angular displacements, "qprod" is the quaternion product. Variable w is equal to the number of offset filter kernel coefficients. Sin(n) and cos(n) (row "trig") are assumed to be calculated in a single operation and the quaternion inverse (qinv) does not check whether the operand's length is greater than zero and therefore doesn't use branching instructions (bra) for this.*

capturing motion data with a frequency of approximately 27Hz at that time this means that the sum of the delays introduced by parts A, B, C and D should not be greater than 75 milliseconds. This observation corresponds with the guideline presented in [Azuma2001] and other work, in which delays in the order of 100 ms are regarded highly undesirable. Even delays as small as 10 ms have statistically significant negative effects on the performance of guiding a ring over a bent wire ([Azuma2001]).

Because several components of the PSS are likely to change in response to the availability of better hardware and/or tracking techniques, the focus of this report is not on the actual details of this virtual reality console, but on minimizing the delay due to filtering, guided by empirical data.

4.1.4 Properties of captured data sets

Real-life experimental data were collected in three sessions with the Personal Space Station at CWI in Amsterdam. Principal investigator of the CWI Visualization and Virtual Reality group Robert van Liere operated the apparatus (see figure 4.3) wearing 3D glasses by manipulating one or two optically marked input devices (*input devices A* and *B*). Motion data was captured in the form of space separated text files formatted

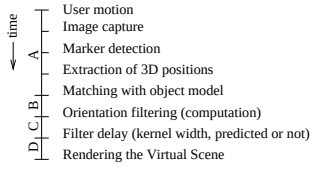


Figure 4.2: *Time-line diagram illustrating the propagation and sources of signal delay between the physical user action and the time at which the user sees the reaction to it in the virtual scene.*

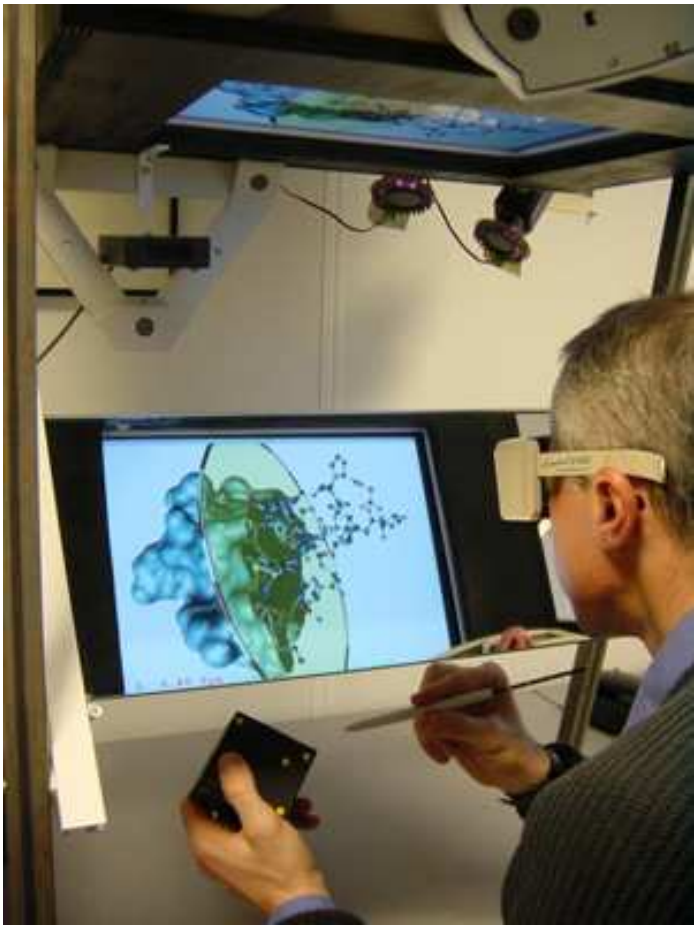


Figure 4.3: *The Personal Space Station presents the user a 3D molecule model, being examined and cut-through (like in session 2, see table* 4.3*) using two* input devices *simultaneously: an input cube and wand, labeled* A *and* B *respectively for this session. The data sets presented in this report are captured motion signals from up to two input devices. The wand on the photograph was not used during captures; another cube was used instead.*

	input device A		input device B	
time T *seconds*	translation *meters*	orientation Q *quaternions*	translation *meters*	orientation Q *quaternions*
`t :`	`x y z`	`([w] x y z)`	`x y z`	`([w] x y z)`
21.0001 :	-0.193 -0.185 0.044	([0.085335] -0.699060 -0.705084 -0.083000)	0.054 -0.187 0.051	([0.374793] 0.002215 -0.927106 -0.000763)
21.0640 :	-0.193 -0.185 0.044	([0.084453] -0.698900 -0.705541 -0.081353)	0.054 -0.187 0.051	([0.375382] 0.003061 -0.926860 -0.002957)

Table 4.2: *File format (space separated text) of the motion data recordings. The first row of the table indicates which input devices the columns represent data for. The second row lists the measured entities and their units. The next row shows the file format with the ordering of parameters and the last row contains example data as actually recorded (the first two entries of session 2).*

as shown in table 4.2. The sessions were recorded during different tasks and labeled as in table 4.3.

In figures 4.5 and 4.6 the rotational part of the captured motion is shown for the three sessions. The experiments with the filter were conducted on these data sets, each consisting of a time and quaternion channel (T and Q). Table 4.4 summarizes overall features of the sets.

Because of a side-effect of the anti-podal equivalence of quaternions (mentioned in section 2.4), the raw data from the capture files had to be corrected for "sign-flips" (see figure 4.4) which would otherwise destroy the direct correspondence between the distance measure of $SO(3)$ and the measure along the quaternion sphere.

The design of a FIR filter is often based on frequency domain information from empirical data. Multidimensional data, if linear and separable, can be analysed in the frequency domain using channel-by-channel power spectrum estimation ([Azuma1995]). As linearity and separability are also the constraints for proper operation of the FIR filtering introduced in chapter **??**, for frequency analysis we will therefore linearize the orientation data in the same way, using the angular displacement values between successive orientations (normalized to sample interval: angular velocities V).

Estimating the frequency power spectrum using the fast Fourier transform assumes a constant sample interval. Oscillations in sampling intervals can cause artifacts in the power spectrum (see figure 4.7) of the captured signal. Spectra of captured white noise will not show the artifacts, because the sequence of sampled values of pure white noise is completely unpredictable and therefore does not contain any periodicity. The artifacts in spectra of captured motion are likely to average out if spectra of data sets captured under differing temporal circumstances are combined. Besides oscillatory temporal effects PSS data sets show differing mean sample intervals. This is easily compensated for in the power spectrum estimates by scaling the frequency axis in proportion.

Figures 4.8 and 4.9 show frequency power spectrum estimates of sample interval variations and angular velocities. Note the correspondence between spectrum peaks in figure 4.1 and the plots for $V_{2,1,2}$ (manipulating a cutting plane) and $V_{3,1,2}$ (rotating a model): Peaks around 0.45π and 0.95π rad/sample for $V_{2,1,2}$ correspond with frequencies of approximately 4 and 8,5 Hz. The peaks 0.35π and 0.55π for $V_{3,1,2}$ correspond with 4,5 and 7,5 respectively (for the sample frequencies, see table above). This seems to link directly with the actigraph results from [VanSomeren1996]. That this link must not be taken for granted all too quickly is seen, however, in the obser-

Data: T (time) Q (orientation)	Gesture type	Input device
Session 1 - Examination		
$T_{1,1}, Q_{1,1,1}$	Selecting, dragging and rotating a complex molecule model.	A
Session 2 - Interactive cut-through		
$T_{2,1}, Q_{2,1,1}$	No motion.	A
$T_{2,1}, Q_{2,1,2}$	Manipulating a cutting plane to "cut" through the outer hull of a multi-layered 3D model.	B
$T_{2,2}, Q_{2,2,1}$	No motion.	A
$T_{2,2}, Q_{2,2,2}$	Selecting points (vertices) on the 3D model.	B
Session 3 - Pulling some strings		
$T_{3,1}, Q_{3,1,1}$	No Motion.	A
$T_{3,1}, Q_{3,1,2}$	Rotating a model.	B
$T_{3,2}, Q_{3,2,1}$	Selecting and dragging points on the model interconnected with elastic edges. Releasing a dragged point relaxes the model.	A
$T_{3,2}, Q_{3,2,2}$	Rotating a model.	B

Table 4.3: *Three capturing sessions were divided into 9 sets of motion data and labeled according to session, type of gestures and input devices.*

data set	**1,1,1**	**2,1,1**	**2,1,2**	**2,2,1**	**2,2,2**	**3,1,1**	**3,1,2**	**3,2,1**	**3,2,2**
number of samples	1551	1001		1587		511		726	
mean sample frequency (Hz)	27.5	18.1		29.8		27.1		36.5	
mean sample interval (sec)	0.0364	0.0552		0.0336		0.0369		0.0274	
stdev sample interval	0.0056	0.0490		0.0053		0.0072		0.0264	
mean angular rotation speed (rad/sec)	0.354	0.144	0.542	0.192	0.372	0.067	0.238	0.284	0.014
stdev angular rotation speed	0.267	0.116	0.661	0.138	0.396	0.105	0.289	0.551	0.063

Table 4.4: *Overall properties of the data sets. Labels correspond with the first column of table 4.3. Angular speed is twice as fast as measured along the unit-quaternion hypersphere.*

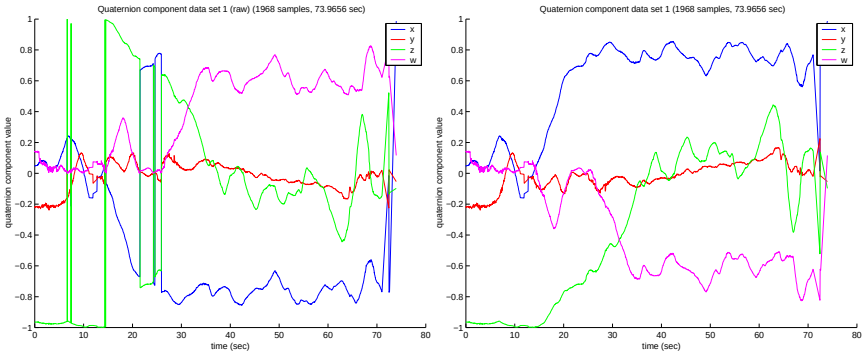


Figure 4.4: *Data set 1 - Graphs of the quaternion components x, y, z and w indicating the orientation of a tracked device, captured while examining a molecule model. The left graph shows that due to equipodal equivalence the raw signal extracted from rotation matrices sometimes "flips" its sign. At these instants the signal seems to "travel" with great velocity over the unit-quaternion sphere, which has an undesirable effect on filter output. In the right picture the signal is checked and corrected for such flips by keeping the distance between successive quaternions smaller than* $\frac{1}{2}\pi$ *along the sphere, which corresponds with an Euclidean distance smaller than* $\sqrt{2}$*. If the distance is equal or smaller the complex sign of the quaternion is flipped: the individual quaternion components are negated. The first quaternion is made to have the least negative signs in the components by negating when required.*

vation that the peaks are most likely due to sample interval variations (as shown in the $\Delta\Delta T_{2,1}$ spectrum) - at least for the manipulation of the cutting plane. In defense: The (large) variations are possibly caused by moving an input device into configurations harder to track. This means that although interval variations are causing the local maxima, these may still originate from actual motion. And ascribing all of the peaks to temporal inaccuracies does not fit in with the plots for set 3,1 (lower left graphic in figure 4.9), where time is fairly uniform whereas the captured motion does still show a characteristic peak.

The peaks are not present in no-motion data (noise) like $V_{2,1,1}$ captured concurrently for the other input device, because the whiteness of noise is likely to conceal temporal periodicity. In the $V_{2,2,2}$ plot the effects from sample interval variation are not distinctively present, because they are much weaker (note the logarithmic dB-scale) and the dataset has fewer dominant frequencies (compare quaternion plots in figure 4.6). We will not concentrate on these higher frequency details because of the decreasing signal/noise ratio toward higher frequencies and because the mentioned peaks do occur in but a few of the captures.

4.2 Filter properties

In chapter 3 it was shown how a linear filter can be applied to rotational data using a discrete convolution kernel (*filter mask*). To use the filter in an actual application, suitable filter parameters have to be determined.

A FIR-filter can be implemented by convolving the input signal with the filter's

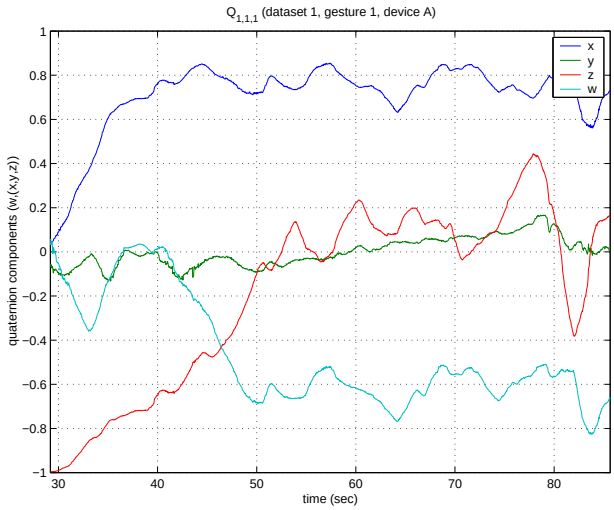


Figure 4.5: *Dataset 1 - quaternion components*

impulse response. The impulse response of the filter defines its convolution kernel coefficients and vice versa. A change in the kernel coefficients is reflected in the impulse response, influencing the *frequency* and *phase response* in the frequency domain. A priori frequency-domain information about the signal can suggest a desirable response of the filter. This can be translated back into kernel coefficients used in the filtering algorithm.

For the kernel coefficients of an FIR-filter several distributions can be chosen, which have their characteristic effects on the signal. The data in [VanSomeren1996] and the observation that the virtual objects are "trembling" too much and that, compared to the erroneous tremor, the intended motions are slow, suggests the application of a low-pass *smoothing* filter to the PSS rotational motion signal. The filter has to filter out the higher frequencies and let the lower through. Frequency domain filter design techniques are used to validate and refine this intuition.

4.2.1 Using a signal/noise model

In [Press2002] the removal of noise from a signal is demonstrated using *Optimal Wiener Filtering.* From a frequency power spectrum of a measured signal the frequency profiles of the uncorrupted signal and noise are estimated. The spectrum should show signal peaks with a "noise tail" which can be extrapolated to generate a *noise model*. Subtraction of the noise model from the measured spectrum yields the signal model: from this frequency domain model the ideal filter $\Phi(\omega)$ can be found by determining the ratio between signal and signal-plus-noise power:

$$\Phi(\omega) = \frac{|S(\omega)|^2}{|S(\omega)|^2 + |N(\omega)|^2} \tag{4.1}$$

This method assumes that measurements containing both noise and signal are used and that the noise model is extrapolated by intuition from the frequency spectra of the

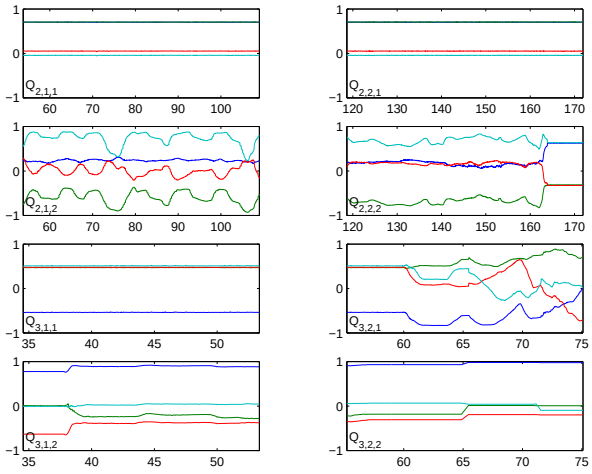


Figure 4.6: *Datasets 2 and 3 - quaternion components*

measurements (C in figure 4.10). Empirical information on noise is available for the PSS though, because in the experiments we captured several episodes of "silence", without intended motion. If the noise (frequency power) profile $|N_2(\omega)|^2$ of errors in a measurement containing intended motion is comparable with (preferably equal to) the profile $|N_1(\omega)|^2$ of the silent episodes, then the optimal filtering relation can be rewritten to yield an estimate of the target filter:

$$\widehat{\Phi}(\omega) = 1 - \frac{|N_1(\omega)|^2}{|S(\omega)|^2 + |N_2(\omega)|^2}, \; N_1(\omega) \approx N_2(\omega) \tag{4.2}$$

It determines the contribution of noise to the total signal, the frequency gain on the signal leaving the noise. The opposite of this, the total signal (i.e. identity) profile minus noise contribution is the desired filter gain.

4.2.2 Determining filter coefficients

Filter coefficients can automatically be found from signal-and-noise data c, noise-only data n and preferred filter order o as follows:

1. $C(\omega)$ = frequency power spectrum estimate of c.
2. $N(\omega)$ = frequency power spectrum estimate of n.
3. Desired filter gain $G(\omega) = 1 - \frac{N(\omega)}{C(\omega)}$ (the signal model $S(\omega)$ is $C(\omega)$ multiplied by $G(\omega)$).
4. Scale G in a way that the gain G ranges from full to none at all: $G(\omega) = G(\omega) - \min(G(\omega))$ and then $G(\omega) = G(\omega) / \max(G(\omega))$.
5. Do this for different combinations of signal-and-noise and noise data sets and average the resulting filter gains.

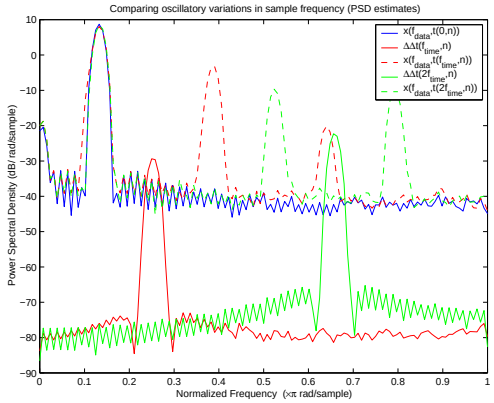


Figure 4.7: *If a sinusoidal signal (of frequency f_{data} plus some white noise) is sampled evenly (at time instants $t(0,n)$, samples are taken with a constant interval) then the frequency power spectrum estimate will show a peak at the frequency of the oscillation in a fairly flat spectrum. If oscillations in sample intervals are introduced (say, of frequency f_{time}), the spectra show artifacts that are characteristic for the temporal oscillations. The signal peak however stays in the same location.*

6. Find filter coefficients matching a desired filter order o (width or support) using an out-of-the-box filter approximation method like Parks-McClellan's using the Remez Exchange Algorithm to find an optimal (equiripple) fit to the target filter gain (see [Press2002] and the MatLab `remez` function).

The resulting gain of this filter design is shown in figure 4.12 (marked with squares).

Refining the filter

While the procedure above does produce a low-pass-like filter, it is not really the most desirable one. We can refine the target filter gain $G(\omega)$ (the averaged gain computed from measured data, see figures 4.12 and **??**) by enforcing the a-priori knowledge that we are searching for a low-pass filter:

1. Determine the point $(\omega_{cut}; G(\omega_{cut}))$ where gain $G(\omega)$ drops below 0.5 (-3 dB) the first time (from low to high frequencies).
2. Fit a straight line to the gain values from 0 Hz up to the -3 dB point $(\omega_{cut}; G(\omega_{cut}))$.
3. Determine ω_{slope}: the lowest frequency where the fitted line drops below 1.
4. Determine ω_{zero}: the lowest frequency where the fitted line is zero or below.
5. Gain constraints $G_{constr}(c) = \{\{\omega_{c,min}, g_{c,min}\}, \{\omega_{c,max}, g_{c,max}\}\}$ where $c = \{1,2\}$ are used as piece-wise linear gain specifications to which filter designs must adhere to fit the derived data model as described by $G(\omega)$ (see figure 4.12). They are expressed in frequency-gain pairs as follows (frequencies are normalized):

$$\begin{array}{cc} & \begin{array}{cc} c=1 & \quad\quad c=2 \end{array} \\ \begin{array}{c} frequency \\ gain \end{array} & \begin{pmatrix} 0 & \omega_{slope} \\ 1 & 1 \end{pmatrix} \quad \begin{pmatrix} \omega_{zero} & 1 \\ 0 & 0 \end{pmatrix} \end{array}$$

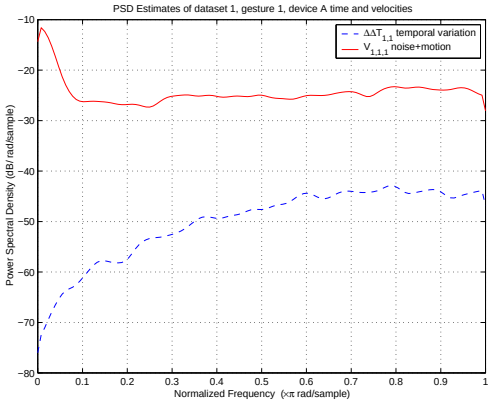


Figure 4.8: *Frequency power spectrum plots of dataset 1 sample interval deviations (the change in $\Delta T_{1,1}$) and angular velocities ($V_{1,1,1}$).*

6. Feed $G_{constr}(c)$ into a filter design method (see above) to get the filter coefficients.

The (filter order 4) result of this is marked with circles in figure 4.12 and has the following coefficients:

$$\frac{\begin{bmatrix} 9 & 12 & 14 & 12 & 9 \end{bmatrix}}{56} \pm 0.5\%$$

The binomial kernel

In [Lee2002] a binomial kernel is suggested for smoothing of orientation data. This distribution of kernel coefficients is interesting because of its smooth and expressive low pass behaviour in the frequency domain. A binomial filter kernel of order four for instance contains five coefficients (see figure 4.13) corresponding with the first values of Pascal's triangle up to a scale factor:

$$binomial(4) = [1,4,6,4,1]/16$$

As can be seen in figure 4.12, the binomial kernel yields a frequency gain curve, which is smooth and monotonously decreasing, but not as steep as the gains of the fitted filter kernel of the same order.

4.2.3 Kernel width

In filter design much revolves about accurately describing the signal (thus separating the signal from noise) while keeping complexity low. This complexity has algorithmic, spatial (memory related) and temporal faces. The trade-off is amongst others prominent in choosing the appropriate width of a filter. The described automatic filter design method and the binomial distribution have one free parameter, the order of the filter, directly related to the width of a filter constructed along this line: a higher order will broaden the filter and increase suppression potential of higher frequency components

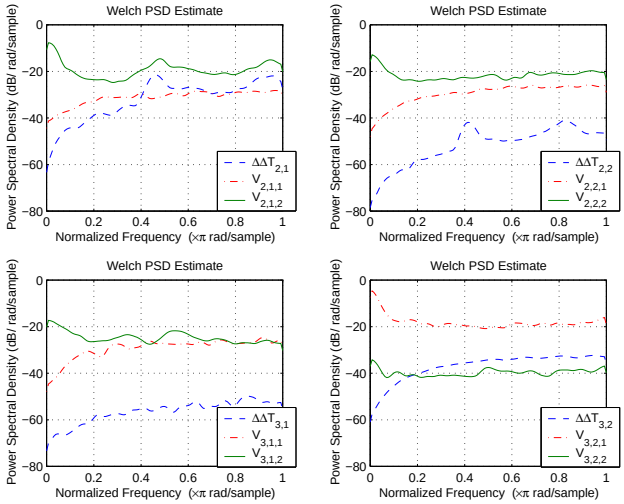


Figure 4.9: *Frequency power spectrum plots of dataset 2 and 3 sample interval deviations (the change in $\Delta T_{dataset,gesture}$) and angular velocities ($V_{dataset,gesture,device}$).*

in a signal, whereas lower orders allow for more of the measured signal to pass (see figure 4.14).

With this in mind one would choose a filter width that tightly fits the properties of the signal and excludes noise to the greatest possible extent, if complexity was not an issue. In the context of FIR filters: For many applications (including the one at hand) the ideal signal is built up from fairly low frequency components and the real signal is accompanied by higher-frequency noise. Therefore the most accurate filter kernel would have the maximum amount of coefficients for which the signal is boosted and not suppressed, because then even noise that is hardly different from the signal[3] will still be attenuated. In many applications this is a possibility indeed, but in the real-time constrained world of virtual and augmented reality a tight fit is often not desirable. Not (only) because of variation in signal characteristics throughout the use of the application, but to a more restraining extent because of *temporal complexity:*

Data sampled in a real-time situation has the (in some cases annoying) property of being late. FIR filters acting on data that is behind in time usually make things worse, because they use (a limited amount of) data around a signal value to calculate the filter result value. For a linear FIR filter to behave properly in the frequency domain filter masks have to be symmetric[4] which causes the filtered value to be located in the "center" of the filter, a value valid for a time instant that occurred several sample

[3] At the right (high-frequency) side of the spectrum in this example, but it extends to band-pass filtering.

[4] Applying a filter kernel $[k_{-n}...,k_{-1},k_0]$ on the data $[d_{t-n}...,d_{t-1},d_t]$ within the support of the filter has the same effect as mirroring kernel and supported data at their endpoints (up to a scale factor equal to the number of "mirrors" plus one). Due to (higher order) discontinuities in the mirrored signal high-frequency artifacts are introduced in the filtered signal. These effects are of greater magnitude for asymmetric kernels.

caption for figure: Applying half of a symmetric filter kernel (to samples $d_{t-n}..d_t$: e.g. $\frac{[1,4,6]}{11} \cdot [d_{t-2};d_{t-1};d_t]$) has the effect of the full kernel (e.g. $[1,4,6,6,4,1]/22$) on mirrored data (in the last sample d_t : e.g. $\frac{[1,4,6,6,4,1]}{22} \cdot [d_{t-2};d_{t-1};d_t;d_t;d_{t+1};d_{t+2}]$), introducing high-frequency artifacts, due to higher order discontinuities in the mirrored signal.

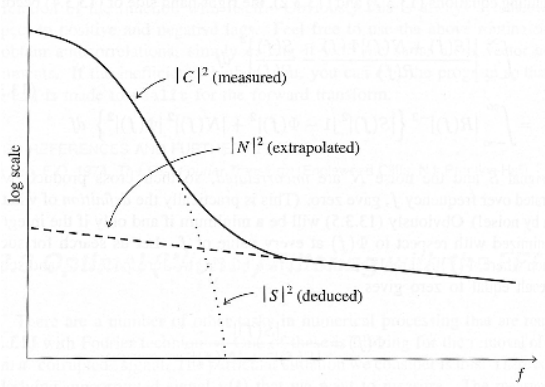


Figure 4.10: *For Optimal (Wiener) Filtering a signal model is deduced by subtracting an extrapolated spectral noise model from the frequency power spectrum of the measured signal* ([Press2002], page 554).

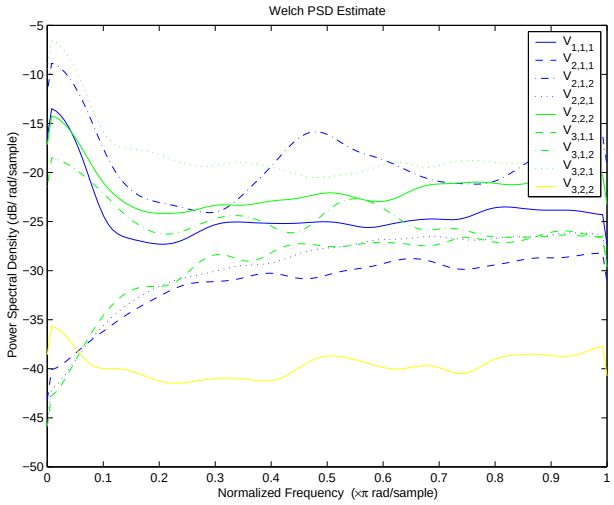


Figure 4.11: *An overlay of the velocity frequency spectrum estimates shows a sketch of an empirically supported signal/noise model. Data set* $3,2,2$ *shows an anomalous overall spectrum power (though the shape is consistent) caused by episodes where the tracker could not find the input device (see figure 4.6; indices correspond with data set labeling in table 4.3). It was included to show the robustness of the filter estimation method.*

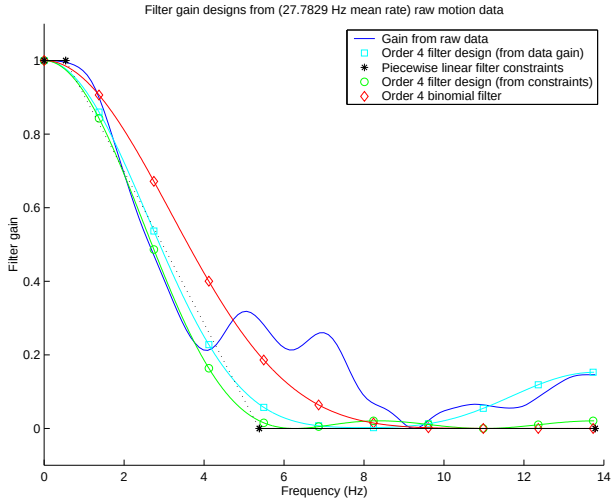


Figure 4.12: *Filter gain designs based on signal gains derived from measured data sets. The line without markers indicates the mean measured signal gain $G(\omega)$. The line marked with squares is the gain of an order-4 filter designed directly from the signal gain. Solid lines between star markers indicate gain constraints G_{constr} used to design the order-4 filter with the circle-marked gain. For comparison the gain of an order-4 binomial filter is also shown (with diamond-shaped markers).*

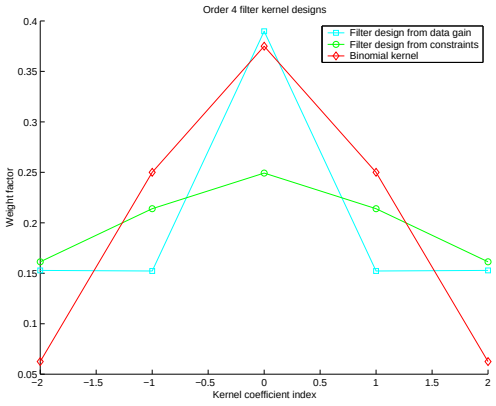


Figure 4.13: *Filter kernels for designs of order 4. Marked with squares the design derived directly from the mean measured signal gain. Marked with circles the design based on filter constraints and for comparison marked with diamond-shapes the order-4 binomial smoothing kernel.*

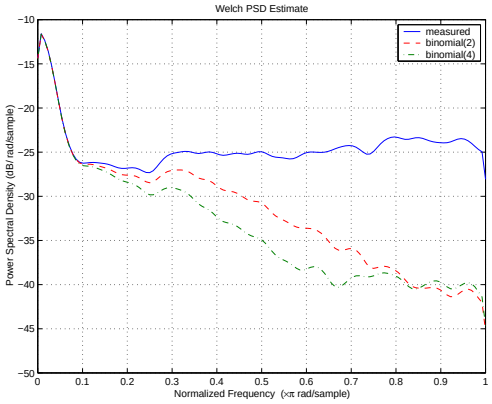


Figure 4.14: *A kernel with fewer coefficients yielding a smaller filter shortens time lag at the cost of attenuation steepness as seen in the frequency power spectra of a signal filtered at two different filter widths.*

intervals in history.

The real-time constraint dictates upper boundaries for the width of useful filters, filtered data has to be as up-to-date as possible: from a certain point the delay introduced by an increased width renders a filter useless. Regarding the PSS and other VR applications, filters must not introduce a delay that hinders the human operators in their interactions with the synthetic world. Unfortunately unwanted effects of delay are typically noticed on a much smaller time scale than interactive motion signals. Taking the delay effects into account quickly narrows down the search for a suitable filter width.

As noted in section 4.1.3, with current sampling frequencies, two samples lag is already noticeable. This means that the radius of symmetric filters like designed above should not exceed 2, corresponding with filter orders 2 and 4 (3 and 5 filter kernel coefficients respectively).

4.3 Lag reduction

4.3.1 Using asymmetric kernels

Kernels discussed above are symmetric and have a maximum in the center which means that for the bigger part the filter output contains information from the sample in the center of the filter's support. By using asymmetric kernels the emphasis can be shifted toward more recent samples. However, this comes at a cost: lag reduction is traded for weaker noise suppression, because of the frequency domain effects of asymmetry. An asymmetric kernel can be formed by truncating and scaling (preserving the sum of coefficients) a symmetric one. Using the first $k+1$ coefficients of an order $1.5k$ kernel meets lag and noise reduction mid-way (see figure 4.15).

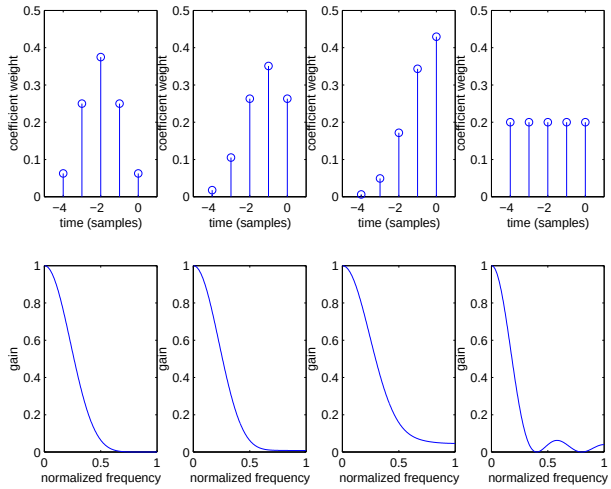


Figure 4.15: *From left to right: a symmetric kernel (upper picture) of order 4 with a frequency gain of 1 at the lowest frequency and a gain of 0 at the highest frequency. The gain decreases smoothly and monotonously. The first 5 coefficients of an order 6 filter form an asymmetric kernel which has a frequency gain that is still smooth and monotonously decreasing, although the gain does not reach zero at the highest frequencies. This effect is emphasized with the truncation of higher order kernels. For comparison the gain of an ordinary moving average kernel is shown. Note that the graphs are in linear, not (logarithmic) dB-scale.*

4.3.2 Prediction by velocity extrapolation

The observation that acceleration has an upper bound can be used to introduce an extrapolation or prediction step that helps to eliminate the time lag of a filter. For short time steps there even might be hardly any change in velocity. If the sample interval of the motion interval is short enough, a predictor could assume constant velocity for several samples.

A predictor was implemented by calculating a weighed average of past velocity values and applying this average $\bar{v}$ to the current orientation τ times.

$$\begin{aligned} \bar{v_i} &= \sum_{m=-n}^{m=-1} a_m v_{i+m} \\ \hat{\mathbf{q}}_i &= \mathbf{q}_i \exp(\tau \bar{v_i}) \end{aligned}$$

This predictor uses a FIR filter on velocity rather than measured orientation data. For the coefficients of the filter, the same distribution as used with the orientation filter can be chosen, noting that the selection of a weight distribution can be focused on minimizing lag, which might deteriorate noise suppression power, however if the prediction is performed on already filtered data it has less noise to suppress to start with. It is a good idea to predict *after* noise reduction, because of the predictor's sensibility to noise.

4.3.3 Other filtering methods

Low-pass FIR filters are traditionally designed symmetrically, with gradually decreasing coefficients toward the extremes of the filter kernel, the outer non-zero values as small as possible. This is because the shape of the filter has to be similar to the target signal. The lag of such filters is proportional to the width of the filter (even, though to a lesser extent, if asymmetric filters are employed). In search for lower lags we regard Kalman filtering and double exponential smoothing, because they both are able to focus on recent samples. The second method is included because in a recent paper ([Laviola2003]) it was compared with Kalman filtering and found to be a computationally cheap method capable of producing results similar to those of the more involved Kalman filter.

Kalman filtering

An out-of-the-box toolkit ([Murphy2003]) was used to Kalman filter the motion data. The system model A incorporates the current orientation represented in quaternion components and their first and second derivative, which for small changes correspond with angular velocity and acceleration:

$$\mathbf{x}_t = \begin{pmatrix} \mathbf{q}_t \\ \mathbf{v}_t \\ \mathbf{a}_t \end{pmatrix} = (q_{t,w}, q_{t,x}, q_{t,y}, q_{t,z}, v_{t,w}, v_{t,x}, v_{t,y}, v_{t,z}, a_{t,w}, a_{t,x}, a_{t,y}, a_{t,z})^T \qquad (4.3)$$

$$A = \begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & & & & 1 & & & & & & 0 \\
0 & & 1 & & \ddots & & 1 & & \ddots & & & 0 \\
0 & & & 1 & & & & 1 & & \ddots & & 0 \\
0 & & & & 1 & & \ddots & & 1 & & & 0 \\
0 & & \ddots & & & 1 & & & & 1 & & 0 \\
0 & & & & & & 1 & & \ddots & & 1 & 0 \\
0 & & & & \ddots & & & 1 & & & & 1 \\
0 & & \ddots & & & & & & 1 & & & 0 \\
0 & & & \ddots & & & \ddots & & & 1 & & 0 \\
0 & & & & & & & & & & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \tag{4.4}$$

Incorporating acceleration could have a positive effect on state predictions calculated by the Kalman filter using the system model (cutting down lag and overshoot), keeping in mind though that higher derivatives are more sensitive to noise. As greater uncertainties are assigned to measurements and the filter relies more on the system model, the data gets smoother, but overshoot increases. This could be overcome to a certain degree by including acceleration in the system model. A damping factor on acceleration (natural motion cannot accelerate freely indeed) might counteract this effect to a greater extent.

Only instantaneous orientation is measured by the PSS so measurement model H and state y include just the current orientation value:

$$\mathbf{y}_t = \mathbf{q}_t = \begin{pmatrix} q_{t,w} \\ q_{t,x} \\ q_{t,y} \\ q_{t,z} \end{pmatrix} \tag{4.5}$$

$$H = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & & \ddots & & \ddots & & \ddots & & \ddots & 0 \\
0 & 0 & 1 & 0 & & \ddots & & \ddots & & \ddots & & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \tag{4.6}$$

In the literature (for instance in [Azuma1995]) it is concluded that the use of accelerometers can greatly improve measurement and prediction accuracy over measuring orientation alone. This is a plausible observation because the constraints on acceleration of the human end effector (the hand) can be more generally described (by less convoluted means, using laws of physics on moving bodies) than constraints on its actual attitude or pose (which are likely to be coordinate- and user-dependent).

Filtered values are normalized as the filter acts on quaternion components in $\mathbb{R}^4$ and the resulting quaternions could diverge from the unit-quaternion sphere. The outcome of the filter could be used as input to a predictor (e.g. the method described in section 4.3.2).

Double Exponential Smoothing

A much simpler (computationally cheaper) filtering method is called *Double Exponential Smoothing* (DES), as found in [Laviola2003]. It mixes the current sample with a smoothed value S_t. This is the first exponential effect. The second is caused by mixing S_t in its turn with the previous double-smoothed value S'_{t-1}:

$$\begin{array}{rcc} S_t & = & \alpha q_t + (1-\alpha)\, S_{t-1} \\ S'_t & = & \alpha S_t + (1-\alpha)\, S'_{t-1} \\ \hat{q}_{lo,t} & = & \left(\left(2+\left(\lfloor\tau\rfloor\,\frac{\alpha}{1-\alpha}\right)\right) S_t - \left(1+\left(\lfloor\tau\rfloor\,\frac{\alpha}{1-\alpha}\right)\right) S'_t\right) \\ \hat{q}_{hi,t} & = & \left(\left(2+\left(\lceil\tau\rceil\,\frac{\alpha}{1-\alpha}\right)\right) S_t - \left(1+\left(\lceil\tau\rceil\,\frac{\alpha}{1-\alpha}\right)\right) S'_t\right) \\ q_{lo,t} & = & \frac{\hat{q}_{lo,t}}{\left|\hat{q}_{lo,t}\right|} \\ q_{hi,t} & = & \frac{\hat{q}_{hi,t}}{\left|\hat{q}_{hi,t}\right|} \\ \rho & = & \tau - \lfloor\tau\rfloor \\ q_{des,t} & = & q_{lo,t}\exp^{\rho\log\left(q_{lo,t}^{-1} q_{hi,t}\right)} \end{array} \tag{4.7}$$

$$\begin{array}{c} 0 \leq \alpha \leq 1 \\ \tau \leq 0 \end{array}$$

The two parameters, α and τ, control the amount of smoothing and prediction respectively. Lower values of α cause the filter to rely more on the smoothed values and the value of τ determines the number of samples ahead the filter will predict. Parameter α could be seen as the balancing factor between a system model and measurements, comparable to the magnitudes of error covariances in a Kalman filter. The double exponential smoothing algorithm also assumes linearity in quaternion components like the Kalman filter above, although for the interpolation between predictions of integer intervals (of τ) spherical linear interpolation (SLERP) is used.

Chapter 5

Results

5.1 Evaluation method

The results of application of filters to captured data are evaluated in the following contexts:

- How well does the filter suppress noise frequencies?
- How much time lag does the filter introduce?
- How much overshoot is introduced by the filter?

In the following sections quantitative criteria are formulated for these contexts, noise reduction, filter lag and overshoot respectively.

5.1.1 Noise reduction

The filter gain model derived automatically in section 4.2 can be used to evaluate the filter's capabilities to suppress noise and let the signal through. For this purpose we first determine the gain of the filter from filtered and unfiltered versions of the same signal and then compute the error with the gain constraints from the filter design:

1. $G_{constr}(c)$ = piecewise linear derived filter gain constraints
2. $P_m(\omega)$ = frequency power spectrum density estimate for unfiltered data
3. $P_f(\omega)$ = power spectrum estimate for filtered data
4. $G_f(\omega) = \frac{P_f(\omega)}{P_m(\omega)}$ the observed gain of the filter
5. $G_{fsupp}(\omega) = \begin{cases} 1, & G_f > 1 \\ G_f, & otherwise \end{cases}$ gain of the filter without overshoot-effects (measuring frequency suppression only, overshoot is dealt with separately)
6. $d_{f,c}([\omega_{c,min};\omega_{c,max}])$ = the squared differences between $G_{fsupp}([\omega_{c,min};\omega_{c,max}])$ and the linearly interpolated gain constraint $G_{contr}(c)$ (from $\omega_{c,min}$ to $\omega_{c,max}$)

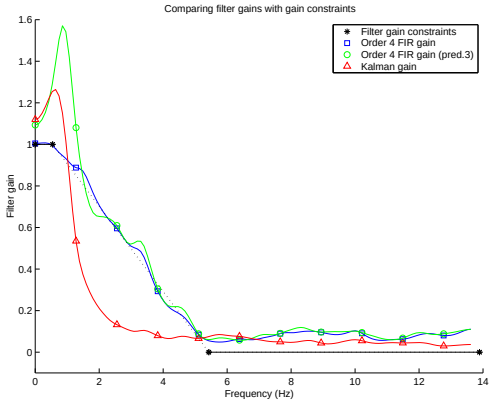


Figure 5.1: *The filter gain performance measure penalizes gain values differing from the values at constrained frequencies (solid lines marked with stars). At frequencies where no constraints are explicitly defined (dotted lines marked with stars) only the exceeding of interpolated values between filter gain constraints is considered a performance cost. In this way the Kalman gain in the picture (line with triangle markers) gets a higher performance value than the gain of the predicted FIR filter (marked with circles), as it should.*

7. $d_{f,c}((\omega_{c-1,max};\omega_{c,min}))$ = the squared positive differences between $G_{fsupp}((\omega_{c-1,max};\omega_{c,min}))$ and the line between the gain constraint values at $\omega_{c-1,max}$ and $\omega_{c,min}$ (the extremities of the space between two constraints): by only counting positive differences, where the calculated gain exceeds the constraints, filters are allowed to be more suppressive in the inter-constraint regions without penalty, while failure to adhere to the upper bound interpolated between constraint extremities is discouraged

8. $d_f(\omega)$ combines $d_{f,c}$ for all c in $G_{constr}(c)$ so that all frequencies in G_f are covered

9. $\varepsilon_f = \sqrt{\overline{d_f}}$ the RMS error against the constraints

10. $p_f = 100 \cdot \left(1 - \frac{\varepsilon_f}{\varepsilon_{id}}\right)$ the noise reduction performance measure in percents

Gain error measure ε_f indicates how well the constraints are met. To get a filter performance measure p_f the ratio between ε_f end error ε_{id} of the identity filter (unit gain for all frequencies) is used in a way that attributes a value of zero percent noise suppression to cases where the filter gain is identical to that of the identity filter, and a value of hundred percent to filters with a gain equivalent to the target gain.

5.1.2 Lag

Because real-time constraints dictate filter lags of order of magnitude 10 or even a single sample, it is convenient to have a filter lag measure with subsample resolution: For different relative displacements of the same data, one unfiltered, the other filtered,

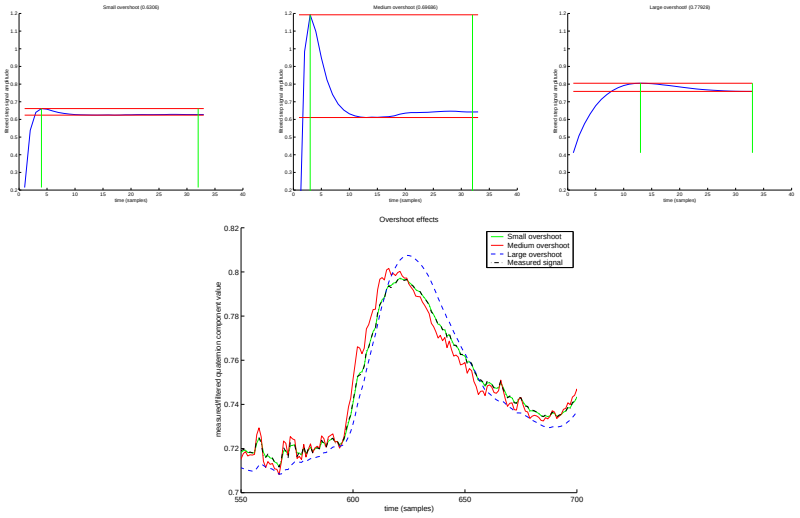


Figure 5.2: *The step response of a filter reveals overshoot behaviour (upper pictures). Note that a slow (low frequency) overshoot (the right picture) can have more impact than a sharp spike with a high amplitude (top middle picture), as can be seen in the graph of an actual signal filtered with corresponding degrees of overshoot (lower picture).*

the mean squared error values are computed, the displacement yielding the minimum value is regarded as the lag value at sample resolution. A second degree polynomial fit is then performed on this and its neighbouring error values. The location with the minimum value in this fit is considered the lag value at subsample precision.

5.1.3 Overshoot

A filter that anticipates future values of the signal is not likely to be always right in its predictions. The signal may show dynamics that are not accounted for in the prediction algorithm. Unforeseen changes in the signal can cause *overshoot*, in the same way the estimates of filters that incorporate one or more trends of some order, like double exponential smoothers, could diverge from reasonable values. From a frequency domain perspective, overshoot amplifies frequencies in the signal, in particular lower frequencies if the overshoot is due to a smoothing (or trend-based) filtering mechanism.

We could examine overshoot effects by looking at filter gains exceeding the value of one, indicating amplification instead of attenuation of frequencies. However, it is hard to define a sensible measure to quantify overshoot in this way. For instance lower frequency overshoot has higher impact than a higher frequency overshoot with the same frequency power (at lower frequencies the effect is smeared).

The filter's *step response* though contains better clues about the amount of overshoot: After *Wiener filtering* a (stair-)step function using the filter gain derived from the frequency power ratio of measured and filtered signals, a step response plot of a filter with a large overshoot will show one or more large oscillations before the filter settles to a stable output value.

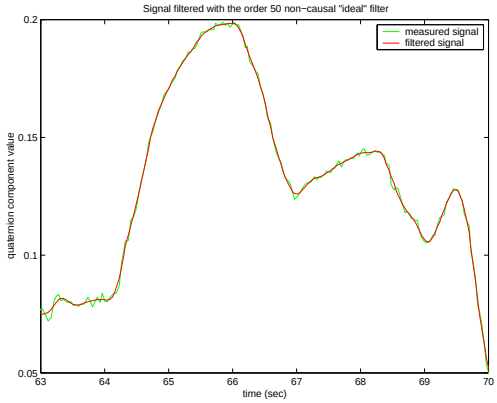


Figure 5.3: *Low- and high-frequency overshoot is measured by calculating the RMS distance of a filtered signal to an "ideal" signal: the measured signal filtered with a non-causal filter designed using the derived frequency spectrum of the motion signal (see figure 4.12). In this picture the measured and "ideal" signal are plotted.*

To rank the filters depending on the impact of the introduced overshoot the mean value of the overshoot sensitive region in the step response (the part of the response that exceeds the minimum value in the response tail) could be calculated. This value measures the signal power of the filtered step signal. The step function produces a signal containing both very high frequencies and very low frequencies, but the largest contribution to total power comes from the lower frequencies. This corresponds with the notion that low frequency overshoots have a large impact on the signal (see figure 5.2). A problem with this power measure is that it is not only influenced by overshoot, but also by *undershoot*, or the *attenuation* of lower frequencies. Between both low-pass FIR and DES filters this is not an issue, whereas Kalman filters could lock onto a band of frequencies not including the lowest, which has a negative effect on the step response power.

A more usable measure seems the RMS angular distance of the filtered signal to the signal filtered off-line using an "ideal" filter (see figure 5.3): a non-causal filter of high order, designed from the derived frequency spectrum of the motion signal. Defined this way the measure has a clear unit (radians or degrees) and yields comparable values for low- and high-frequency overshoot corresponding with what one would consider equal amounts of overshoot just by looking at the data bare-eyed. The measure is applicable to band-pass as to low-pass filters equally well.

5.2 Experiments

To evaluate filters one dataset, from session 1 (see table 4.3), was filtered using different methods and parameter values. This dataset was chosen for its temporal stability and richness of rotational motion (compare figures 4.5, 4.6 and time deviation frequency power and distribution in figures 4.8 and 4.9). The performance of each filter was measured by examining noise reduction (resemblance of the resultant filter gain with the gain design), signal lag and overshoot measure as described in section 5.1. The

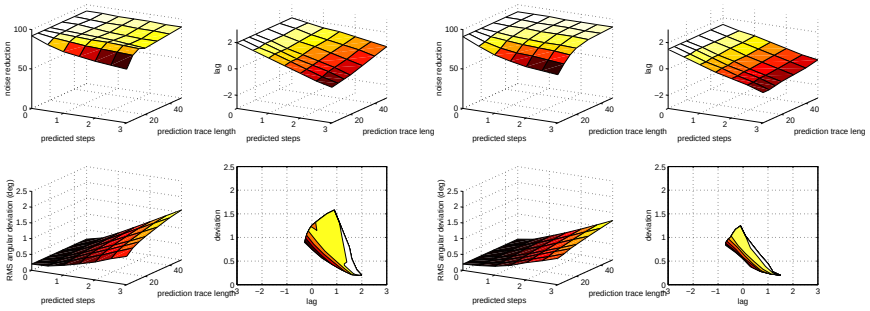


Figure 5.4: *Evaluating FIR with prediction: Lighter colors represent higher values. Colors in the lower right sub-graphs correspond with the colors in the upper left sub-graphs - indicating noise reduction performance (0-100%). The picture to the left shows the results for a symmetric versus those for an asymmetric kernel and prediction window in the picture to the right.*

overshoot measure of raw data, RMS deviation to the “ideal" signal, is 0.4 degrees.

5.2.1 Predictive FIR filtering

In these experiments a FIR filter was applied to the signal using the tangent-space (offset) filtering method described in chapter **??**. It was configured to use a filter kernel refined as described in section 4.2.2. Using various parameter values from the intervals $[0;3]$ for the number of samples to predict (*Predicted Steps*) and $[5;50]$ for the number of samples the prediction is based upon (the prediction support or *Prediction Trace Length*) the motion signal was filtered. Two sets of results are shown in figure 5.4. One set using a symmetric filter kernel and prediction trace and one set using an asymmetric kernel (of order 6, designed as described in section 4.2.2, truncated and renormalized to the first 5 coefficients) and trace (binomial distributed) as suggested in section 4.3.1. Viewing the graphs of the results for predictive FIR filtering the following can be observed:

- Noise reduction performance decreases with more prediction and/or prediction based on shorter signal traces (upper left sub-graphs);
- Signal lags are lower where more is predicted based on more recent history (actual lag reduction follows the amount of predicted steps, see upper right sub-graphs);
- Prediction causes overshoot (lower left sub-graphs);
- As more values in history are used to predict (longer prediction trace), overshoot increases (lower left sub-graphs), and prediction strength diminishes (upper right sub-graphs);
- The use of an asymmetric kernel lowers lags with half a sample interval while decreasing noise reduction performance only slightly;

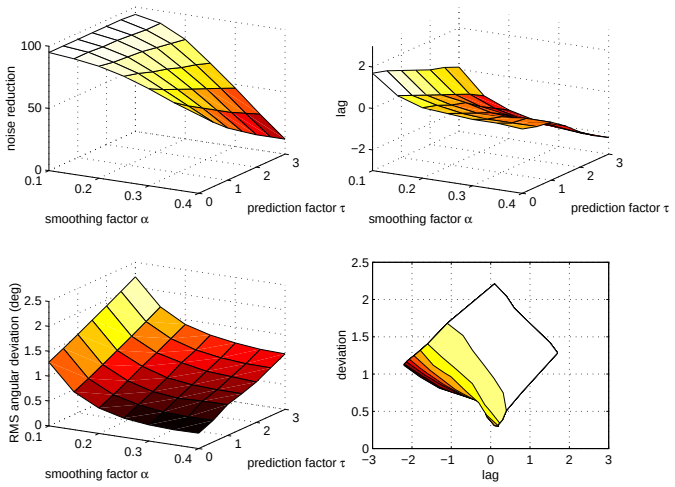


Figure 5.5: *Evaluating the Double Exponential Smoothing filter: Lighter colors represent higher values. Colors in the lower right graph correspond with the colors in the upper left graph - indicating noise reduction performance (0-100%).*

- Shorter asymmetric prediction traces have a greater negative impact on noise reduction performance, longer traces show less overshoot than their symmetric counterparts.

To examine the advantages of filtering in orientation tangent space as opposed to renormalizing the results of quaternion component filtering, motion data were also filtered with a traditional (euclidean) predictive FIR filter directly on quaternion components, using the same set of parameters as with its spherical counterpart described above. Visually the result graphs are indistinguishable (also see section 5.2.5).

5.2.2 Predictive Double Exponential Smoothing

For the predictive DES experiments α was chosen to vary from 0.1 to 0.4. Value 0 and 1 were excluded because they represent the degenerate cases where no signal gets through and where the algorithm is undefined (because of a division by zero, see section 4.3.3) respectively. Values above 0.4 were not included because their noise reduction performance is below 50% (and the graphs are more readable when they are left out). Prediction parameter τ was evenly sampled from $[0;3]$ with distances of 0.5. Viewing figure 5.5 the following can be observed:

- The lower the α factor, the more the filter smooths, suppressing noise (upper left graph);
- Both heavy smoothing and predicting increase overshoot (lower left graph);
- Signal lags are low where less smoothing is performed or more is predicted (the actual reduction in signal lag follows the specified prediction time parameter τ, see the upper right graph).

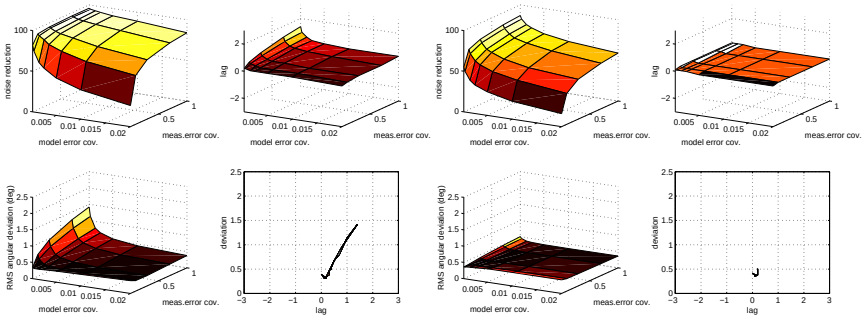


Figure 5.6: *Evaluation: Kalman filtering without (left) and with (right) acceleration in the system model. Lighter colors represent higher values. Colors in the lower right graph correspond with the colors in the upper left graph - indicating noise reduction performance (0-100%).*

In equation 4.7 the filtered quaternion $q_{des,t}$ is calculated using slerp interpolation, as proposed by [Laviola2003], along the quaternion sphere. An approximation of $q_{des,t}$ could be obtained by linear (euclidean) interpolation and renormalization:

$$\begin{aligned} \hat{q'}_{des,t} &= q_{lo,t} + \rho\,(q_{hi,t} - q_{lo,t}) \\ \hat{q}_{des,t} &= \frac{\hat{q'}_{des,t}}{|\hat{q'}_{des,t}|} \end{aligned}$$

This approximation introduces small errors into the filtered signal, however the increase of angular RMS errors for the conducted experiments stays below $3 \cdot 10^{-15}$ degrees.

5.2.3 Kalman filtering

Using the system model with and without acceleration a Kalman filter was applied as described in section 4.3.3. The error covariance magnitudes for the system model were taken from the range $[0.0003; 0.02]$, the measurement error covariance magnitudes were varied from 0.01 to 1. Additionally the same prediction method and parameters as used with the FIR scheme were applied to a Kalman filtered signal (see sections 4.3.2 and 5.2.1). The predictive filter had no acceleration in its system model, because the advantage of acceleration is not clear when looking at corresponding noise reduction levels (in fact, Kalman without can achieve the maximum reduction of Kalman with acceleration having a mere tenth of a sample more lag and less overshoot). The results are plotted in figures 5.6 and 5.7:

- Smaller model error and greater measurement error covariance magnitudes yield stronger noise reduction (upper left sub-graphs);
- If acceleration is included in the system model, the filter produces much less overshoot, though at the expense of noise suppression for corresponding parameters;
- The predictive Kalman filter has a sub-sample lag at maximum performance;

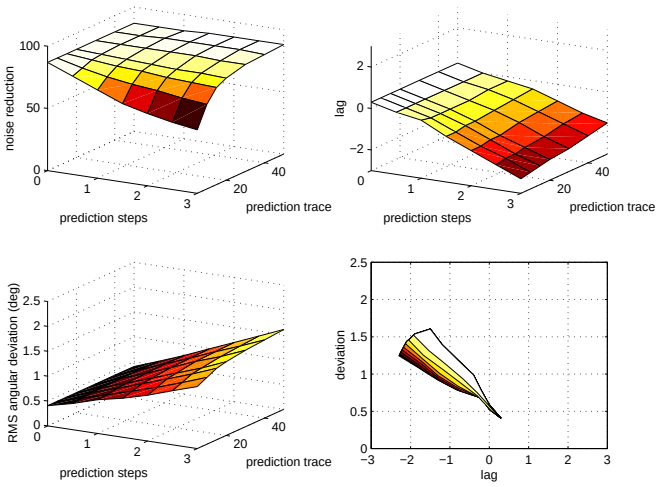


Figure 5.7: *Evaluation: Predictive Kalman filtering without acceleration in the model. Lighter colors represent higher values. Colors in the lower right graph correspond with the colors in the upper left graph - indicating noise reduction performance (0-100%).*

- Remarks about prediction in section 5.2.1 also hold for predictive Kalman, because the same predictor was used.

5.2.4 Computational cost

To indicate the computational cost of each method, time of execution was measured during the experiments (see figure 5.8). Although (even standard non-extended) Kalman-filtering turns out an expensive method, execution times for the lightest and heaviest method did not diverge by more than one order of magnitude.

5.2.5 Euclidean vs. spherical

The hassle with logarithms and exponents may give rise to questions on what is gained with it, compared with quaternion component-wise, euclidean filtering. Therefore the RMS spherical differences (angular deviation, see figure 5.9) between spherical and renormalized euclidean FIR filtered data were measured along with their deviations from the given ideal signal described in section 5.1.3. The angular differences between the methods are small (below one fifth) compared to their angular deviations from the ideal signal. The differences are in the order of 0.01 degree.

5.3 Discussion

The performances of the filters overlap partly and have similar shapes, as seen in figure 5.10 and 5.11. Nevertheless some observations can be made:

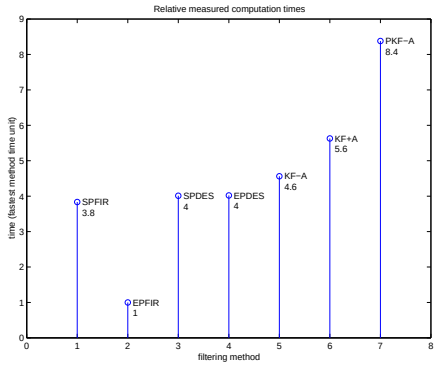


Figure 5.8: *Computational costs of the filtering methods compared: spherical predictive FIR (SPFIR), euclidean predictive FIR (EPFIR), spherical and euclidean DES (SDES/EDES), Kalman filtering with acceleration in the system model (KF+A) or not (KF-A) and predictive Kalman filtering without modelled acceleration (PKF-A). Time unit is the calculation time of euclidean predictive FIR filtering, the cheapest method.*

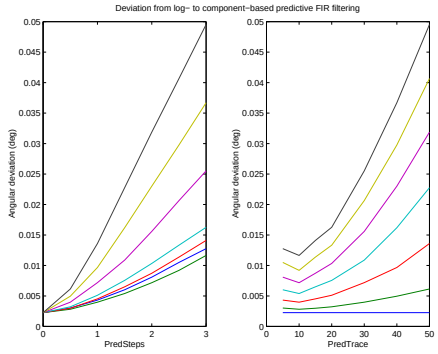


Figure 5.9: *Comparing euclidean and spherical methods: errors due to the use of euclidean instead of spherical evaluation of the predictive FIR filtering method increase as expected when more prediction is required and/or more signal history is taken into account. Nevertheless the errors are of sub-degree order and may not be of practical significance.*

filter	lag (samples)	noise reduction (%)	deviation (degrees)	parameter 1	parameter 2
PFIR	-0.3	89	1.2	3.0	40
PDES	-0.3	86	0.96	0.2	1.5
PKF-A	-0.3	87	0.77	1.0	30
PFIR	-0.5	84	1.0	3.0	20
PDES	-0.8	84	1.1	0.2	2.0
PKF-A	-0.7	84	0.92	1.5	20
PFIR	-	-	-		
PDES	-1.2	82	1.3	0.2	2.5
PKF-A	-1.2	83	1.1	2.0	20

Table 5.1: *Best noise reduction results for a lag just below zero (top rows), best results for 84% noise reduction and overshoot of approximately 1 degree (center rows, minimum lag for PFIR above 80%), best results for heavy prediction (bottom rows, PFIR cannot has no matching result above 80%). Columns "parameter 1" and "parameter 2" show the filter parameters: predicted steps, prediction trace for PFIR and PKF-A, and* α, τ *for PDES respectively.*

- If noise suppression and minimum overshoot are the focus then the FIR filter performs best: none of the other filters can achieve noise reduction of more than 90% percent while deviating a mere 0.2 degrees from the ideal signal, although at a lag of 1.5 samples (which corresponds with the half order truncation of the filter kernel);
- Double exponential smoothing manages to achieve more than 90% for lower lags, though with much more deviation (such as 92%, -0.1 sample lag, 1.2 degrees deviation);
- Extrapolation of the results (see the dotted and dash-dotted lines) indicates that for lags below zero predictive Kalman can achieve lower lags with less deviation than predictive FIR with comparable noise reduction performance;
- For lags below zero double exponential smoothing may be seen as a computationally cheap alternative for the predictive Kalman filter, although Kalman performs better (see also table 5.1). However, predictive FIR is even cheaper, with comparable results for lags near zero.

In [Laviola2003] it is concluded that with a computational cost over a hundred times lower than for (extended) Kalman filtering, predictive double exponential smoothing (PDES) can yield comparable results. However, in the experiments conducted above even a standard (linear) Kalman filtering corresponding with a relatively low computational cost (see figure 5.8) can outperform PDES in the sense of several tenths of a degree overshoot and one percent of noise reduction (see table 5.1).

Euclidean predictive FIR filtering is about 4 times cheaper than its spherical counterpart and PDES, which may advocate its use in applications which do not require much prediction, for instance embedded devices with limited calculation power but fast data acquisition. Speaking of limited resources: although not as cheap computationally, PDES requires only a very small signal memory of 2 quaternion values. An

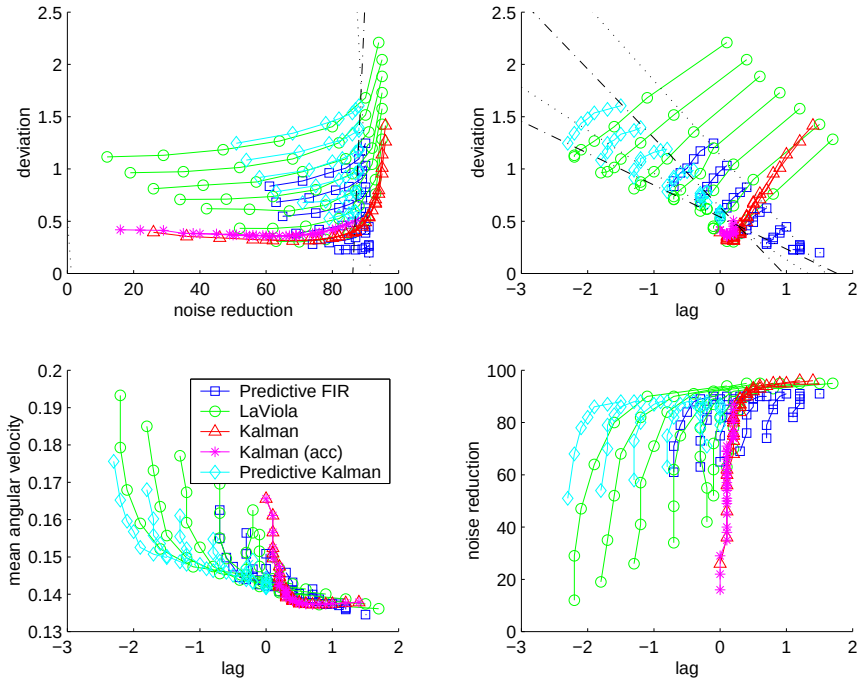


Figure 5.10: *Combined overview of all results. Note the common shape of the curves for the different methods. Connected points have the same amount of prediction (number of predicted samples). In the upper left graph the dotted and dash-dotted lines indicate maximum noise reduction performance for predictive FIR and Kalman respectively. In the upper right picture dotted and dash-dotted lines extrapolate the outline of results for these two filters.*

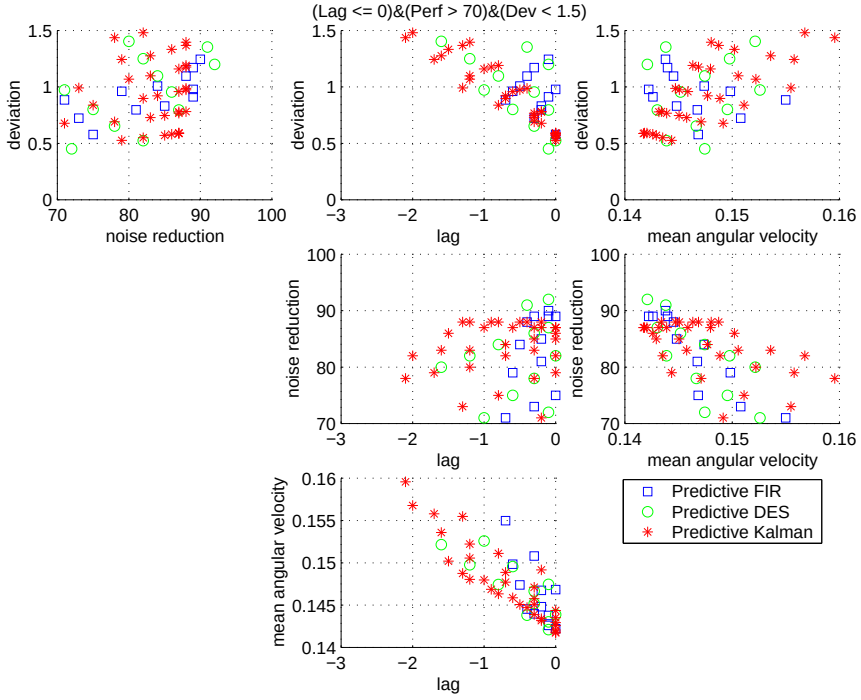


Figure 5.11: *Overview of results showing a lag below zero, noise suppression performance of 70% or higher and deviation (overshoot measure) of 1.5 degrees or less. Filters not adhering outside these requirements are not likely to suffice in a practical situation.*

other disadvantage of predictive FIR and Kalman in the comparison with PDES is that the latter only requires two scalar parameter values (i.e. smoothing factor α and prediction strength τ) to be adjusted to the context in which it is used, whereas kernel width, kernel coefficient distribution, prediction strength and prediction window length and shape are adjustable for PFIR, as model matrices, covariance matrices and the prediction parameters (like PFIR's) are for predictive Kalman.

Tests with the MoVis application (see section 3.2) point out that differences of 2% noise reduction for performance values about 80% are already visually noticeable. Values below 80% leave much of the unwanted oscillations through. Deviations in the neighbourhood of 1 degree seem visually agreeable. In this light predictive FIR does not perform badly at all (see the top row of table 5.1).

Answering questions involves raising new ones: Now that common overshoot (deviation) levels have been quantitatively determined, user tests could shed a light upon the actual impact of overshoot during practical use. Refining the underlying system and noise models could improve rotational motion prediction, although more is likely to be gained from higher sampling rates. The different filtering schemes could also be applied to translations. FIR filtering in tangent space might be applicable to other domains than orientation filtering as well.

Chapter 6

Conclusion

The Personal Space Station is a near-field virtual reality console with a reach-in graspable user interface and is designed to be compact and affordable. Therefore cost effective means are sought to reduce erroneous oscillations present in captured orientation data. This report explores a solution where a signal processing filter is used to achieve this goal. In existing work offline or computationally expensive on-line methods were used to filter orientations or the geometry of rotations was not adhered to. A paper from Lee and Shin did offer a computationally cheap method respecting rotational geometry, which was easily transferable to on-line use.

The quaternion representation of rotations was identified to be a compact, handy and forgiving way to describe rotations and to calculate with them. Coordinate invariant angular differences and velocities became almost trivial to use and it turned out relatively simple and computing power-friendly to stay in or get back to rotation space by renormalization. The errors due to quaternion renormalization were avoided by the use of the Lee and Shin linear finite impulse response (FIR) rotational filter working in quaternion tangent space.

Coordinate invariant linear measures such as angular speed (velocity magnitude) can be used to analyse rotational motion in the frequency domain and relate findings to results in research on human motion such as [VanSomeren1996]. An automatic procedure for designing a FIR filter based on empirical frequency domain data was described and executed. Because of the real-time constraints on the PSS application we explored options to reduce signal lag: the use of asymmetric filter coefficients, a velocity based predictor and the use of alternative filtering methods: Kalman and double exponential smoothing. Quantitative measures were presented to evaluate the orientation filters for their noise suppression, signal lag and overshoot behaviour.

For practical applications renormalization errors might turn out mission-insignificant as the computational cost of going to and from tangent space is not likely to counterweigh error reduction in current technology. The conducted experiments show that almost four times the computing power was traded against an overall signal accuracy improvement of a mere twentieth of a degree spherical RMS error, which is nothing compared to deviations introduced by for instance prediction (as in an order of magnitude).

This is also reflected by the performance of a standard linear Kalman filter on quaternion components in euclidean space. This filter is computationally much cheaper than the (extended) Kalman filter reported of in [Laviola2003], where it was compared to a predictive double exponential smoothing (PDES) method. PDES and Kalman were

said to perform equally, while PDES was calculated more than a hundred times faster than the Kalman filter. The experiments in this report show that the much cheaper non-extended Kalman filter also performs comparably using only slightly more than two times the calculation time needed for PDES.

Predictive double exponential smoothing turned out very flexible and compact in memory requirements and tuning parameters: with two free parameters and two signal values to store during operation it manages to reduce noise heavily at low lags with agreeable overshoot and performs well with lags as low as -1 sample. For lower lags it is advisable to switch to the presented predictive Kalman filter.

Very high noise reduction performance at a neglectable overshoot (of 0.2 degrees) is only achieved by predictive FIR filtering, which is nice because it is the cheapest method, but at the cost of signal lag: this solution is only suitable for applications where the tracked signal is relatively slow or sampling is fast. Using asymmetric kernels 1.5 times the sampling interval should be less than the maximum acceptable lag time, known to be 10 ms for critical hand-eye coordination (resulting in sampling frequencies of 150Hz minimally if noise properties are retained, i.e. constant in the frequency domain on sample basis, not time).

Recommendations for applications such as the PSS:

- Increasing the sampling frequency (like doubling it) has a positive effect (for all filter methods) on prediction performance because it decreases the maximum lag in seconds to start with and because there is less motion dynamics going on between samples.
- Predictive double exponential smoothing is resource-friendly, requires only two parameters to be (maybe adaptively) tuned to the application context and performs only slightly worse than predictive Kalman or FIR at extremes (strong prediction or ultra low overshoot).
- If sampling frequency is higher than 150Hz, predictive FIR filtering would be a good choice, because it is cheapest by far when used on quaternion components directly (which can help to maintain the high sampling frequency) and it reduces overshoot to a minimum. The total end-to-end delay can drop below 10 ms this way.

Bibliography

[Azuma1995] Ronald Azuma, Gary Bishop, *A Frequency-Domain Analysis of Head-Motion Prediction*, Proceedings of SIGGRAPH-95, August 1995, pp. 401-408

[Azuma2001] Ronald Azuma et al., *Recent Advances in Augmented Reality*, IEEE Computer Graphics and Applications, vol. 21, no. 6, November/December 2001, pp. 34-47

[Azuma2002] Ronald Azuma, Gary Bishop, *Improving Static and Dynamic Registration in an Optical See-through HMD*, Proceedings of the 21st annual International Conference on Computer Graphics and Interactive Techniques, ACM Press ISBN 0-89791-667-0, 1994, pp. 197-204

[Berkeley- CS184] Laura Downs, *CS184: Using Quaternions to Represent Rotation*, Web resource for students taking the "CS184: Foundations of Computer Graphics" course at UC Berkeley (spring 2003), *http://www.cs.berkeley.edu/~laura/cs184/quat/quaternion.html*

[DeSmit2003] Bart de Smit, Hendrik W. Lenstra Jr., *The Mathematical Structure of Escher's Print Gallery*, Notices of the AMS, vol. 50, no. 4, April 2003, pp. 446-455 (also see *http://escherdroste.math.leidenuniv.nl/*)

[Dooijes1999] Edo Dooijes, Frans Groen, *Sensor Informatica*, Course notes, 6th edition, Universiteit van Amsterdam, 1999, Chapter 3

[Laviola2003] Joseph J. Laviola Jr., *An Experiment Comparing Double Exponential Smoothing and Kalman Filter-Based Predictive Tracking Algorithms*, IEEE Proceedings of Virtual Reality 2003, March 2003, pp. 283-284

[Lee2002] Jehee Lee, Sung Yong Shin, *General Construction of Time-Domain Filters for Orientation Data*, IEEE Transactions on Visualization and Computer Graphics, vol. 8, no. 2, April-June 2002, pp. 119-128

[Liere2002] Robert van Liere, Jurriaan D. Mulder, *Optical Tracking Using Projective Invariant Marker Pattern Properties*,

Proceedings of the IEEE Virtual Reality 2003, March 2003, pp. 191

[MathWorld- Euler Angles] Eric W. Weisstein et al., *"Euler Angles"*, from *MathWorld* – A Wolfram Web Resource, CRC Press LLC, 1999, *http://mathworld.wolfram.com/EulerAngles.html*

[MathWorld- Euler Formula] Eric W. Weisstein et al., *"Euler Formula"* from *MathWorld* – A Wolfram Web Resource, CRC Press LLC, 1999, *http://mathworld.wolfram.com/EulerFormula.html*

[MathWorld- Lie Group] Eric W. Weisstein et al., *"Lie Group"* from *MathWorld* – A Wolfram Web Resource, CRC Press LLC, 1999, *http://mathworld.wolfram.com/LieGroup.html*

[Mulder2002] Jurriaan D. Mulder, Robert van Liere, *The Personal Space Station: Bringing Interaction Within Reach*, in: S. Richer, P. Richard, B. Taravel, editors, Proceedings of the Virtual Reality International Conference, June 2002, pp. 73-81

[Murphy2003] Kevin Murphy, *Kalman Filter Toolbox for MatLab*, 1998 (updated 2003), *http://www.ai.mit.edu/~murphyk/Software/Kalman/kalman.html*

[NASA- Gimbal Lock] Eric M. Jones, Paul Fjeld, *Gimbal Angles and Gimbal Lock*, Apollo Lunar Surface Journal, November 2002, *http://www.hq.nasa.gov/office/pao/History/alsj/gimbals.html*

[Press2002] William H. Press et al., *Numerical Recipes in C++, second edition*, Cambridge University Press ISBN 0-521-75033-4, 2002

[Shoemake1985] Ken Shoemake, *Animating Rotation with Quaternion Curves*, SIGGRAPH'85 Conference Proceedings, vol. 19, no. 3, July 1985, pp. 245-254

[VanSomeren1996] Eus J. W. van Someren et al., *Gravitational artefact in frequency spectra of movement acceleration: implications for actigraphy in young and elderly subjects*, Journal of Neuroscience Methods 65, 1996, pp. 55-62