Finding Cognitive Strategies for Navigating with 1D-Controls

Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Andreas Nett

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Thomas Seidlv

Registration date: April 1st, 2012
Submission date: Nov 12th, 2012v

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, November 2012
Andreas Nett

Contents

List of Figures

List of Tables

Abstract

Überblick

Acknowledgements

Thank you!

Conventions

Throughout this thesis we use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:

Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Source code and implementation symbols are written in typewriter-style text.

`myClass`

The whole thesis is written in American English.

Download links are set off in coloured boxes.

File: myFile[a]

[a]http://hci.rwth-aachen.de/public/folder/file_number.file

Chapter 1

Introduction

1.1 Interactions: Action and Feedback

Whenever an *interaction* with an input device takes place to complete a task, one can discriminate between an *effector space* and a *task space* for that interaction.

The concept of effector- and task space

We make the differentiation between the two spaces because the input gestures (actions) and the visible change induced in a system take place in spatially and logically separated regions with different properties and characteristic features. To begin, I want to explain the idea of effector space and task space in more detail.

For this thesis, I want to limit the term *interaction* to interactions with physical input devices manipulated by hand, and *feedback* to *visual* feedback. This, for example, excludes camera-controlled interaction techniques and audible feedback.

1.1.1 Differentiation of Task Space and Effector Space

When a user is interacting with a system, the input devices are usually controlled with the end-effectors of the motor system, e.g., your fingers. The type of input device utilized comes down to the effector space. *Effector Space* denotes the spatial region, in which an input device allows the controlling end-effector your motor system to move, so that valid input commands are generated.

Input devices determine the effector space

This way, the *effector space* is naturally determined by the characteristics of the *input device*. The effector space generally does not cover the whole space and degrees of freedom your limbs are able to move in – it is limited by the interactions the input device affords [?]. A comprehensive review and categorization of input devices, more formal and slightly less entertaining than Don Normans book from [?], can be found in 'The Design Space of Input Devices' by [?]. I. Scott MacKenzie et al. review input devices with regard to pointing tasks in [?]. But in this thesis, I will only refer to a fraction of existing input devices – mice and sliders in the main. Before I go on with a description of task spaces, I want to conclude with short examples for effector spaces: For a computer mouse, the effector space would be the flat, horizontal area on your desk the mouse is placed on; when a stylus on a tablet is utilized, the borders of the tablets 'active' area confine the effector space to a rectangular surface. For sliding controllers, as found in a mixer console, the effector space would be defined as a straight line segment of the sliders length.

'The Design Space of Input Devices'

Examples for effector spaces

Characteristics of (visual) feedback determine the task space

Now *Task Space* simply denotes the space in which an interactive system displays its state and presents visual feedback. For example, the display area on a computer screen, or a scale displaying a value, are task spaces. A task space can be spatially detached from the effector space – as a mouse and a computer display are – but it does not have to be; think of the interaction of with a stylus on a touch sensitive display.

In addition to giving visual feedback, the task space plays another important part. Almost exclusively, *goals* that require interactions are defined in *task space*. Goals are *'get the pointer to that button and click it'* or *'reach the bar at the upper edge of the window'* – you do not care where the input device ends up, or know in advance where to put the mouse to accomplish the goal. Two tasks that are semantically completely different in task space could be identical in effector space - in the sense that to complete both tasks, identical gestures are necessary. Take the setup with a computer display and a digitizer tablet in absolute positioning mode: Wether the systems desktop is rendered on a large 30 inch display or a rather small 11 inch display, does not affect the gestures necessary on the tablet, although the instructions for tasks in task space differ in the amount of spacial displacement required.

Goals are defined in task space, not in effector space

Disparity of Effector and Task Space

Looking at the examples for task and effector space makes clear that the apparent properties of the two spaces can differ in various degrees. Next, I want to highlight two important points that are relevant for this thesis, regarding the static and dynamic spatial relation between the two spaces:

Different characteristic concepts of effector- and task space

- The spatial arrangement of objects in task space, i.e., their mutual relative positions and distances at any time, does not necessarily have to be reflected by the spatial state of the input devices elements in effector space.

Static differences of effector- and task space

- This static disparity creates a dynamic disparity that emerges when interactions (input and feedback) take place. The gestures and movements in effector space do not usually result in a *proportionally* sized spatial displacements in task space – not even the direction or the type of movement have to match the input gestures.

...create dynamic differences for interactions

Let me illustrate both issues. For a first example, an 8 inch

Examples for static and dynamic differences digitizer pad, in absolute positioning mode, is utilized to control the curser position on a 24 inch computer display. Here, illustrated in Figure 1.1, the size of the effector space and task space differs by a factor of 3.0, and the same applies to interactions: Displacements of the stylus result in a pointer-displacement three times larger. In a second ex-

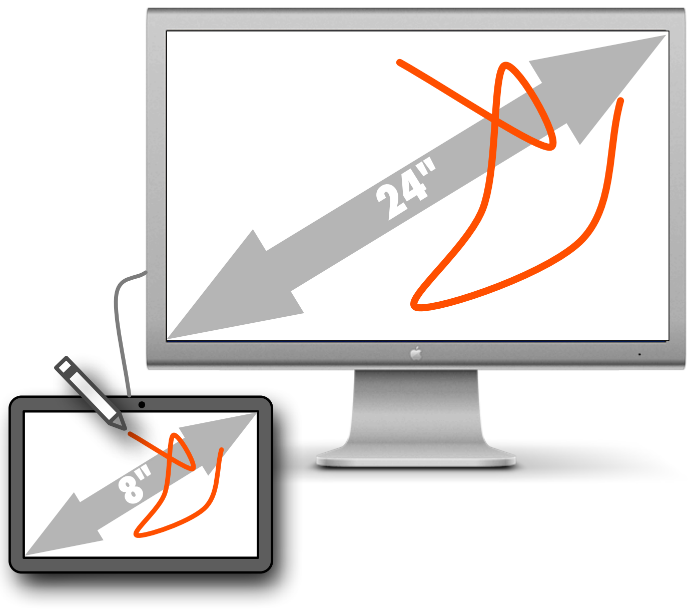


Figure 1.1: An example for an interaction in effector space and feedback task space. Gestures on an 8 inch digitizer tablet result in similar, but larger movements a 24 inch screen.

ample, when a mouse-wheel is used for vertical scrolling on a website, moving your finger downwards on a mouse wheel will move the content on the screen upwards. Here, as illustrated in Figure 1.2, the magnitude and direction of the gesture is different from the spatial characteristics of the feedback.

Displacements of the pointer not identical to device-displacement

In the WIMP setup mentioned earlier, the movement of the pointer on screen is controlled by the movements of the mouse on the table – but in most setups, the operating

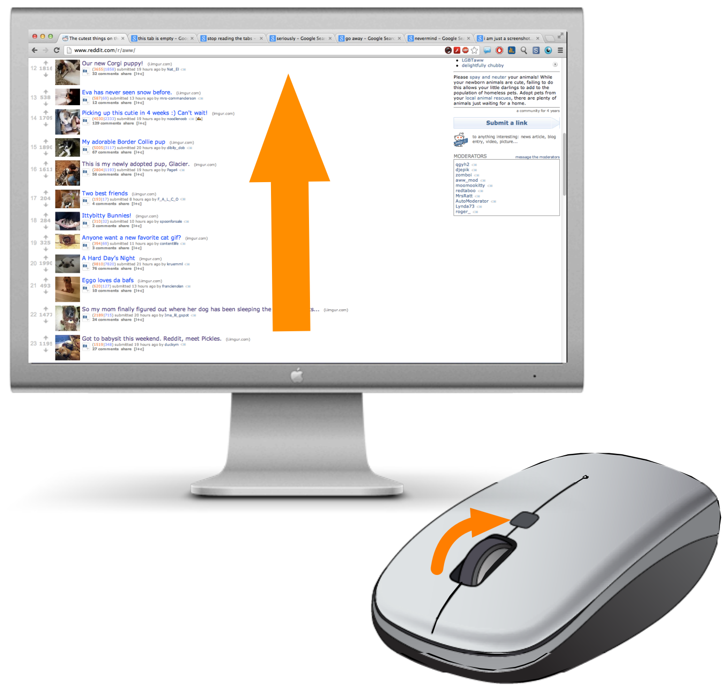


Figure 1.2: Moving the finger downwards on the mouse wheel moves the content of a website upwards.

system does not convert displacements of the mouse, in effector space, to identical, proportional displacements of the pointer. For obvious reasons: The screens area is limited by its borders, and does not match the effector spaces limits (except when your desk is really tiny). Screen sizes and resolutions also vary widely, which is rarely taken into account. In this context, another issue has to be addressed. Developers and users also can deliberately choose a certain factor for the ratio of displacement of pointer vs. mouse. This factor is called C-D ratio (Control-Display ratio). The C-D ratio does not have to be constant: Operating systems, responsible for the management of incoming information about the mouses displacement, can dynamically adjust the C-D ratio depending on the acceleration of the mice, this approach is called *mouse acceleration*. Although mouse acceleration is prevalent in todays operating system, I want to exclude

Mouse Acceleration

these ’dynamically adjusted’ C-D ratios, when they are are put in place by the operating system, and assume that the ’unaltered’ displacement of an input device is processed.

Effector and task space have been introduced as elements of an interaction. For input gestures in effector space there is visual feedback in task space. The characteristics of this feedback, in dependency on the type and magnitude of the input gesture, defines a relationship between the two spaces. Now that we can cleanly distinguish effector- and task spaces of an interaction, I want to further address the relationship between them. We refer to the predefined properties of this relationship as *mapping*.

1.1.2 Mapping

Mapping defines the manner in which an interactive system gives feedback in task space in response to gestures in effector space. Input gestures are executed to alter the properties of a designated object in task space, e.g., its position or size, or to alter the spatial arrangement of a set of objects in task space. In this context, I want to present a formal definition for mapping, as follows:

Definition: *Mapping between Effector and Task Space*

Mapping between Effector and Task Space:

The relation between the type, direction and magnitude of movements in effector space and the resulting alteration of the feedback in task space is referred to as *mapping*.

For a specific combination of effector space, task space and an interaction, the *mapping* can be specified by a *mapping function*, mathematically expressed or verbally described.

When I described the disparity between effector space and task space, specific mapping functions have already been mentioned: By example of gain factors for mice, or for the mapping between the mouse wheel and the movement of the content on screen. Let me give another example of a specific mapping. In the standard-scenario with a mouse on the desk and a pointer on a computer display, one possible mapping could be: *'For every inch the mouse is moved, the pointer is displaced two inches, in the same direction on the display'*. In terms of C-D ratio, or *gain*, this would be represented as a ratio or gain-factor of 2.0. The mapping present in the interaction of scrolling in a text document by turning the mouse wheel could be: *'For every turn upwards or downwards, the content moves 60 lines of text downwards, or upwards, respectively'*.

Examples for mapping.

Summarizing, the environment in which an interaction takes place can be sufficiently defined by describing the effector space, the task space and a mapping. I want to proceed with an outline on how the average performance of users conducting interactions can be analyzed and predicted.

1.1.3 Predictive Models for Performance

When an interaction is examined in view of performance, predictive models might save the efforts of time-consuming experiments and user studies. [**?**]. Predictive models, specific to human-computer interaction, offer a metric of task execution performance, calculated from the tasks parameters. Next, I want to go into the details of predictive models for one task in particular. A fundamental [**?**] task in human-computer interaction, is *pointing*: Moving a pointer from a starting position to target position. Before the first predictive models for pointing tasks were formally proposed, the underlying human motor control was examined by researchers in psychology. The progress in this field is based on the pioneer work of two influential psychologists, who deeply analyzed human motor control and subsequently created first predictive models for voluntary movements [**?**], [**?**]. I want to give a detailed overview on the pioneer work of Robert S. Woodworth (1869-1962) and Paul Fitts (1912-1965) in Chapter 2—"Related Work", but for now, a short introduction that leads to one model in particular, will be sufficient.

Insight into human motor control was the foundation for first predictive models in HCI

Woodworth empirically analyzed the movements of the

Two-component model motor control

upper limbs for fundamental tasks. He presented a two-component model of *'drives'* and *'mechanisms'* [?] that describes the basic human motor abilities (drives) and the *mechanism* that integrates visual feedback in the process of movement. Based on this insight, he suggested a two-phase model for the execution of voluntary movements: In the first phase, an *initial impulse* of rapid movement covers the major part of the distance, bringing the end effector of the limb into the proximity of the target as quickly as possible. In the subsequent *homing phase*, visual feedback of the current spatial relation between limb and target is used to make adjustments to the ongoing movement. The homing phase also covers directional corrections, for example, if the target was missed or the limb was moved too far. Woodworth discovered that the influence of visual feedback depends on the phase it is present in, and that for movement times of 450 milliseconds and shorter, visual feedback in the second phase does not longer benefit the movement process in terms of accuracy at all. (Although these results are greatly) [?], [?]

Woodworth: Two phases for voluntary movements

Benefit of visual feedback depends on phase

Fitts: Performance tradeoff between speed and accuracy

Paul Fitts proposed, based on Woodworth's seminal understanding of human motor control, the idea of performance as a *trade-off* between the *speed* and *accuracy* of a movement. Based on the results of a user study that examined performance in reciprocal tapping tasks, he derived a formula inspired by Shannons theorem for the information-transfer capacity of noisy communication channels [?]. His formula, which became famous as FITTS' LAW, predicted movement time as the result of an *Index of Difficulty* (ID) and external factors. Mathematically, he defined the ID as a logarithmic function of the distance divided by the width of the target. With this definition of ID, performance can be predicted as a *linear* function of ID. His original formula from [?] was improved (for smaller amounts of ID) by I. Scott MacKenzie in [?], who proposed the *Shannon Formulation* of Fitts' Law.

FITTS' LAW:
One of the most commonly used formulations of Fitts' Law is the *Shannon-Formulation*, proposed by I. Scott MacKenzie in [?]:

$$MT = a + b \cdot \underbrace{\log_2(1 + \frac{D}{W})}_{\text{Index of Difficulty (ID), in bits}}$$

a and b are device specific constants, D and W are the distance and width of the target.

Robustness Fitts' Law - for Pointing Tasks

In Fitts' pre-digital–era user study that lead to Fitts' Law, the participants executed tasks with a pen and target areas marked directly on a table. In this case, the **effector space** and *task space* were, semantically speaking, very close to each other, so that the mapping was very direct. Fitts' Law has been extended by Accot et al. for two-dimensional pointing tasks in general [?], and it has been proven to be robust for a wide range of use cases It is widely applied for the evaluation of interactions and user interfaces and can be applied for pointing tasks with a wide range of input devices by adjusting the device-dependent parameters of the equation.

It has also been expanded to be better suited for types of tasks prevalent in modern operating systems, for example, pointing through a bordered path – like cascaded menus. Johnny Accot an Shumin Zhai proposed the *Steering Law* in [?], blending in Fitts' Law into an integral version that operationalizes the difficulty of pointing along a path.

Nonlinear gain not covered by Fitts' Law

Although the area were Fitts' Law generates valid predictions is wide, virtually all applications have in common that the mapping between the task space and the effector space in the examined use cases is direct, or, at most, linear. In fact, although Fitts' Law parameterizes device dependent properties, it does *not* account for nonlinear gaining in the mapping function – for example mouse acceleration.

1.1.4 Simplifying Fitts' Pointing Tasks

There have been several approaches to improve the performance in pointing tasks that have been examined in user studies. I want to highlight the approaches involving *tweaks* of the – traditionally assumed to be linear – mapping function.

First and foremost, *mouse acceleration* is a prevalent and widely implemented concept, applying dynamically adjusted C-D ratio to speed up pointing tasks by increasing the displacement of the pointer in relation to the mouse-gesture dependent on the acceleration. Performance is increased because the system anticipates the distance users want the the mouse pointer to travel. This *anticipation* of the full distance of the ongoing movement is justified, because magnitude of the acceleration, when initiating a movement, is a valid clue for the distance users want to cover in task space. [?], [?]

Acceleration at start of movement indicates distance intended to be covered.

User studies have been conducted by Bootsma et al., Nieuwenhuizen et al., where pointing tasks have been executed in conditions with linear gain and in conditions with a predefined gain function, specifically optimized for the task. In these studies, participants performed significantly better with the 'tweaking' in action. [?], [?],

When the mapping function is not linear, the general handling and tactile feedback of the input device, and therefore the effector space, are unaffected, except for the magnitude of the gestures necessary to complete a pointing task shrinking when higher gain factors (C-D ratios) or optimized mapping functions are applied.

1.2 Media Players: Timeline Sliders for Accurate Navigation

The topic of this thesis was inspired by one setup and interaction technique in particular, which is *accurate navigation* in video scenes. In the following, I want to address issues that arise when users interact with video player interfaces to accurately find a particular *situation* in a movie scene. Of course, not every narrative content is expressed through a course of action that changes the spatial arrangement of visible objects, but I want to focus on *situations* that can be comprehensively described by a particular spatial arrangement of objects that takes place during that scene. For example, a ball being hit by a tennis racket, or a car crossing a line on a street. For the scope of this thesis, I want to exclude any form of audible content alone, or audile feedback during a movie, and concentrate on purely visual content.

Task: Find a specific spatial arrangement in video content

1.2.1 Situation and Setup

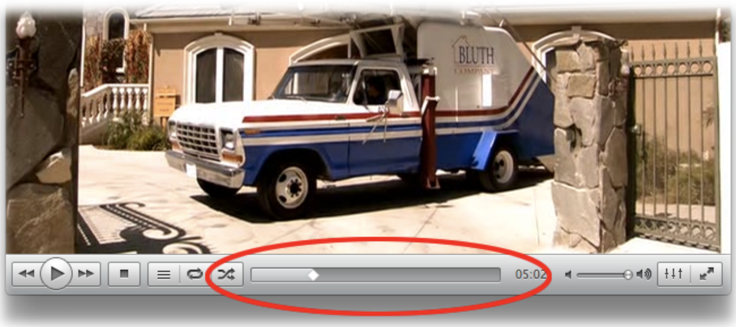


Figure 1.3: The timeline-slider widget in the popular VLC media player application.

By default, almost all video applications feature a timeline-slider–widget as shown in Figure 1.3 (VLC Media Player)[1] , and of course an area of screen estate where the video actual content is displayed in. The general schematic of a typical media player is shown in Figure 1.4.

The length and size of both the timeline-slider and the players *viewport* (the rectangular area where the movie is actually displayed in) are usually either hardcoded in the application, or can be resized with the applications window. When we leave the audio track aside, we can define a movie scene as a *sequence of frames* that, when played back, presents the

Narrative content hardcoded in sequence of frames

[1]http://videolan.org

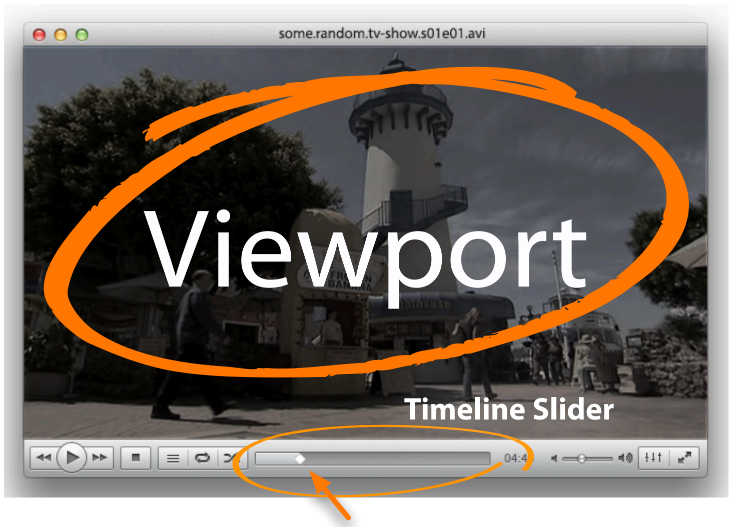


Figure 1.4: Setup of a media player, consisting of a viewport to show the content in, a timeline slider with knob (arrow), and several additional widgets for input and feedback.

movies *narrative content*. Besides pressing the playback button, sitting back and watching a movie, users can drag the timeline-sliders knob to navigate forwards and backwards in the sequence of the movie, while the viewport steadily updates to a frame corresponding best to the sliders updated position. At this point, we assume the media player is implemented this way, and that the implementation works flawless and quick, to update the viewport steadily and free of delay, when the slider is dragged.

With this agreement on a certain type of video player application and behavior, I want to describe video players in terms of *effector space*, *task space* and *mapping*.

Presence of Task Space and Effector Space

When we observe the interaction of *dragging the slider to manually control the movies playback*, both *effector space* and *task space* can be recognized. The concept of distinguished

Two ‘layers’ of mapping: mouse — widget and widget — content task space and effector space can be found twice for this interaction, i.e., in two layers. In first layer, the *effector space* according to definition in 1.1.1, is determined by the input device (let's say, a mouse), and the task space is formed by the media players sliding-widget that works as a timeline-slider. The second layer is present between the slider widget itself and the *content* of the movie scene. The timeline slider exclusively affords horizontal movements and forms the effector space, while the task space displays the frames of the video scene that are indicated by the timeline-slider widget. Goals in this task space are cases of *accurate navigation* – for example, *'find the situation where the guy drops his ice cream'* or *'find situation where the race car collides with the other car'*. For these tasks, the timeline slider has to be moved to the correct position, thus invoking the display of a frame that shows the desired content. I want to assume a setup that enables the mapping in the first layer, between the users hand and the slider-widget, to be as as direct as possible. For example, setups where the slider-widget is operated with a pen on a touch screen, or even with a hardware-based sliding controller device that affords the same input gestures as the timeline slider widget. Based on this assumption, I do not want to attend to the interaction of controlling an intermediate element of an application interface, but focus on the interaction of users navigating in video content with one-dimensional sliders.

Task space = content, sequence of frames

Ignore the first 'layer' of mapping, assuming it is very direct

1.2.2 Mapping between Timeline Slider and Content

Now that the effector space and task space for accurate navigation have been established, I want to return to the interaction itself and explain the mapping. To further describe the mapping between the input device and the viewports content, it is necessary to make a short excursus into the technical details of video player applications.

A movie scene is understood as a sequence of frames that are displayed in rapid succession, in a fixed order and in fixed intervals. Every frame has a timestamp, which corresponds precisely to the point in the movies duration where that one frame is shown. The timeline slider in a video player application can be seen as an array of discrete positions. When the application is set up to show a video, it assigns a subset (in fixed intervals) of the movies frames to the discrete positions of the timeline slider. (Figure 1.5)

Frames in fixed intervals are assigned to positions on the timeline slider

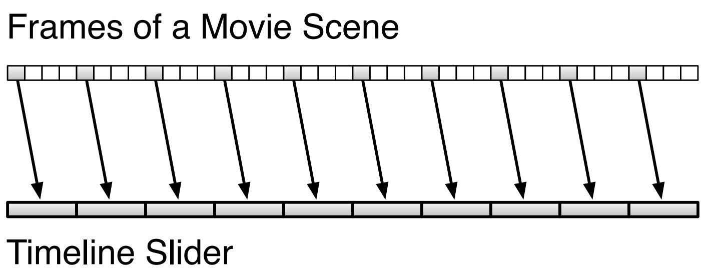


Figure 1.5: At intervals, a subset of a scenes frames are assigned to discrete positions of the timeline-slider.

When the video applications plays back a video, it continuously updates the viewport with the subsequent frame, and moves the timeline-sliders knob as as the movie progresses. Thus, the speed of the timeline sliders knobs displacement is determined by the movies length (temporal) and the sliders length (spatial). On the other hand, when the timeline-slider is *manually dragged*, the application turns off the automated playback. Instead, the application steadily updates the viewport with the frame that is assigned to the sliders current, discrete spatial position. By dragging faster or slower (left or right – forwards or backwards), the user can adjust how long each of the assigned frames is shown, before the viewport is updated with the next – or previous, respectively, frame. Actually, this behavior might be somewhat implementation-specific. But for example, the VLC Media Player[2] and the Apple Quicktime Player[3] player behave this way. Next, I want to describe how an accurate navigation task looks like in this environment.

Dragging the slider updates the viewport with the frame assigned to that position.

Implementations might vary, we assume the described one

Through the frame-wise progression of a scene, visible ob-

Narrative defines trajectories of depicted objects

[2]http://videolan.org
[3]http://www.apple.com/quicktime

jects in the scene move along fixed trajectories, which are predetermined by the narrative content of the scene. Therefore, during playback, the mutual spatial arrangement of objects, visible in the media players viewport, changes. Moving the timeline-slider *does* control the advance of objects, forward and backwards on their trajectories, but the vector (direction and amplitude) of motion on the trajectory is encoded in the video scene itself. In other words, the displacement of the sliders knob is only a factor for the predetermined displacement of in-scene objects on their trajectories. For example, in two scenes with object trajectories differing in momentum, equal displacements of the slider will cause the objects to advance differently on their trajectory. The two scenes (A and B) shown in Figure 1.6 show two identical displacements of the slider, which effectuated two different displacements of the ball. In terms of *gain factor*, scene A features a gain factor of 1.0; the in-scene object (a ball) moves in the same manner the slider is moved, while in scene B, a gain factor of 2.0 is present.

Feedback for input with slider encoded in narrative

Linear gain created by narrative

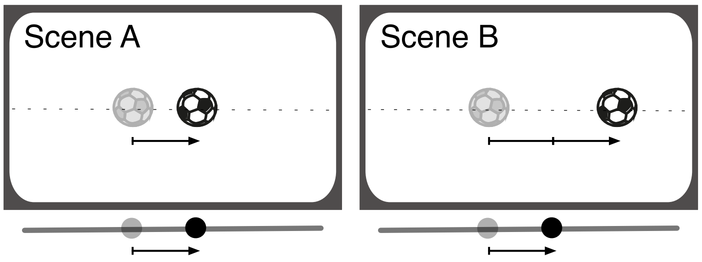


Figure 1.6: Two scenes with a different narrative content: In scene B, the ball moves twice as fast as in scene A. Although the slider is displaced by equal amounts in both scenes, the resulting displacement of the object is different.

Instead of *linear gain factors* as depicted in Figure 1.6, the narrative content of a scene can result in *nonlinear* gain. When nonlinear gain is present, the distance that an object will advance on its trajectory will depend on the absolute position *where* the timeline-sliders knobs displacement takes place. Therefore, the same gesture can have different effect on an object, depending on the absolute advance of

Nonlinear gain created by narrative

Example for nonlinear gain

the object. Also, the ratio of slider-displacement vs. object displacement can change during steady movements of the slider. Figure 1.7 shows two examples. In the left scene, the ball's momentum increases as the scene progresses. The further the timeline-sliders knob is moved, the higher the gain factor becomes. (It is the other way around in the scene depicted on the right.) Of course, the examples of linear and nonlinear gain shown here are just very synthetic examples for the manifold of possible narrative content, but they can easily be derived from natural content: Imagine a rolling ball loosing momentum on the grass, a race car accelerating, or a train coming to a halt.

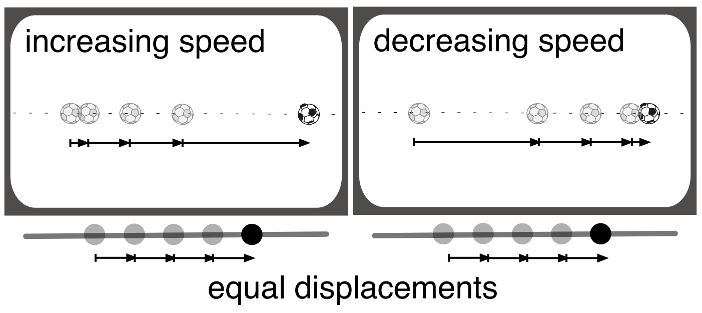


Figure 1.7: Two scenes with simple examples of nonlinear gain. As the scene progresses, the ball picks up speed (left) or slows down (right). This steadily changes the gain factor present in the mapping between the slider and the in-scene object.

1.2.3 Accurate Navigation in Video Scenes with Timeline-Sliders

Spatial mapping between slider and content is arbitrary

Summarizing, the mapping between the timeline-slider and the content in the players viewport may contain linear or nonlinear gain. When a user wants to navigate in a scene, to reach a particular spatial arrangement of objects in the viewport, the gain factor – linear or nonlinear – has to be accounted for. Examples for interactions with linear gain have been shown in Figure 1.6. Figure 1.8 shows the course of an interaction where the user operates the slider at a constant

Nonlinear, increasing gain might lead to overshoots speed. The user want to navigate to that specific situation in the scene, where the red object, moving on the dotted trajectory, is located in the area marked green. Initially, the gain factor between the slider and the object on its trajectory is low, the object (red) moves only slightly. While the user keeps on moving the slider, the gain factor steadily increases, and the object displaces more and more in response to the steady gesture in effector space. Eventually, the red object overshoots the green target position, because the user did not account for the increasing gain.

From Pointing to Accurate Navigation in Video Scenes

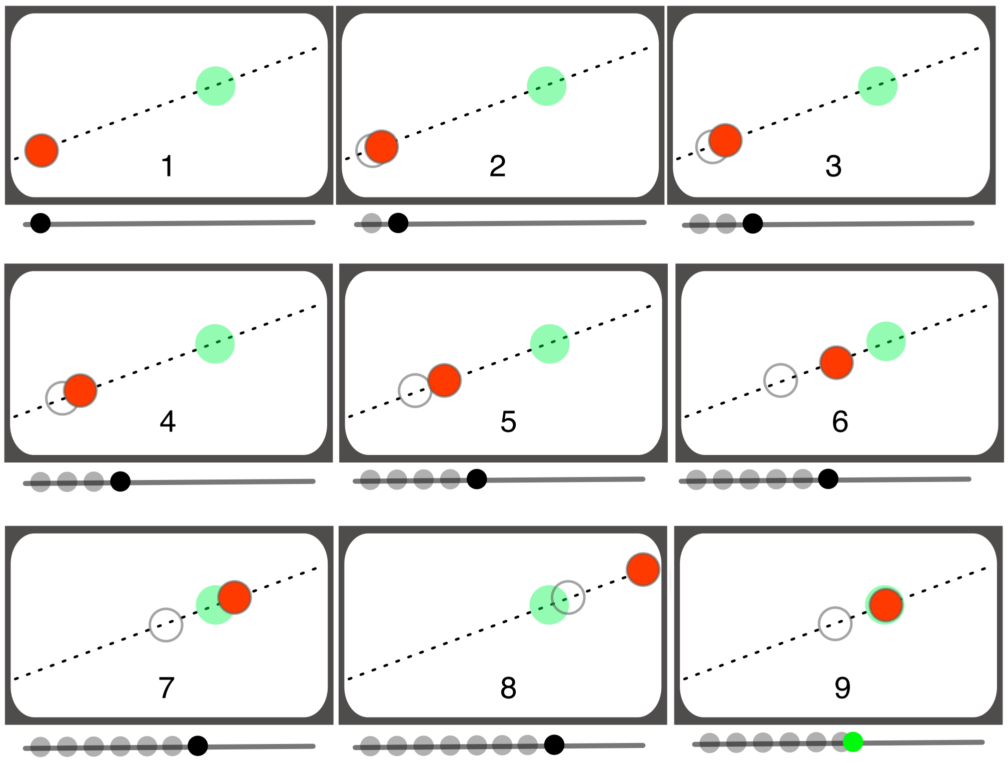
The interaction with a one-dimensional input device for accurate navigation under the presence of nonlinear gain will be further examined in this thesis. First, I want to summarize the commonalities and differences of classical pointing tasks and accurate navigation tasks.

In a 'classical' pointing task, the interaction takes place in an environment, where the mapping is very direct (physical pointing with a limb) or includes linear gain (WIMP setup, excluding mouse acceleration). The user is in control of the pointer, and gestures in effector space cause feedback in task space that is always reproducible, regardless of the absolute positions of the pointer.

Similarities between pointing and accurate navigation...

It has been shown that the difficulty of moving a pointer from a starting position to a target position depends mainly on the distance and the width of the target area. [**?**] The task of accurate navigation has similarities to a classical pointing task: The user wants an object, currently located at a starting position, to move along its trajectory until it reaches a target position and then stop the advance. But in this case, the trajectories of the object are encoded in the narrative content, and the only means of input is a one-dimensional input device with absolute positioning. While the user is executing the input gesture to that is supposed to bring the object into the target area, the gain factor may change, and the user has to adapt the gestures to account for

...and differences



s

Figure 1.8: Mapping between the timeline slider and a scenes narrative content: This figure shows video frames, corresponding to the position of the timeline slider underneath. For the frames labeled **1** to **8**, the sliders knob has been displaced in equal steps. The red ball indicates an object in the movies scene, moving along the dotted line. The green area is a *target area*, where the user wants the red ball to be. The user displaces the slider steadily, but the narrative content of the movie determines the red ball to pick up speed. Between frames **6** and **7**, the user moves the slider too far, and misses the target. Frame **9** shows the supposed-to-be location of the sliders knob.

altered gain. If the gain factor changes during the interaction, and the user does not take the new gain factor into account, two things can happen: If the gain factor increases, and the movement of the slider is not shortened, the object might *overshoot* the target area (Compare Figure 1.8). If the gain factor decreases and the user continues to move the slider as initially planned, the object does not move as far as intended. In both cases, correctional movements are necessary, and that potentially increases the overall task completion time.

In summary, it can be stated that in both pointing tasks and accurate navigation tasks, a certain spatial arrangement of visible objects is desired. In both tasks, the user operates an input device (or, moves his limb, for physical pointing tasks) to shift the spatial arrangement towards the desired state. In classical pointing tasks, the mapping is direct and linear, or even optimized (mouse acceleration). On the other hand, in accurate navigation tasks, the mapping is governed by the movement of objects on trajectories that are encoded in the videos narrative. Also, in virtually all available interfaces for vide playback, the means of input are reduced to a one-dimensional sliding controllers.

1.2.4 Inadequacies of Timeline Sliders

More frames than pixels on slider widget?

Timeline sliders have two major inadequacies for accurate navigation. One problem arises simply from the fact that a movie scene usually has more frames than there are discretely selectable pixels on the timeline slider widget. (Compare Figure 1.5, the scene has far more frames than the slider has discretely selectable positions.) This problem has been addressed by several research projects, with the goal to implement enhanced video players with additional interaction techniques. Two examples are *PVSlider* and *TLSlider*, presented by [**?**] and based on ideas from [**?**]. The PVSlider widget interprets the vertical distance between the mouse pointer and the slider as a refining parameter for the desired scrolling speed. The TLSlider implementation dynamically expands the timeline slider spatially, at any location, to locally enable a finer positioning. Both widgets allow users to give their gestures more meaning: By signaling a C-D ratio to the media player, users can chose how fast to skim through frames in response to an input gesture. These approaches are helpful for frame accurate navigation, but leave out the previously introduced issue due to nonlinear mapping. This problem was approached by researchers with *direct manipulation interfaces*.

1.3 Direct Manipulation Interfaces

The content of a movie scene, defined by the spatiotemporal interaction of in-scene objects, issues a new level of indirection from users interaction with the timeline-slider. Different from classical pointing tasks, users have to take nonlinear gain into account, and possibly adjust their initial gestures during task execution, as the original gesture would either overshoot the target, or would have to be extended. With the motivation in mind that this additional burden on a user decreases both the perceived user experience and objective task performance, several research projects addressed this issue. In the context of media players, implementations of the *direct manipulation interface* concept ([**?**] [**?**]), allow users to bypass timeline-slider widgets entirely.

Dragging objects, not sliders

Instead of implementing enhanced timeline-slider interfaces, direct manipulation enables users to directly interact with the objects visible in a scene. With these direct manipulation video-player–interfaces, users can click objects and drag them along on their trajectories until they are located in a desired target position, as part of a desired spatial arrangement. As the user clicks on an object and moves the pointer, an underlying algorithm steadily looks up the corresponding frame were the object is in the current position of the pointer, and instantly forwards or rewinds the movie to the appropriate timestamp.

Replacing indirect mapping by direct mapping

The specific user interfaces and visual presentations of the interaction have been implemented differently in the existing research projects. But they all have in common that they replace the indirect, potentially nonlinear mapping between the timeline-slider and the in-scene objects with a direct mapping between mouse and in-scene object. By applying the concept of direct manipulation to accurate navigation tasks, the indirect mapping between gestural manipulation of the slider, and the spatial movement of scene-objects vanishes.

Important research research projects with prototype applications for direct object manipulation video players are:

- **Trailblazing**, a system enabling direct manipulation based on object recognition, suitable for multi-camera environments.

  Developed by [**?**]

- DRAGON[4] , based on precomputed optical flow fields, developed by Thorsten Karrer in [**?**], and extended by Moritz Wittenhagen with **DragonEye** [**?**].

- DimP[5] , presented by [**?**].

All research projects with implementations of direct manipulation interfaces evaluated their application in user studies. [**?**], [**?**], [**?**]) The user studies compared user performance for navigation in video content with traditional timeline sliders to the performance with direct manipulation. In all user studies, participants performed significantly better (in terms of speed) with direct manipulation interfaces.

1.4 Research Questions

The fundamental motivation, for the pursuit of direct manipulation techniques in video applications, is the idea that the indirect, ambiguous mapping between effector and task space places an additional burden on users.

This is reflected in the results of user studies, that have been conducted by researchers with direct manipulation interfaces, where participants not only performed significantly better with direct manipulation, but also stated that this interaction felt more natural. These results corroborated the hypotheses constructed prior to the experiments, which stated that task performance will increase with direct manipulation techniques. The focus in this research projects was on testing and eventually proving the superiority of direct manipulation versus traditional navigation with timeline-sliders.

[4]http://hci.rwth-aachen.de/dragon
[5]http://www.lri.fr/∼dragice/dimp/

In my thesis, I want to further investigate the – presumably – negative impact of nonlinear mapping in accurate navigation tasks. One approach will involve the review and evaluation of existing models. In the second approach, I will evaluate the results of a user study that I conducted for this thesis with the goal to pinpoint the reasons for degraded task performance.

I will conclude the introduction with an outline of my user study.

1.4.1 User Study

Quantitative user study

In a quantitative user study, developed, conducted and evaluated during the course of this thesis, participants were challenged to complete a series of tasks. The presented tasks resembled synthetic, abstracted versions of accurate navigation tasks. The abstraction to synthetic classes of tasks was necessary to reduce the manifold of actual video content to a portion that can be operationalized and evaluated.

Five targets in effector space

Different mappings were tested

Tasks with five different levels of difficulty (in terms of ID, see [**?**] or) in effector space were presented, by selecting five different spatial target positions on a hardware slider. Participants were then asked to move the slider into the target position. By presenting the visual representation of both the sliders current position and the currently requested target position on a computer screen, it was possible to introduce different types of mapping and differently visualized task spaces. Each target, from the set of five in effector space, was requested multiple times, under different conditions for task space and mapping. This approach allowed to measure and evaluate the impact of nonlinear, ambiguous gain for tasks that, in effector space, require identical gestures to complete them.

By examining the patterns of effects that degrade performance, I tried to highlight the underlying cognitive structures that prevent users from adequately adjusting to nonlinear, ambiguous gain. The results of this study underline the motivation for direct manipulation interfaces, in particular for media player applications, by empirically showing that, for accurate navigation tasks, the mapping has to be taken into account to state the difficulty of these tasks.

Chapter 2

Related Work

The field of research relevant for this study can be divided into two parts:

- **Theoretical Research** and
- **Practical Approaches**

In this chapter I give an overview on past achievements and current advances in HCI-research, for both the theoretical and the practical direction.

In the first part of this chapter I will describe the origins, refinements and generalizations of models relevant for my thesis, in particular for pointing tasks, value setting and navigating in media. In the second part I will give an overview on the practical approaches to improve user performance for these tasks.

2.1 Theoretical Research: An Overview on Models

Theoretical research in HCI is, for the most part, concerned with understanding the underlying cognitive and sensomotoric concepts that can explain the characteristics of an interaction in terms of performance and applied strategies.

Predictive models: For development and evaluation

Ideally, descriptive and predictive models emerge from research. These models can be applied to explain real life situations - because the model generates helps to design, analyze, compare and *prove* the quality of an interactive system.

Models from user study data

In general, models can be created through examining and recombining the - suspected or known - underlying cognitive and sensomotoric structures, or by conducting a user study. For these user studies, the characteristics of the interface, the interaction and the goal are parameterized. Subsequently statistical methods are used to to generate a generalized, fitting model for the data.

HCI research based on foundation of human psychology

This field of HCI-research is strongly based on the results of seminal studies in human psychology, a field much older than the relatively modern field of HCI. For example, the pioneer work of Robert S. Woodworth, *'The Accuracy of Voluntary Movements'* from [**?**], dating back over one century, explains how visual feedback is beneficial for motoric tasks.

Woodworth's two-component model

Woodworth was the first to propose a *two-component* model for pointing tasks, stating that the full act of a voluntary movement is comprised of a *ballistic* phase (initial impulse) and a following phase for homing in on the target (*current control phase*). In his experiments, he precisely controlled the start and end of visual feedback during a pointing movement by extinguishing the light source during a task, and measured the impact on the movement time and error rate. This allowed him to derive the impact of visual feedback during the process of the movement dependent on the movement phase For a review on his experimental setup and several advanced models directly based on Woodworth's research, see Elliott et al. [**?**].

Fitts' Law

Fifty years later, in 1954, Paul Fitts extended the work of Woodworth by established a model for the act of pointing in continuous tasks [**?**], stating that *performance* is a *trade-off between speed and accuracy*. By parameterizing distance and the necessary accuracy of the target, as well the characteristics of a pointing device (if one was used used), Fitts described the simple *predictive model* that quickly gained fame in HCI-research as **Fitts' Law**.

Fitts' user study: Index of Difficulty (ID)

Fitts conducted a user study to examine the performance in repetitive (reciprocal) tapping tasks, with a stylus and two target positions. Participants were asked to move the stylus back and forth between both target positions, as quickly as possible. He observed that, when the distance was increased, the *movement time* (MT) increased. When he increased the target width (which determined the necessary accuracy), movement times decreased. Based on this insight and the evaluation of his user study data, he proposed a term to define the difficulty of aimed movements. He stated that the *Index of Difficulty* (ID) can be calculated from the distance (D) to the center of the target and the width of the target location (W) by:

Speed-accuracy–tradeoff

$$ID = \log_2 \frac{2D}{W}$$

In Fitts' results, movement times could then be explained by the linear formula

$$MT = a + b \cdot ID$$

...where *a* and *b* are device-dependent parameters that can be derived from measured data via linear regression.

Fitts' original model formula was refined by I. Scott MacKenzie [?], who extended Shannons theorem for the information-transfer capacity of noisy communication channels. [?]) His *'Shannon-formulation'* is widely used today:

I. Scott MacKenzie's *Shannon Formulation* of *Fitts' Law*

$$MT = \quad a \quad + \quad b \cdot \underbrace{\log_2 (1 + \frac{D}{W})}_{\text{Index of Difficulty (ID), in bits}}$$

Robert S. Woodworths (1869-1962) and Paul Fitts (1912-1965) pioneer work in the field of human psychology and motor control did produce influential models that have been proven robust enough for the transition into the digital age. To put things into perspective:

- Woodworth conducted his user study in [?] with paper on a rotating barrel that recorded the movements of a pen participants moved across a slot.
- Fitts fundamental law originated from user study in [?], the year the first silicon transistor was produced.

Fitts' Law might be, as Woodworth's work was before, one of the most cited, approved and refined models in HCI.

Fitts' Law applicable in WIMP-setups

For example, by elevating the applicability from one-dimensional pointing tasks (on a straight line) to navigating through laterally bounded paths, Johnny Accot and Shumin Zhai proposed the *Steering Law* for trajectory-based interactions [?], and later refined Fitts' Law for bivariate pointing, allowing to factor in the shape of target in a two-dimensional pane [?]. Both comes in handy for evaluating modern user interfaces, in particular the WIMP (Windows, Icons, Menus, Pointer) environments common these days. Guiard et al. showed that Fitts' Law is robust for very high values of ID, when the task requires very fine grained movements with input devices that offer a high spatial resolution. [?]

Fitts' Law applicable for high precision tasks

One important aspect of Woodworth's and Fitts's research, as mentioned above, was the analysis, not only of the overall task execution times in relation to the difficulty, but of the movement patterns that were observed. This approach was further investigated by Bootsma, Fernandez and Mottet in [?]. They observed that recurring patterns in the kinetic profiles of movement change their shape depending on the difficulty of the task.

While these approaches, in their original form, aim at explaining the performance for specific tasks, the *Model Human Processor* by Stuart Card, Thomas P. Moran and Allen Newell [?] proposes a multi-purpose model of the human cognitive structures.

CMN-Model of human cognition

Card, Moran and Newell divided the system of human cognition, the sensor and motor systems processes as a whole, into discrete, individual components. They came up with a *table of costs* (in time duration) for each utilization of the subsystems, allowing to predict the time to complete any arbitrary interaction by breaking the task down into small, basic building blocks until they match the purpose of a subsystem. This concept gave rise to a variety of more specialized models, to simplify the application to classes of tasks common in HCI, i.e., for typing. (*KLM-GOMS*, *Keystroke-Level Model GOMS* by [**?**])

These models are widely used as tools to analyze and evaluate interactive systems and to predict, analyze and explain user performance. Also, the research that lead to the declaration of these models opened insight into the workings of the sensor system, cognitive structures, and the motor system.

2.2 Overview on Practical Approaches: Making Tasks Easier

While mapping, between the timeline slider and the spatiotemporal structure of movie scenes, states a problem in frame-accurate navigation tasks, researchers and developers have investigated ways to improve performance in related tasks, by deliberately tweaking the mapping involved. I want to present two exemplary research projects that explored the effect of non-linear gain, induced in a user study setup with pointing tasks.

Mapping as a Performance-Enhancing Factor

With the emergence of computers in everyday life, for production, personal and casual use, and the upcoming of graphical user interfaces and the WIMP setup, Fitts' Law has been used to evaluate the pointing tasks occurring in modern interfaces. It also inspired the research to improve the usability of widespread interfaces. I want to show two research projects that experimented with mappings introducing non-linear gain to pointing tasks.

Mouse Acceleration

A fundamental and widespread concept involving adaptive and nonlinear mapping is *mouse acceleration*. The option for mouse acceleration is present in most operating systems for WIMP setups. Mouse acceleration is defined by an adaptive control-display (C-D) ratio, that increases for faster accelerated displacements of the mouse in effector space, and decreases for slower movements. This is based on the idea that the magnitude of the acceleration phase, when initiating a movement, is a valid clue for the distance users want to cover in task space. [**?**], [**?**], [**?**], [**?**]. The benefits of adaptive C-D ratios for pointing tasks have been shown in several studies for different contexts. [**?**], [**?**]

Benefits of adaptive C-D ratios

Semantic Pointing, dynamically decreasing gain near target

In ***Semantic Pointing: Improving Target Acquisition with Control-Display Ratio Adaptation***, [**?**] presented a user study in which participants had to complete discrete, one-dimensional pointing tasks. A puck on a digitizing tablet served as an input device. The task space was formed by a computer display, showing a vertical line line that was controlled by the pucks position on the tablet, and a target area the line had to be moved into. With this setup, different mappings between the displacement of the input device and the displacement of the line on screen were tested. The non-uniform C-D ratio applied increased the displacement of the line, in reaction to the pucks movement, when it was further away from the target position, and decreased the displacement when as it came closer to the target area. Thus, the distance and width of the target area was different according to whether it was determined separately in effector space or task space.

The hypothesis was confirmed, that task difficulty was, for the main part, defined by the tasks definition in effector space – by the difficulty of the movement necessary to accomplish the task – and largely independent on the proportions of the visual representation. This was an important conclusion, because common models to predict the performance in pointing tasks rely on a measurement of task difficulty operationalized by distance and width of the target, for example Fitts' Law.

Based on this conclusion, Blanch et al. reason that performance in pointing tasks, typical for WIMP setups, can be increased by tweaking the C-D ratio depending on the mouse pointers position in relation to click-able objects on screen. The C-D ratio should be decreased near or over locations in task space that users are more likely to visit, making it easier to precisely home in on, e.g., a *'Save File'* button. On the other hand, near and over critical options, e.g., *'Discard Changes'* buttons, the C-D ratio should be increased, making it harder to select these options.

Nonlinear gain in a reciprocal tapping task

In their contribution from [?] (***'Making Fitts' task a bit easier'***), Laure Fernandez and Reinoud J. Bootsma used a mapping function with nonlinear gain, optimized for the expected movement patters in pointing tasks.

In a study with *reciprocal pointing tasks*, similar to the primary study that lead to Fitts' Law, they compared two conditions with linear gain and a predefined mapping function that was *optimized to improve the movement time and accuracy*. In a setup with a tablet and stylus and a pointer and with two target areas on screen, Fernandez' and Bootsma used the same approach as Blanch et al. to create the mapping function: The closer the pointer was moved to the target areas, the smaller the C-D ratio became. (*logistic mapping function*) As their setup for a reciprocal pointing task was symmetric, the function symmetrically scaled the C-D ratio for both end points, left and right as well, with the highest C-D ratio in the middle between targets in task space.

Different from the approach in [?], Fernandez and Bootsma evaluated their results not only based on task completion times, but *analyzed the movement profiles* or *kinematic patterns*.

The user study showed that, with *logistic mapping*, movement times significantly improved, and an analysis of the *kinematic patters* showed that participants recognized the spring like behavior of the system when logistic mapping was present. [?]

Chapter 3

User Study

3.1 Introduction

According to the questions defined in Chapter 1.4, a quantitative user study was conducted. The user study took place in September and October, 2012. Rooms and material of the Chair for Computer Science 10[1] at RWTH Aachen University[2] were used for the course of the user study.

Structure of this chapter

I will begin this chapter with a short summary of the motivation for the user study, followed by an introduction to the tasks that were created for the participants. After giving an overview on the framework for the experiment, covering both the hardware and software aspects of task presentation, I will give a detailed description of the conditions. Then I will describe the procedure for the user study, and conclude with a summary on the conditions.

In Chapter 1, one specific task was described: accurate navigation in movie scenes. The common method these applications offer for video navigations are timeline-sliders. Contrary to classical pointing tasks with linear gain, the mapping between the timeline-sliders widgets (effector space) and the advance of objects on their trajectories (visible in the

[1]http://hci.rwth-aachen.de
[2]http://rwth-aachen.de

viewport) can include nonlinear gain. When users interact with the one-dimensional sliding controller, they have to account for the possible presence of nonlinear gain, and adapt their gestures to changes in the gain factor as they move the sliders knob.

Researchers have come up with a new interaction technique, by adapting the concept of direct manipulation, introduced by Ben Shneiderman [?], to video navigation. The newly proposed interfaces allowed users to control the advance of objects by *directly manipulating* – dragging – objects along their trajectories, thus bypassing any arbitrary nonlinearity between displacements of the timeline slider and displacements of objects in the scene. Experiments with user studies have shown empirical evidence that these direct manipulation interfaces allow users to perform significantly better, compared to interactions with timeline-sliders.

Only empirical evidence that direct object manipulation is superior

In my thesis, I want to investigate the impact of different characteristics of nonlinear gain on task performance. Therefore, I designed a series of synthetic tasks with the goal to observe this influence. In the following, I want to describe the tasks presented in the user study.

3.1.1 Deriving Tasks for the User Study

In theory, an accurate navigation task is simple: The particular spatial arrangement of objects a user desires to navigate to, has a particular timestamp or a particular range of contiguous timestamps. The user simply has to move the timeline-sliders knob to the spatial position on the sliding distance that corresponds to a timestamp in the correct range. An easy task – but the target is *not defined* in effector space. The user does not pay attention to the spatial setting of the timeline-slider, visual feedback [?] is not obtained from the spatial arrangement of the timeline-sliders knob. The users attention is solely on the computer screen, observing and interpreting the displacement of the objects in a scene, in reaction to his manipulation of the timeline slider. Until the correct spatial arrangement of objects is found, this feedback is used to determine the next actions. This was also confirmed in a pilot study during the development phase of my user study: Participants focused entirely on the computer display, and, for example, did not notice that the same target in effector space was requested several times in a row. (This came up when I checked for the presence of learning effects, which I was worried about because only five target positions were requested overall.)

Although this task can be operationalized completely in effector space [**??**], the difference created by the mapping plays a significant role in the outcome of the task in regard to completion time and the pattern of movement applied. (Compare user studies in [**???**] Contrary to reciprocal pointing tasks, the tasks in this study were discrete, and designed to resemble accurate navigation in video content, where a timeline slider (effector space) - is used to alter the arrangement of objects on trajectories (task space).

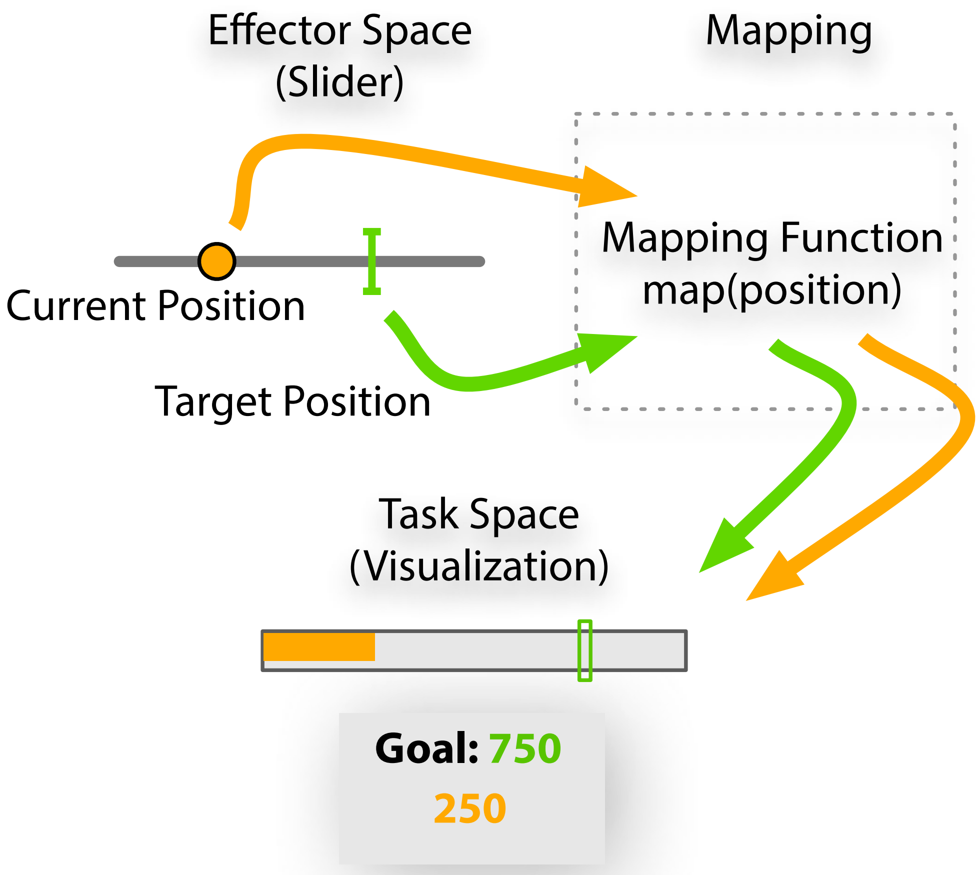
Discrete tasks that resembled accurate navigation tasks

I will now describe the tasks that participants were asked to complete, in terms of effector space, task space and mapping, as introduced in Chapter 1.1.1—"Differentiation of Task Space and Effector Space". During the trials, participants operated a hardware based sliding controller. Five distinct positions on the slider were chosen to determine targets in effector space. Values, representing the current spatial position of the sliders knob and the position of the currently valid target, were indicated on a computer display. These two substantial values in *effector space*, the *current position* of the slider and the *target position*, were subjected to a *mapping function* before they were visualized in the *task space*. (This was taken care of by a software framework, written for this user study.) A schematic diagram in Figure 3.1 shows how both values from effector space are read, converted by the same mapping function, and the converted values were subsequently visualized in task space. This approach allowed to create task conditions with the same target position in effector space, but differing in the mapping function utilized, and the type of task space used for the visualization. In each trial during a session of the experiment, one of the five possible targets was requested and visualized on the screen. The feedback for the participants consisted only of

Five target positions in effector space

Feedback mainly on a computer screen

Positions were read, mapped, and displayed (schematic diagram)



s

Figure 3.1: Schematic diagram for effector space, target space and mapping. The values, for both the current position of the sliders knob (orange) and the currently requested target (green) in effector space, are first read, then subjected to the mapping function, the then visualized in a variant of the task space.

the visual representations of the target and current position, after being subjected to the mapping function. Participants were then asked to move the slider from the furthermost left (*null position*) to the target position, as fast and accurately as possible.

Each of the five targets in effector space was requested several times, with different conditions for mapping function and task space visualization. In two conditions with the same target position in effector space, but with different mapping functions, the target would have appeared to be at different positions in task space, and identical displacements of the slider caused different effects in one task compared to the other. This approach draws the parallel between the accurate navigation tasks discussed in Chapter 1: Users have to complete a goal in task space, while the interaction is subject to a mapping function - linear or nonlinear.

3.1.2 Outline on Conditions

I will now present the conditions tested in this user study. Summarizing, every condition was characterized by three variables:

- ***target position***
- ***mapping function***
- ***task space***

Five distinct **target positions**, in effector space, were chosen for the user study, at *10, 25, 50, 75* and *90* millimeter distance from the furthermost left position on the sliding controller. Spatial positions on the slider were converted using 3 different types of parameter **mapping functions**, which could be adjusted by different parameters to control the intensity or their effect. These were:

- ***id***-mapping (representation in task space without gain)
- ***linear gain*** (amplifying or reducing distances on the slider linearly)
- ***nonlinear gain*** (the effect of gain de- or increases depending on the position on the slider).

For the visualization of the **task space**, three variants were tested:

- ***physical*** visualization (by directly indicating the target position on the sliding controller)
- ***progress bar***, imitating a common progress bar on the computer display
- ***numerical*** representing the positions on the slider by numbers on the display

The sense and purpose of these variables will be explained in detail in section 3.2—"Method and Apparatus". I want to highlight that, due to the functional principle of the framework implemented for this study (Figure 3.1), the virtual types of task space visualization (presented on the computer display) are interchangeable. In other words, when the same mapping function is utilized, the target position and the current position on the slider are indicated in different ways, but proportionally identical.

3.2 Method and Apparatus

3.2.1 Effector Space: The Hardware-based Sliding Controller

The accurate navigation tasks introduced in section 1.2—"Media Players: Timeline Sliders for Accurate Navigation" were centered on the utilization of horizontal sliding controllers. To complete the tasks in this study, participants should be provided with a similar input option. But there were several external requirements in regards to the applicability of the input option that I considered necessary. Of course, it had to resemble a timeline-slider in terms of user experience and should be operated with ease, and be simple. But it should also offer a high sampling rate and a high spatial resolution and should be easy to integrate into the hardware and software framework for this study. Last but not least, it had to be economically priced.

Standard slider widgets in Apple Mac OS X did not offer satisfactory spatial and temporal resolutions, and the intervals of generated system events were irregular. On touch-screen devices (Apple iPad2 and iPad3, Samsung Galaxy Note), the lag between gestures and visible effects on the display was distracting and interfered with task performance.

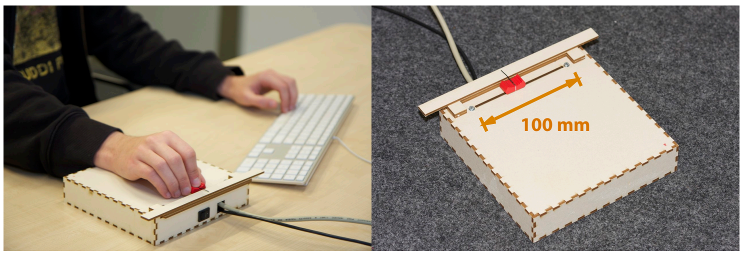


Figure 3.2: The input device: A sliding controller with a 100 millimeter sliding distance was custom built for the user study. It featured a high sampling rate (200 Hz) and a high spatial resolution (0.2 millimeters).

After an extensive test series with different input options, I decided to design and build a custom input device. The input device for the tasks was a hardware-based sliding controller, as shown in Figure 3.2. The sliding distance of the device was 100 millimeters. It was custom built for this user study, and offered sampling rate (200 Hz) and a spatial resolution (0.2 millimeters. It was built based on a commercially available microcontroller (mbed NXP LPC1768) and a sliding potentiometer (ALPS R60N).

An additional, important argument for using a hardware-based sliding controller was that the syntactic distance between this input device and the progress bar task space (introduced in 3.2.2—"The Task Space Visualizations") is very small. As touch-screen devices were out of the race, manipulating a slider widget with a mouse was the next feasible option. This would have introduced a more complex, composite mapping, comprised of the mapping between mouse and pointer, and between slider widget and content.

s

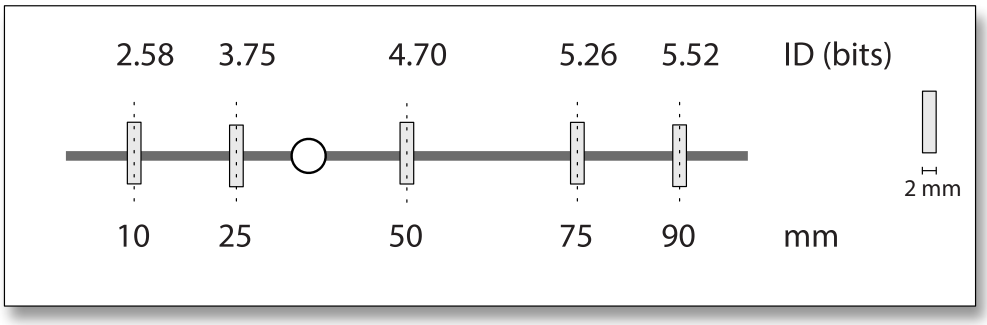


Figure 3.3: The target positions in effector space.

Targets in Effector Space

The targets distances on the slider were set at 10, 25, 50, 75 and 90 millimeters. A tolerance region, 2.0 millimeters wide, was defined around each target position. This target width was considered after an initial pilot study. When the target areas were narrower than two millimeters, the mechanical properties of the the hardware-based sliding controller, in particular friction, stiction and play made it inappropriately hard to move the sliders knob into the requested position.

This target locations covered a wide range of limb movements and were tested in pilot studies during the development phase to work well with the mapping functions utilized in the study. A task was completed successfully when the sliding controllers knob was inside the target area at the time the participant ended the task with the space bar. (For more details on the procedure, see Chapter 3.3—"Procedure".) Of course, conducting the study with only five different target positions (in effector space) for many iterations, caused concerns about bias by learning effects. This issue was tested in a pilot study with 100 random conditions, but only two different targets in effector space. Learning effects could not be observed – and participants were unaware that only two different target positions were requested.

With the distances and target widths set, the Index of Difficulty (ID) in terms of Fitts' Law can be calculated. The target positions on the slider, and the corresponding resulting levels of ID are shown in Figure 3.3 and Table 3.1.

Target Distance (D) (mm)	ID (bits)
10	2.58
25	3.75
50	4.70
75	5.26
90	5.52

Table 3.1: Distances of targets on the slider, in effector space. All targets are 2.0 millimeters wide. The values for ID have been calculated by ID = $\log_2{(1+\frac{D}{W})}$.

Technical Details of the Sliding Controller

Specifications:

- outside measurements 180 x 180 x $40mm$,
- 100 millimeters effective and physical sliding distance
- sampling rate: 200 Hz
- spatial resolution: 0.2 millimeters

The input device was assembled from a standard sliding potentiometer for fader control (ALPS RS60N: 10K Ω , linear) with a sliding distance of 100 millimeters. Several slide potentiometer models were tested, this model was selected to its superior characteristics respective to stiction, friction and precision. The micro-controller and wiring was installed in a box of dimensions 180 x 180 x $40mm$, with the sliding controller embedded in the upper side of the box.

The micro-controller (mbed NXP LPC1768) was programmed to read out the voltage across the potentiometer. The read-out voltage value was then filtered to eliminate jitter inherent to the electrical properties of the potentiometer. Physically, jitter was reduced by integrating a small

(100nF) capacitor between the potentiometer and the analogue/digital converter input pin. To further eliminate any electrical jitter, each voltage value was generated by calculating the mean average of 100 samples taken in 0.1 milliseconds. The filtered voltage was then sent to the machine running the application for the study, via ethernet.

Latency and Delay:
From the time the sliders position was sampled to an actual update of the screens content, 35 to 40 milliseconds elapsed. These specifications were determined and measured during the development phase. This was accomplished by precisely measuring the difference between the moment in time the sliders position was sampled, and the change in brightness of the screens relevant pixels. The reasons for the lag of approx. 40 milliseconds are mainly the screens inherent refresh rate and the transmission of data via Ethernet.

The 35 to 40 millisecond delay might have to be considered when the recorded data is analyzed and evaluated, although participants did not report any noticeable or even distracting lag while operating the input device.

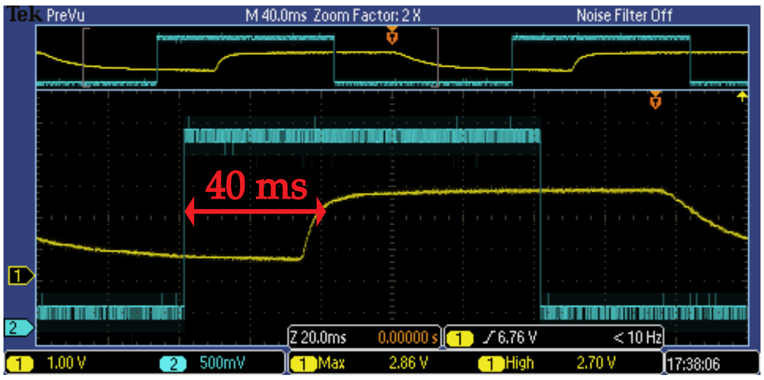


Figure 3.4: A screenshot of the measuring process to determine the delay between sampling the slider and the update of the screen. The rising of the blue plot marks a sampling event, the yellow plot indicates the change in the screens brightness. Between the sampling event and the screens update, approx. 40 milliseconds elapsed.

3.2.2 The Task Space Visualizations

Next, I want to describe the different types of visualizations that were used to display the task space. The task space had to express the current position of the slider and the target position – both after being subjected to the mapping function. I tested three types of different task spaces.

- **Physical**: A scale with printed target areas was attached directly to the hardware slider. A needle-pointer was attached to the sliders knob to indicate the current position on the scale.
- **Progress Bar**: The current slider position was represented by the filled ratio of a progress bar. The target area was indicated as a rectangular overlay on the progress bar.
- **Numerical**: Both the target area and the current slider position were represented by numbers on the screen.

Physical Task Space Visualization

A paper strip, featuring markers for the five possible target areas, was attached to the hardware slider. The five targets were indicated as gray rectangles and labelled with letters. Of course, the target areas on the paper strip were exactly as wide as the accepted target areas in other visualizations of the task space. When a task was running, the label of the valid target was shown on the computer screen. The needle on the sliders knob was hovering (frictionless) over the paper strip, naturally indicating the current position. With this setup, the task space was semantically as close to the effector space as possible, creating a natural and direct mapping.

Because of the absence of on-screen visualization, this visualization of the task space did not induce any latency or delay, in contrast to the $30 - 40$ milliseconds when using the computer screen for task space visualization. This might

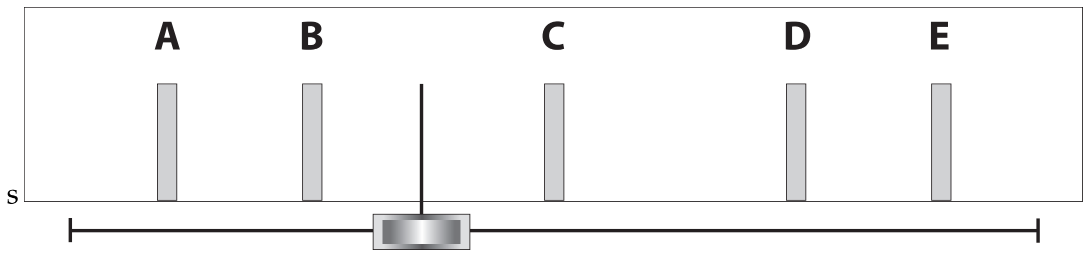


Figure 3.5: Physical task space visualization: Gray rectangles, two millimeter wide, cover the target areas and are labeled A to E. A needle shaped pointer is attached to the sliders knob, indicating the current position

have to be considered for the evaluation. And of course, due to the physical mapping this setup provided, neither gain nor a nonlinear function could be applied.

Progress Bar Task Space Visualization

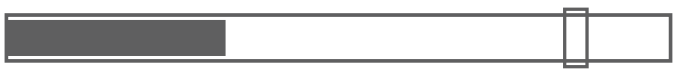


Figure 3.6: Task visualization with a progress bar. The outline shows the full range of possible values, the small rectangular outline represents the target area. The solid bar, starting from the left, visualizes the slider position. Note that the length of the outline and the dimensions of the target are are dependent on the mapping specified by the condition

This task space visualization was based on common progress bars. The progress bar was displayed as a horizontal, rectangular outline, and the progress indicator was displayed as a opaque bar inside this outline. The indicator bar started at the left side of the outline, and represented the pointer value by its length proportional to the outlines length. A smaller rectangular outline was superimposed on a segment of the bars outline, to indicate the target area. To successfully complete tasks with progress bar visualization, the end of the indicator bar had to be inside the target rectangle. The start and end position of the target outline was determined by applying the mapping function to the lower and upper bound of the two millimeter wide target area. This approach assured that, when the sliders knob entered the target area in effector space, the indicator bar would enter the target rectangle in task space, and leave it when the sliders knob was moved beyond the target area.

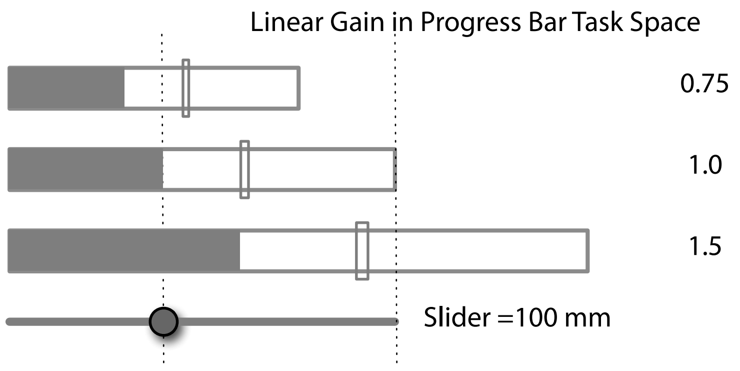


Figure 3.7: The effect of mapping functions with linear gain progress on the progress bar task space visualization. The progress bars and the target area rectangles are scaled to match the sliders length for a gain factor of 1.0, equal to the *id*-mapping.

The length of the progress bar was equal to the sliding distance of the input device, when the condition specified mapping with the *id*-mapping function, and was scaled linearly when mapping with *linear gain* was specified. If *nonlinear gain* was specified, the length of the progress bar was always set to match the sliders length, but the position of the targets, and, of course, the advance of the progress indicator, was subjected to the mapping function.

This task space visualization was chosen, because visual appearance, proportions and behavior resembled the characteristics of the timeline slider very closely. The direction of movement of the indicating part of the progress bar matched the gestures applied to the slider, and with linear gain, the movement was always proportional to the gesture. With nonlinear gain, all properties except for the spatial relationship between positions on the slider and in the progress bar stayed the same. Visual feedback from the progress bar

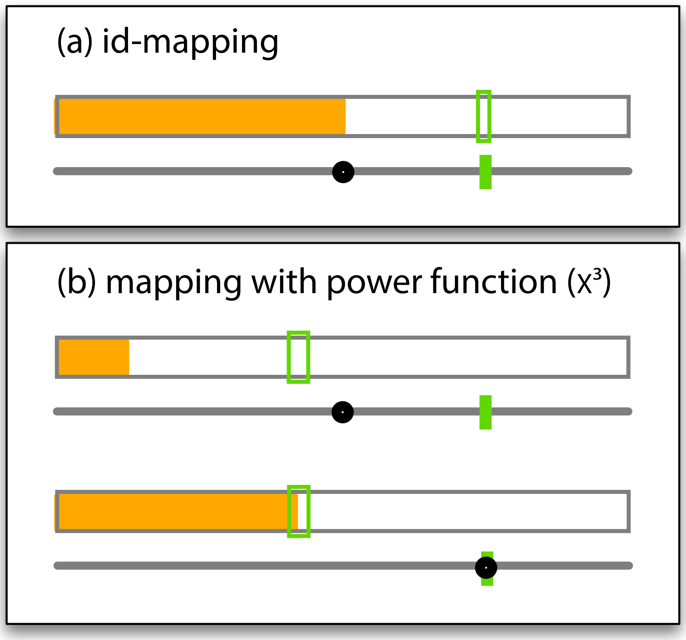


Figure 3.8: Two conditions with progress bar task space visualization and different mappings in place. In **(a)**, the mapping is linear. The position of the sliders knob and the location of the target area in effector space are mapped proportionally into the progress bar visualization. In **(b)** the task space is shown for two different slider positions. All spatial positions are subjected to mapping with the power-function. Note that the position and size of the target rectangle changed.

was simply expressed by progress and movement in one dimension, matching the input device. With this task space, participants were expected to notice the presence and effect of nonlinear gain with the least effort and most quickly, because otherwise, the progress bar would mimic the users gestures very similar.

In this task space, participants were supposed to be able to plan and execute their gestures based on their visual, spatial perception. The spatial relationship of the elements in task space were clues for the remaining distance to to the target.

Numerical Task Space Visualization

With this task space visualization, the current position of the slider and the target position were represented by two integer numerical values, after being subjected to the mapping function. In every *numerical* condition, the range of both numbers was from 1 to 1000. Figure 3.9 shows three scenarios and illustrates the effect of different mappings on this task space. To successfully complete tasks with numerical visualization, the the lower number, controlled by the slider, had to match the upper number, which represented the target position. It was a requirement for this task space visualization that only integer numbers with 1 to 3 digit were used, and the target was represented not by a range, but by single value. This way I wanted to make sure to minimize needless cognitive load for the participants.

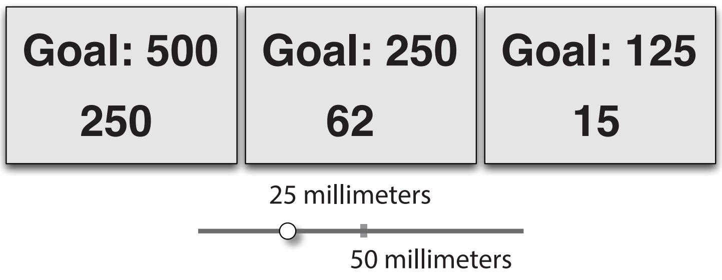


Figure 3.9: Numerical task space visualization with different mappings in place. This type of task space displays the slider position and the target position as integer numbers. Three scenarios are shown, all for the same slider position and target condition (25mm and 50mm), but three different mappings. From left to right: (id)-mapping, power functions x^2 and x^3.

Naive approaches, simply applying the mapping function to every read-out slider position and the target position and displaying the results and subsequently displaying the numerical values, had several disadvantages.

- The resulting refresh rate of the displayed number, even for moderate slider movements, was too fast to convey accurate information. Participants often stopped mid-gesture to get a reliable glimpse at the integer value.

- The target area had to be expressed as a single integer number, so that participants only had to check two numbers. Simply displaying the same number for the corresponding 2mm wide section on the slider made entering the target area in effector space much easier to detect in task space. The sudden drop of the refresh rate, when the slider entered the target area, was easy to pick up by the participants.

I want to show a brief outline on how the task space visualization was realized: The effector space was partitioned into 50 segments, 2 millimeters wide each. The partitions were laid out so that the bounds of the currently valid target area were aligned with the bounds of a single segment. Now, each segment was represented by the position of its center. This partition ensured that the target area was covered by exactly one segment, and every slider position in effector space could be assigned to a discrete segment of the effector space. Finally, to visualize the task space, the target number was determined by applying the mapping function to the target segments center-position, and the sliders position was visualized by applying the mapping function to the center-position of the segment it was currently in. With this approach, it was ensured that the refresh rate of the number for the slider position was constant, when the slider was moved with a steady speed. In addition, the target area could be represented by exactly one number. A sketch of the solution can be seen in Figure 3.10.

Contrary to the progress bar task space, this visualization offered basically no way for users to capture a spatial relationship between the representation of the slider position and the target. Still, the displayed values were an ordered set, so by comparing the slider- with the target-position, participants were able to estimate the remaining distance to the target.

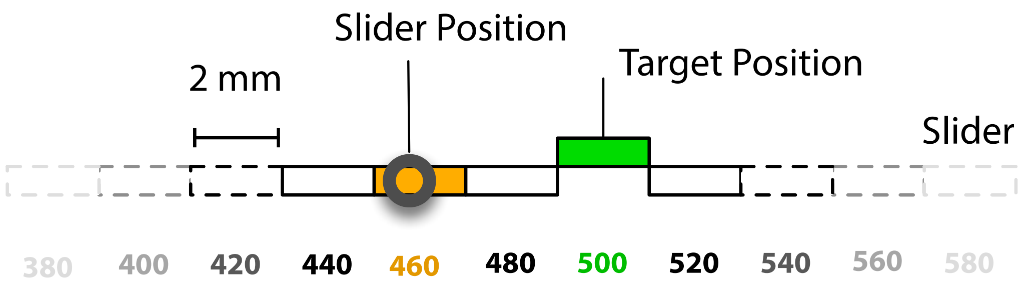


Figure 3.10: Partition of the effector space for the numerical task space. The slider was divided into 2 millimeter wide segments, to discretize the intervals for the numerical values displayed on screen. Also, this ensured that the target area could be represented by one single value.

3.2.3 The Mapping Functions

After introducing the types of task space visualizations, I want to give an overview on the mapping functions tested in the user study. I will begin with a brief outline on the general usage of the mapping functions, and follow with a detailed description of the mapping functions (with parameters) used in the user study. The mapping functions were implemented to recreate task space feedback that recreates situations in accurate navigation tasks with variable gain. They were utilized to determine mapped values for the slider position, and the bounds of the target area, which were then visualized in task space. All three values were required, because the task space visualization had to aware of the start and end of the target to display. Figure 3.11 shows a schematic diagram of how the mapping functions were used. Nonlinear mapping functions were realized by reading the spatial positions on the slider as a value in the range of 0.0 to 1.0 (furthermost left and right), and then applying the mathematically defined mapping function to this value. (Additional efforts were necessary when the condition specified a numerical task space, see 3.2.2—"Numerical Task Space Visualization".)

I tested two types of mapping functions in this user study:

Linear gain, including **(id)**-mapping (no gain) and **nonlin-**

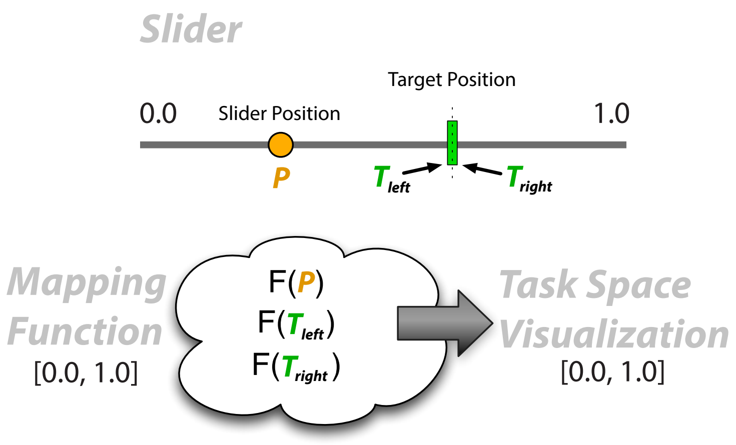


Figure 3.11: Application of the mapping functions. The slider position, and the bounds of the target area were mapped and then passed to to be displayed in the task space.

ear gain. The effects of different mapping functions in task space have already been illustrated in the corresponding chapters.

(id)-mapping Function and Linear Gain

With linear gain, relative movements of the slider had the same effect in task space, regardless of the absolute position.

(id)-mapping, essentially **linear gain with a gain factor of** 1.0, was tested with all three types of task space. In the physical task space, the mapping was naturally identical. With the progress bar task space, (id)-mapping was conducted with a progress bar with dimensions equal to the slider. In conditions with numerical task space, the numerical values were evenly distributed between the furthermost left and right position of the slider.

Linear gain factors $\neq 1.0$**:**
Conditions with progress bar task space were tested with linear gain factors $\neq 1.0$. The gain factor was realized by

'compressing' or 'stretching' the length of the bar, this attenuated or amplified the response to displacements of the slider by the gain factor. (For details, see Chapter 3.2.2—"Progress Bar Task Space Visualization" The following linear gain factors were tested by scaling the size of the progress bar: **0.5**, **1.0 (≙ (id)-mapping)**, **2.0** and **4.0**

Mapping with Nonlinear Gain

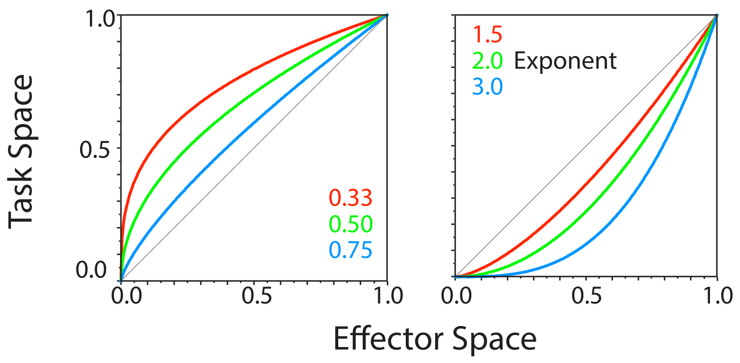


Figure 3.12: Power-functions were used to realize mapping between effector space (x-axis) and task space (y-axis). The graph lines show the mapping for different exponents compared to the *id*-mapping.

Nonlinear gain was realized with **power-functions**. These functions induced steadily increasing or decreasing gain, depending on the exponent it was parameterized with. With increasing gain, the effect of slider-displacements increased with its absolute distance from the *zero* (furthermost left) position, and vice versa for decreasing gain. Spatial positions on the slider were read as a value between 0.0 and 1.0, and then exponentiated by a parameter to acquire the values displayed in task space. The following exponents were used as parameters in this study:

- **for increasing gain:**

  0.33, 0.50, 0.75

- **for decreasing gain**:

  1.50, 2.00, 3.00

Figure 3.12 shows the mapping between effector space and task space with power functions.

Task conditions with power functions recreated scenarios as shown in Figure 3.13. Steady movements of the slider, with constant speed, cause the visualized effect to increase (or decrease, not depicted in the Figure) depending on the absolute position of the slider. (Also, compare 1.7 in Chapter 1.2.2—"Mapping between Timeline Slider and Content".)

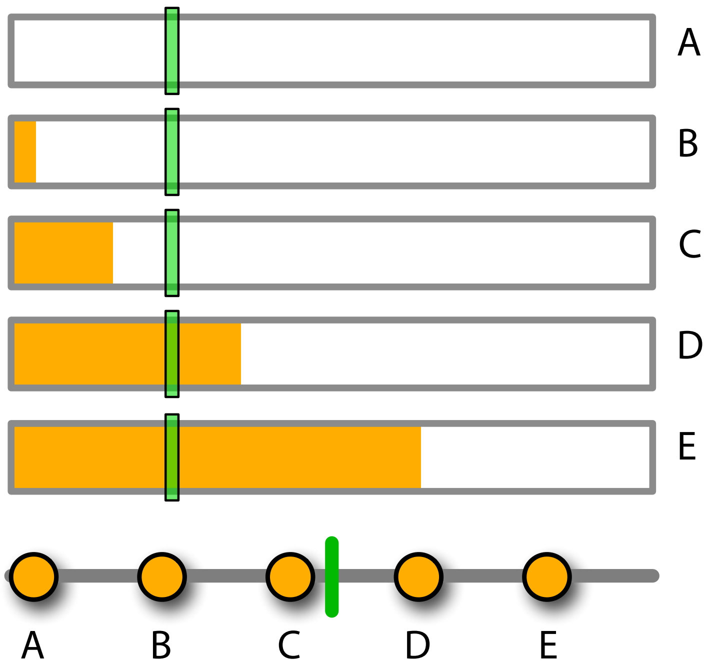


Figure 3.13: The effect of nonlinear, increasing gain in progress bar task space conditions. As the slider is displaced steadily with constant speed, the displacement caused in task space increases.

3.2.4 Summary of Conditions

Before describing the procedure for the user study, I want conclude the outline of the conditions. Table 3.2 shows the combinations of mapping function, task space type and targets that were presented to the participants. The order of the conditions was randomized for each session. Some target positions were not applicable with certain mapping functions: For highest and lowest exponents, some target positions proved to be impracticable, because the target area created in task space became too small for visual perception.

The independent variables, forming a task condition in my user study were:

The visual presentation of the task space: The task space was created:

- **physically**, directly on the sliding controller
- by a **progress bar** on screen
- by a **numerical** representation on screen

The classes of mapping between effector space and task space were:

- the *id*-function
- linear, constant gain (parameterized by gain)
- **power-functions** (parameterized by an exponent)

  exponents: 0.33; 0.5; 0.75; 1.5; 2.0; 3.0.

  Please see Figure 3.12 for details.

And, of course, the **target positions** in effector space,

- **at 10, 25, 50, 75 and 90 millimeters distance**

	Physical	Progress Bar	Numerical
Linear Gain			
0.5	*n/a*	ID1–ID5 (2x)	*n/a*
1.0	ID1–ID5 (4x)	ID1–ID5 (2x)	*n/a*
2.0	*n/a*	ID1–ID5 (2x)	*n/a*
4.0	*n/a*	ID1–ID5 (2x)	*n/a*
Nonlinear Gain			
0.33	*n/a*	ID3–ID5	ID3–ID5
0.55	*n/a*	ID2–ID5	ID2–ID5
0.75	*n/a*	ID1–ID5	ID1–ID5
1.50	*n/a*	ID1–ID5	ID1–ID5
2.00	*n/a*	ID2–ID5	ID2–ID5
3.00	*n/a*	ID3–ID5	ID2–ID5

Table 3.2: TODO

- with the corresponding **Indices of Difficulty**

  $2.59, 3, 75, 4, 70, 5, 27$ and $5, 52$

Every target position was requested four times in physical task space, to calculate a mean average value. With linear-gain–mapping, every target was requested twice, also for a mean average value. This was supposed to result in more stable data, to compare with the results from the conditions with nonlinear gain.

3.3 Procedure

Now that the tasks and the setup for the user study is defined, I will explain the procedure. First, for the study in general, followed by a detailed description on how the actual tasks were executed.

Study-Procedure:
After the participants signed an informed consent form and filled out a questionnaire (A—"Appendix - Material for the User Study"), I asked the participants to take a seat at the

table where the experimental setup was prepared. Participants were asked to adjust the chair at the table themselves, for a comfortable seating position that allowed maintaining an attention span during the study. The sliding controller was placed at the side of the participants handedness, and also adjusted by the participants, to enable a comfortable position during the session. After the sliding controllers position was settled, it was firmly attached to the desks surface. Next, I gave a short demo how to operate the sliding controller properly, and gave the participants a chance to try the setup with a series of sample conditions, which covered every type of task space and mapping. After I made sure the participants had understood the instructions and executed the tasks properly, the experimental conditions were presented. The first block of conditions was comprised of the tasks with physical task space, followed by the second block, with both the progress bar and the numerical task space visualized on the screen. The scale with target markers printed on it, was removed from the slider after the first block, leaving no clues of the distance to the target on the slider. The sequence of conditions in both blocks was randomized for every participant, to balance learning effects.

Instructions:
Bias: It became clear during the pilot studies that the instructions given to the participants had an strong influence on the behavior of the participants. (Focus on speed, focus on accuracy, etc.) To balance possible bias created by instructing the participants, a fixed set of instructions was read to the participants, and the same phrasing was used every time participants asked for further explanations. The full set of instructions read to the participants can be found in A—"Appendix - Material for the User Study".

Breaks:
At least every five minutes, a short break was mandatory, to minimize fatigue and boredom during the monotonous sequence of tasks. Participants were encouraged to take breaks in between tasks if they experienced any kind of discomfort or fatigue. Including breaks, the study took about 20 to 30 minutes per participant. Including paperwork, introduction

and briefing, participants donated about 40 to 50 minutes of their time.

Task-Procedure:
Each task started with the slider in the furthermost left position. After ending the previous task by pressing the space bar, participants *nulled* (move to zero) the slider, and had to rest there for at least a second. This resting phase was introduced because participants were inclined to rapidly yank the slider right after reaching the left side - without visually perceiving the next target. This behavior produced unnecessary overshoots, thus, the resting was was mandatory. In conditions with physical task space, participants were asked to take a look at the requested target position on the screen, and then complete the task with visual focus on the sliding controller. In conditions with virtual task space representations, participants had to focus entirely on the screen while operating the slider. When the slider set to the final position, participants dismissed the task by pressing the space bar, thus invoking the next task. The sliders position was only rendered in task space when the slider was moved out of the idle position, so no clues about the mapping in the upcoming task was given away. The clock started at the moment the slider was moved out of the nulled position, and was stopped right after the last correctional movement. If a participant ended a task by pressing the space bar without having the slider in the valid target area, the next task was invoked regardless. The incomplete task was then inserted into the remaining task sequence at random, to be presented again.

3.3.1 Demographic

28 unpaid volunteers, 6 female and 22 male, participated in the user study. The majority of the volunteers were students of computer science the RWTH Aachen University. Participants were aged 17 to 41, and had normal or corrected vision. Four participants were left handed, 26 were right handed, and all participants were encouraged to use their dominant hand for the relevant interaction during the user

study. No participant showed any signs of impairments in vision or motor system that would have impacted the general performance during the study.

During the course of this diploma thesis, sixteen additional volunteers participated in the user study. I decided not to include the data from these experiments for different reasons: Some of these participants were too intimately involved in the development process of the user study, others participated in early iterations of the user study, were minor bugs in the setup might have influenced their performance. Finally, some participants repeatedly disregarded the instructions they were given, and generated an high number of outliers that made it impossible to cleanly interpret the rest of their results.

Chapter 4

Results and Evaluation

The presentation of data gathered during the user study will be structured by the mapping functions applied. I will start by showing an overview on each data set and a first analysis based on descriptive statistics. The results were checked for statistical significance with standardized repeated measurements ANOVA tests, Tukey's test, and Student's t-test methods built-in to SAS JMP (v10)[1] and IBM SPSS Statistics (v19)[2] .

4.1 Results for Mapping with (id)-function

An overview of the data for conditions with (id)-mapping is shown in Table 4.1 and Figure 4.1.

[1]http://www.jmp.com
[2]http://www.ibm.com/software/de/analytics/spss/products/statistics

Task Space and Target		Mean Avg. MT (ms)	Std. Deviation of MT (ms)
Physical	ID 1	621	230
	ID 2	905	313
	ID 3	1084	290
	ID 4	1163	331
	ID 5	1273	313
			(ø295)
Progress Bar	ID 1	920	436
	ID 2	1032	315
	ID 3	1290	292
	ID 4	1513	550
	ID 5	1639	530
			(ø424)
Numerical	ID 1	2346	1106
	ID 2	1971	1847
	ID 3	1964	1176
	ID 4	2544	1063
	ID 5	2994	1290
			(ø1296)

Table 4.1: (id)-mapping: Mean movement times (MT) and standard deviations.

4.1.1 (id)-Mapping: The Standard Deviation is Significantly Higher for Numerical Task Space Conditions

The standard deviation for conditions with physical task space was lowest (mean 295 ms), followed by progress bar conditions with a mean standard deviation of 424 ms. Conditions with numerical task space representation stands out, the standard deviation of performance was more then three times higher (1290 ms), with more and farther outliers. This was confirmed with a oneway ANOVA test ($F[1, 8] = 37.690$, $p = 0.0003^*$), after merging the results from physical and progress bar task space conditions, respectively.

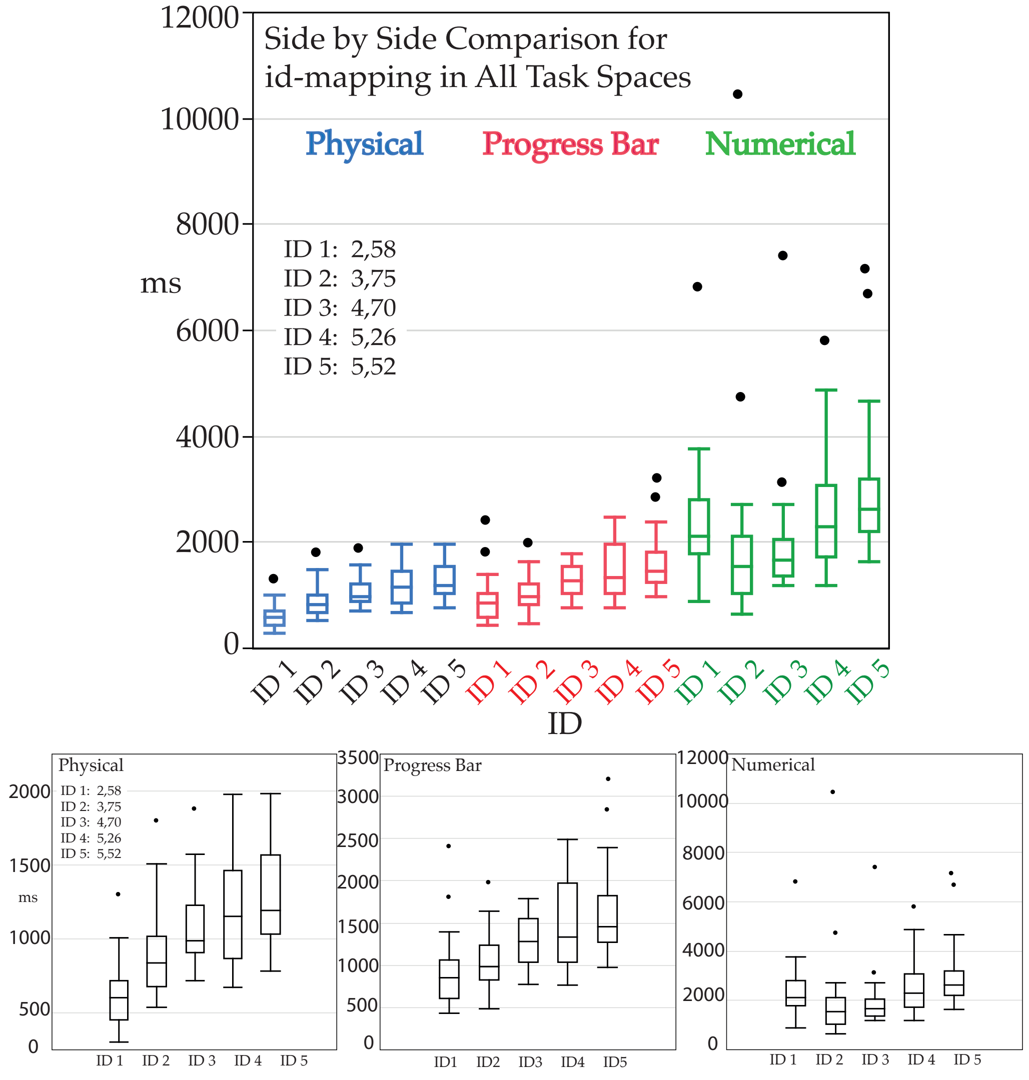


Figure 4.1: (id)-mapping: Boxplots for movement time (MT) measurements, comparing the data sets for task space visualization types. The Index of Difficulty (ID) is calculated using the expression $ID = \log_2{(1 + \frac{D}{W})}$, with the distance of the target position and the width of two millimeters in effector space.

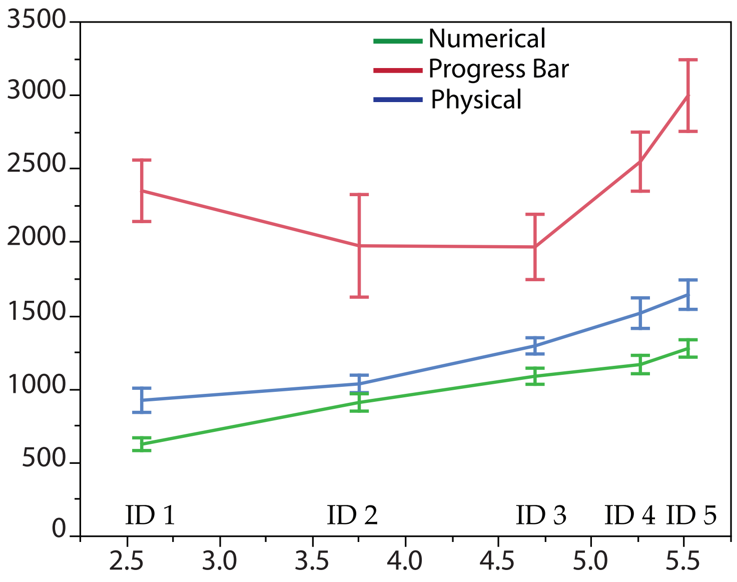


Figure 4.2: Comparing mean average MT for (id)-mapping, with bars for standard error. The movement time increased significantly for higher values of ID. Also, MT was significantly depending on the task space the task was performed in.

4.1.2 Movement Time Was Significantly Lower in Physical- and Progress Bar Task Space Conditions

Figure 4.2, comparing the mean average performance by task space, indicates significantly different levels of performance for different task spaces, regardless of the IDs level. The statistical reports (*ANOVA for effects within subjects* and *Tukey's HSD Connected Letters*) generated for all target positions confirm that participants performed significantly better in physical and progress bar task space conditions, compared to the numerical task space. (See Figure 4.3 for detailed results.) This effect is consistent for all levels of ID. A subsequent test, checking the physical vs. progress bar conditions, verified that the movement time was significantly different in for all target positions, except for the second. Figure 4.4 shows the reports from ANOVA (to test the significance) and the all-pairs–Tukey-Kramer graph (to visually compare the means).

4.1.3 (id)-mapping: Effect of Difficulty on Movement Time

After these general observations, I examined the movement time results for different levels of ID. Figure 4.2 indicates that movement time increased with higher levels of ID, in physical- and progress bar task space. A repeated measurement ANOVA test (for each task space condition), confirmed significantly different movement times for different levels of ID.

There was a **significant** effect of ID on MT in **physical task space**

- ANOVA: $F[4, 108] = 31.553, p < 0.0001^*$

  (Sphericity assumed, $ChiSquare = 14.182$, $p = 0.116$)

There was a **significant** effect of ID on MT in **progress bar task space**

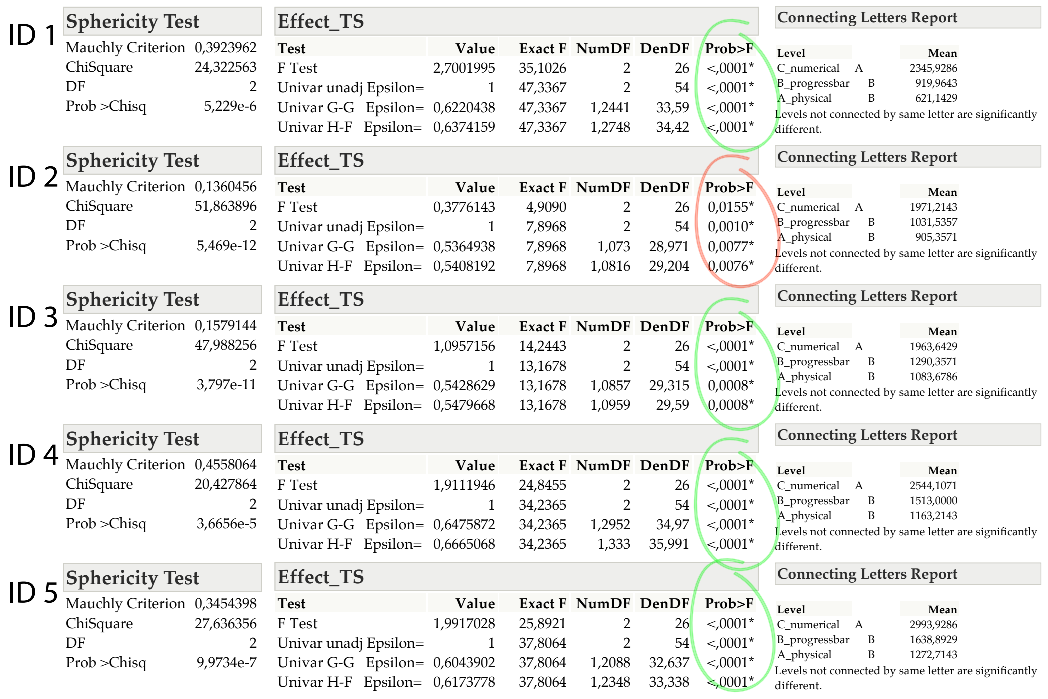
- G.-G.: $F[2.925, 78.977] = 21.773, p < 0.0001^*$

  (Sphericity violated, $ChiSquare = 14.182$, $p = 0.006^*$. Therefore the Greenhouse-Geisser test was used.)

There was **significant** effect of ID on MT in **numerical task space**

- G.-G.: $F[2.452, 66.213] = 7.316, p = 0.0006^*$

  (Spericity violated, $ChiSquare = 29.151$, $p = 0.0006^*$. Therefore the Greenhouse-Geisser test was used.)



s ID 1

Sphericity Test	
Mauchly Criterion	0,3923962
ChiSquare	24,322563
DF	2
Prob >Chisq	5,229e-6

Effect_TS

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	2,7001995	35,1026	2	26	<,0001*
Univar unadj Epsilon=	1	47,3367	2	54	<,0001*
Univar G-G Epsilon=	0,6220438	47,3367	1,2441	33,59	<,0001*
Univar H-F Epsilon=	0,6374159	47,3367	1,2748	34,42	<,0001*

Connecting Letters Report

Level			Mean
C_numerical	A		2345,9286
B_progressbar		B	919,9643
A_physical		B	621,1429

Levels not connected by same letter are significantly different.

ID 2

Sphericity Test	
Mauchly Criterion	0,1360456
ChiSquare	51,863896
DF	2
Prob >Chisq	5,469e-12

Effect_TS

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	0,3776143	4,9090	2	26	0,0155*
Univar unadj Epsilon=	1	7,8968	2	54	0,0010*
Univar G-G Epsilon=	0,5364938	7,8968	1,073	28,971	0,0077*
Univar H-F Epsilon=	0,5408192	7,8968	1,0816	29,204	0,0076*

Connecting Letters Report

Level			Mean
C_numerical	A		1971,2143
B_progressbar		B	1031,5357
A_physical		B	905,3571

Levels not connected by same letter are significantly different.

ID 3

Sphericity Test	
Mauchly Criterion	0,1579144
ChiSquare	47,988256
DF	2
Prob >Chisq	3,797e-11

Effect_TS

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	1,0957156	14,2443	2	26	<,0001*
Univar unadj Epsilon=	1	13,1678	2	54	<,0001*
Univar G-G Epsilon=	0,5428629	13,1678	1,0857	29,315	0,0008*
Univar H-F Epsilon=	0,5479668	13,1678	1,0959	29,59	0,0008*

Connecting Letters Report

Level			Mean
C_numerical	A		1963,6429
B_progressbar		B	1290,3571
A_physical		B	1083,6786

Levels not connected by same letter are significantly different.

ID 4

Sphericity Test	
Mauchly Criterion	0,4558064
ChiSquare	20,427864
DF	2
Prob >Chisq	3,6656e-5

Effect_TS

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	1,9111946	24,8455	2	26	<,0001*
Univar unadj Epsilon=	1	34,2365	2	54	<,0001*
Univar G-G Epsilon=	0,6475872	34,2365	1,2952	34,97	<,0001*
Univar H-F Epsilon=	0,6665068	34,2365	1,333	35,991	<,0001*

Connecting Letters Report

Level			Mean
C_numerical	A		2544,1071
B_progressbar		B	1513,0000
A_physical		B	1163,2143

Levels not connected by same letter are significantly different.

ID 5

Sphericity Test	
Mauchly Criterion	0,3454398
ChiSquare	27,636356
DF	2
Prob >Chisq	9,9734e-7

Effect_TS

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	1,9917028	25,8921	2	26	<,0001*
Univar unadj Epsilon=	1	37,8064	2	54	<,0001*
Univar G-G Epsilon=	0,6043902	37,8064	1,2088	32,637	<,0001*
Univar H-F Epsilon=	0,6173778	37,8064	1,2348	33,338	<,0001*

Connecting Letters Report

Level			Mean
C_numerical	A		2993,9286
B_progressbar		B	1638,8929
A_physical		B	1272,7143

Levels not connected by same letter are significantly different.

Figure 4.3: (id)-Mapping: ANOVA: Significantly different movement times for different task space conditions. The corrected Greenhouse-Geisser (G.-G.) tests should be considered, as Mauchly's sphericity test is significant for all ID conditions. The G.-G. test confirms a highly significant difference for MT for different task space conditions.

Detailed reports are also shown in Figure 4.5.

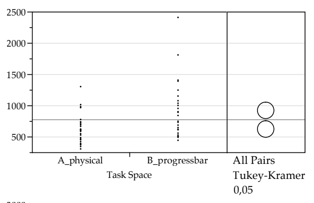
In addition, the effect of ID on performance was significantly different for different task space conditions (G.-G.: $F[5.717, 231.54] = 4.041$, $p = 0.0009^*$, sphericity violated: $ChiSquare = 55.312$, $p < 0.0001^*$) Detailed reports are also shown in Figure 4.6.

Linear Regression: Fitts' Law Model for (id)-Mapping

The next step was to derive parameters from the data, to establish a Fitts' Law model that predicts movement time from the Index of Difficulty. Models were derived for each task space condition individually, as there were significant

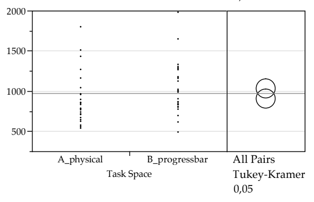
Effect_TS

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	0,4045727	10,9235	1	27	0,0027*
Univar unadj Epsilon=	1	10,9235	1	27	0,0027*
Univar G-G Epsilon=	1	10,9235	1	27	0,0027*
Univar H-F Epsilon=	1	10,9235	1	27	0,0027*



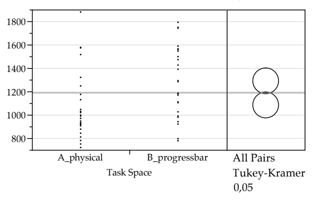
Effect_TS

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	0,096999	2,6190	1	27	0,1172
Univar unadj Epsilon=	1	2,6190	1	27	0,1172
Univar G-G Epsilon=	1	2,6190	1	27	0,1172
Univar H-F Epsilon=	1	2,6190	1	27	0,1172



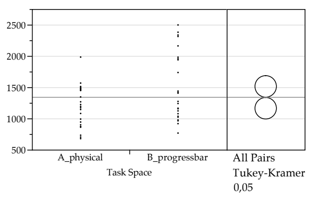
Effect_TS

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	0,5972963	16,1270	1	27	0,0004*
Univar unadj Epsilon=	1	16,1270	1	27	0,0004*
Univar G-G Epsilon=	1	16,1270	1	27	0,0004*
Univar H-F Epsilon=	1	16,1270	1	27	0,0004*



Effect_TS

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	0,5735055	15,4846	1	27	0,0005*
Univar unadj Epsilon=	1	15,4846	1	27	0,0005*
Univar G-G Epsilon=	1	15,4846	1	27	0,0005*
Univar H-F Epsilon=	1	15,4846	1	27	0,0005*



Effect_TS

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	0,5962866	16,0997	1	27	0,0004*
Univar unadj Epsilon=	1	16,0997	1	27	0,0004*
Univar G-G Epsilon=	1	16,0997	1	27	0,0004*
Univar H-F Epsilon=	1	16,0997	1	27	0,0004*

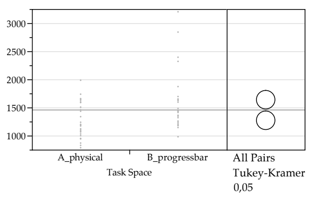
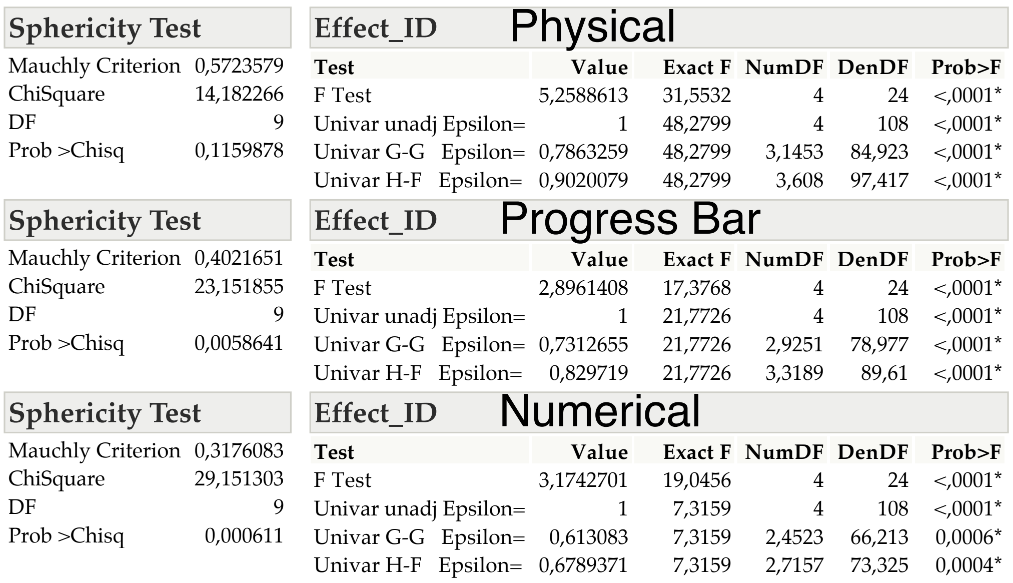


Figure 4.4: (id)-Mapping: ANOVA: The results for the physical and progress bar task space differ significantly for all levels of ID, except for the second.

interactions between ID and task space type. The parameters for a the Shannon-formulation of Fitts' Law ($MT = a + b \cdot ID$) have been derived by linear regression, and establish the following models:

Models for the **physical-**, **progress bar-** and **numerical** task space:

- $MT_{(id),phys} = 90 + 211 \cdot ID$
- $MT_{(id),bar} = 210 + 245 \cdot ID$



Sphericity Test	
Mauchly Criterion	0,5723579
ChiSquare	14,182266
DF	9
Prob >Chisq	0,1159878

Effect_ID Physical

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	5,2588613	31,5532	4	24	<,0001*
Univar unadj Epsilon=	1	48,2799	4	108	<,0001*
Univar G-G Epsilon=	0,7863259	48,2799	3,1453	84,923	<,0001*
Univar H-F Epsilon=	0,9020079	48,2799	3,608	97,417	<,0001*

Sphericity Test	
Mauchly Criterion	0,4021651
ChiSquare	23,151855
DF	9
Prob >Chisq	0,0058641

Effect_ID Progress Bar

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	2,8961408	17,3768	4	24	<,0001*
Univar unadj Epsilon=	1	21,7726	4	108	<,0001*
Univar G-G Epsilon=	0,7312655	21,7726	2,9251	78,977	<,0001*
Univar H-F Epsilon=	0,829719	21,7726	3,3189	89,61	<,0001*

S

Sphericity Test	
Mauchly Criterion	0,3176083
ChiSquare	29,151303
DF	9
Prob >Chisq	0,000611

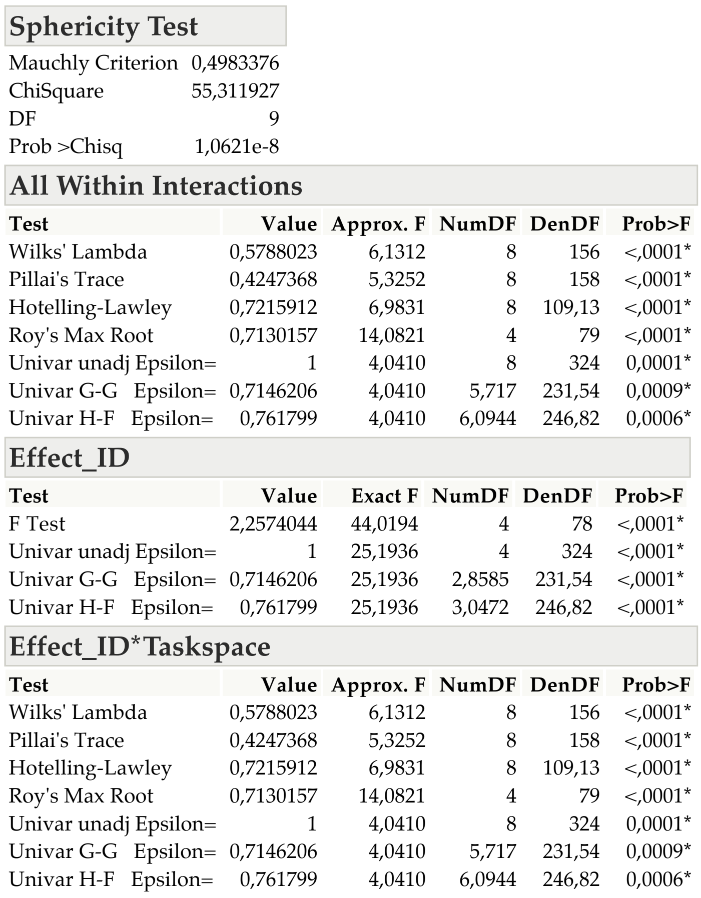
Effect_ID Numerical

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	3,1742701	19,0456	4	24	<,0001*
Univar unadj Epsilon=	1	7,3159	4	108	<,0001*
Univar G-G Epsilon=	0,613083	7,3159	2,4523	66,213	0,0006*
Univar H-F Epsilon=	0,6789371	7,3159	2,7157	73,325	0,0004*

Figure 4.5: (id)-Mapping: ANOVA: The effect of ID on MT was significant in all task space visualizations. The F-test statistics can be used for the physical task space, while the effects for the other conditions have to be verified with the Greenhouse-Geisser (G.-G.) test, as the spericity criterion is violated.

- $MT_{(id),num} = 1590 + 177 \cdot ID$ (invalid)

Plotted graphs for the models are shown in Figure 4.7. The regression analysis also confirmed that the effect of ID on MT is different for each task space condition, resulting in models of different goodness. The models for the physical- and progress bar task space are rather good (Persons' R^2=0.375 and R^2=0.269), although the value for **a** (intercept) is not significant. The goodness of the numerical task space model is too low (Pearson's' R^2=0.013) to say the model is valid. Detailed results are shown in Table 4.2.



Sphericity Test

Mauchly Criterion	0,4983376
ChiSquare	55,311927
DF	9
Prob >Chisq	1,0621e-8

All Within Interactions

Test	Value	Approx. F	NumDF	DenDF	Prob>F
Wilks' Lambda	0,5788023	6,1312	8	156	<,0001*
Pillai's Trace	0,4247368	5,3252	8	158	<,0001*
Hotelling-Lawley	0,7215912	6,9831	8	109,13	<,0001*
Roy's Max Root	0,7130157	14,0821	4	79	<,0001*
Univar unadj Epsilon=	1	4,0410	8	324	0,0001*
Univar G-G Epsilon=	0,7146206	4,0410	5,717	231,54	0,0009*
Univar H-F Epsilon=	0,761799	4,0410	6,0944	246,82	0,0006*

Effect_ID

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	2,2574044	44,0194	4	78	<,0001*
Univar unadj Epsilon=	1	25,1936	4	324	<,0001*
Univar G-G Epsilon=	0,7146206	25,1936	2,8585	231,54	<,0001*
Univar H-F Epsilon=	0,761799	25,1936	3,0472	246,82	<,0001*

Effect_ID*Taskspace

Test	Value	Approx. F	NumDF	DenDF	Prob>F
Wilks' Lambda	0,5788023	6,1312	8	156	<,0001*
Pillai's Trace	0,4247368	5,3252	8	158	<,0001*
Hotelling-Lawley	0,7215912	6,9831	8	109,13	<,0001*
Roy's Max Root	0,7130157	14,0821	4	79	<,0001*
Univar unadj Epsilon=	1	4,0410	8	324	0,0001*
Univar G-G Epsilon=	0,7146206	4,0410	5,717	231,54	0,0009*
Univar H-F Epsilon=	0,761799	4,0410	6,0944	246,82	0,0006*

Figure 4.6: (id)-Mapping: ANOVA: The effect of ID on MT, previously verified, is subjected to a significant interaction with the task space condition.

Task Space	Goodness of Model
Physical	R^2=0.375 ANOVA, whole model: $F[1,138] = 82.921, p < 0.0001^*$ Parameter Estimates: a: t-ratio=0.86, Prob.> $\lvert t\rvert$: $= 0.389$ b: t-ratio=9.11, Prob.> $\lvert t\rvert$: $< 0.0001^*$
Progress bar	R^2=0.269 ANOVA, whole model: $F[1,138] = 50.583, p < 0.0001^*$ Parameter Estimates: a: t-ratio=1.36, Prob.> $\lvert t\rvert$: $= 0.177$ b: t-ratio=7.11, Prob.> $\lvert t\rvert$: $< 0.0001^*$
Numerical	R^2=0.013 ANOVA, whole model: $F[1,138] = 2.781, p = 0.098$ Parameter Estimates: a: t-ratio=3.33, Prob.> $\lvert t\rvert$: $= 0.001$ b: t-ratio=1.67, Prob.> $\lvert t\rvert$: $= 0.098^*$

Table 4.2: (id)-Mapping: The derived predictive models are of different quality.

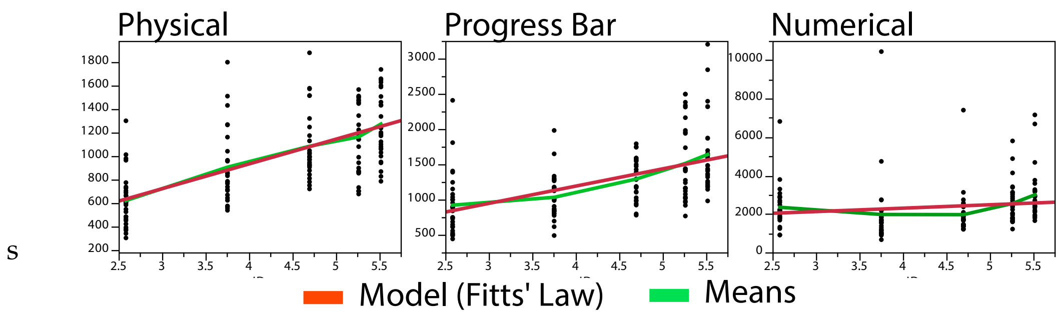


Figure 4.7: Parameters for a model according to *Fitts' Law* can be derived from the experimental data for (id)-mapping conditions by *linear regression*. However, the models for the **physical** and **progress bar** task space conditions are of higher quality then the model for the **numerical** task space.

4.2 Results for Mapping with Linear Gain

I will now present the results for conditions with **progress bar** task space visualization and different linear gain factors. Again, I want to start with a presentation of the datas distribution, followed by standardized tests for significant effects. Three orders of gain besides 1.0 were tested: 0.5, 2.0 and 4.0. A gain factor of 1.0 would match the (id)-mapping on progress bar visualization previously presented, but will be shown again in the graphs and tables to simplify the comparison. An overview of the data for conditions with (id)-mapping is shown in Table 4.1 and Figure 4.8.

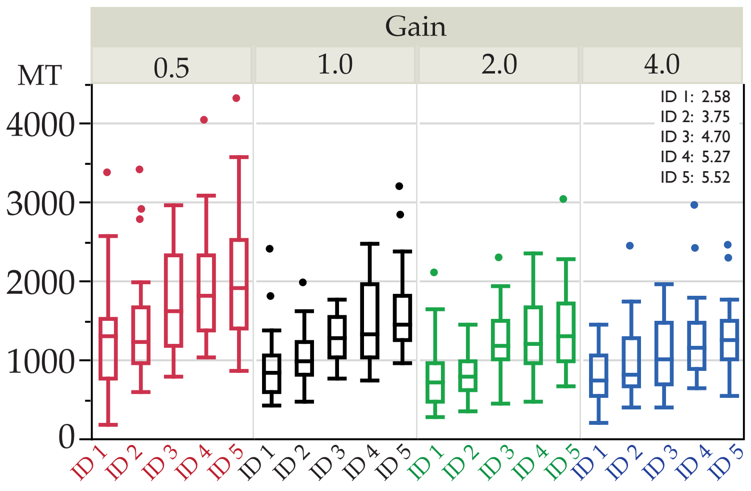


Figure 4.8: Linear Gain: Boxplots for movement time (MT) measurements, comparing the data sets for task space visualization types. The Index of Difficulty (ID) is calculated using the expression $ID = \log_2{(1 + \frac{D}{W})}$, with the distance of the target position and the width of two millimeters in effector space.

4.2.1 Linear Gain: The Standard Deviation is Significantly Higher for Gain Factor 0.5

Figure 4.8 and Table 4.3 indicate that the standard deviation is higher for for conditions with gain factor 0.5. This was

	Linear Gain Factor left: **øMT**, right: *Std. Deviation*							
ID (bits)	0.5		1.0		2.0		4.0	
ID 1	**1279**	*717*	**920**	*436*	**793**	*424*	**822**	*344*
ID 2	**1426**	*701*	**1032**	*315*	**841**	*295*	**996**	*469*
ID 3	**1834**	*678*	**1290**	*292*	**1247**	*410*	**1091**	*454*
ID 4	**1899**	*702*	**1513**	*550*	**1326**	*450*	**1250**	*516*
ID 5	**2009**	*824*	**1639**	*530*	**1439**	*552*	**1302**	*449*
		(ø724)		(ø424)		(ø426)		(ø446)

Table 4.3: Mean average moving times (MT) and standard deviation of MT, for measurements of conditions with **progress bar** task space and mapping with different linear gain factors.

verified with a oneway ANOVA test ($F[1, 8] = 47.403$, $p < 0.0001$*), after merging the values for standard deviation for the other gain factors, respectively.

4.2.2 Movement Time Was Significantly Higher in Conditions with Gain Factor 0.5

A comparison of the mean average MT, as shown in Figure 4.9, indicates that the performance in conditions a gain factor of 0.5 is considerably worse than for other linear gain factors. The same figure does not indicate considerable difference between he the other gain factors, therefore I tested the significance of a difference between gain factor 0.5 and all other linear gain conditions combined. Repeated measurements ANOVA tests, for all target positions individually, confirmed that MT was significantly higher in conditions with linear gain 0.5. Detailed reports are shown in Table 4.4.

4.2.3 Linear Gain: Effect of Difficulty on Movement Time

Figure 4.9 indicates that, regardless of gain factor, movement time increased with higher levels of ID. This was also confirmed by a repeated measurements ANOVA test (for

ID	F Statistic
ID 1	$F[1,27] = 4.896, p = 0.036^*$
ID 2	$F[1,27] = 13.184, p = 0.001^*$
ID 3	$F[1,27] = 19.587, p < 0.0001^*$
ID 4	$F[1,27] = 8.846, p = 0.006^*$
ID 5	$F[1,27] = 10.743, p = 0.003^*$

Table 4.4: Linear Gain: Reports for repeated measurements ANOVA, comparing the MT for gain 0.5 with the other conditions.

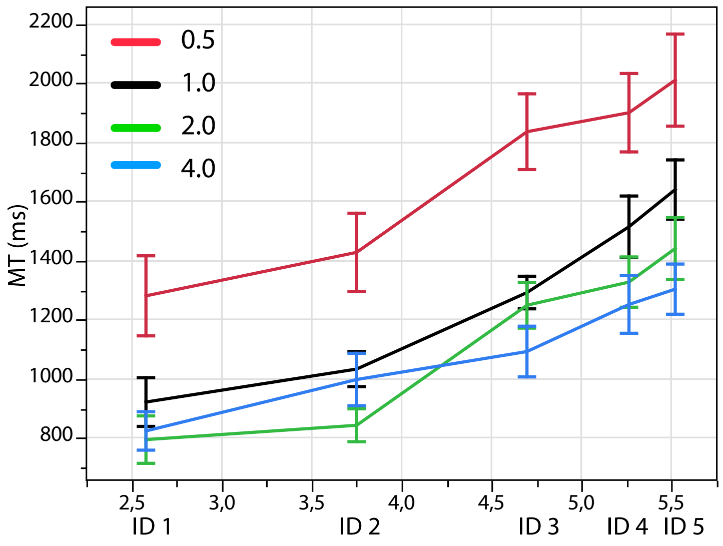


Figure 4.9: Linear Gain: Comparing mean average MT for different ID, with bars for standard error.

each gain factor).

There was a **significant** effect of ID on MT for **gain factor 0.5**:

- ANOVA: $F[4,24] = 6.717, p = 0.001^*$

  (Sphericity assumed, $ChiSquare = 5.045$, $p = 0.830$)

There was a **significant** effect of ID on MT for **gain factor**

1.0:

- G.-G.: $F[2.925, 78.977] = 21.772, p < 0.0001^*$

  (Spericity violated, $ChiSquare = 23.152$, $p = 0.006^*$. Therefore the Greenhouse-Geisser test was used.)

There was a **significant** effect of ID on MT for **gain factor 2.0**:

- ANOVA: $F[4, 24] = 10.740, p < 0.0001^*$

  (Sphericity assumed, $ChiSquare = 11.450$, $p = 0.246$)

There was a **significant** effect of ID on MT for **gain factor 4.0**:

- ANOVA: $F[4, 24] = 6.930, p = 0.001^*$

  (Sphericity assumed, $ChiSquare = 10.899$, $p = 0.283$)

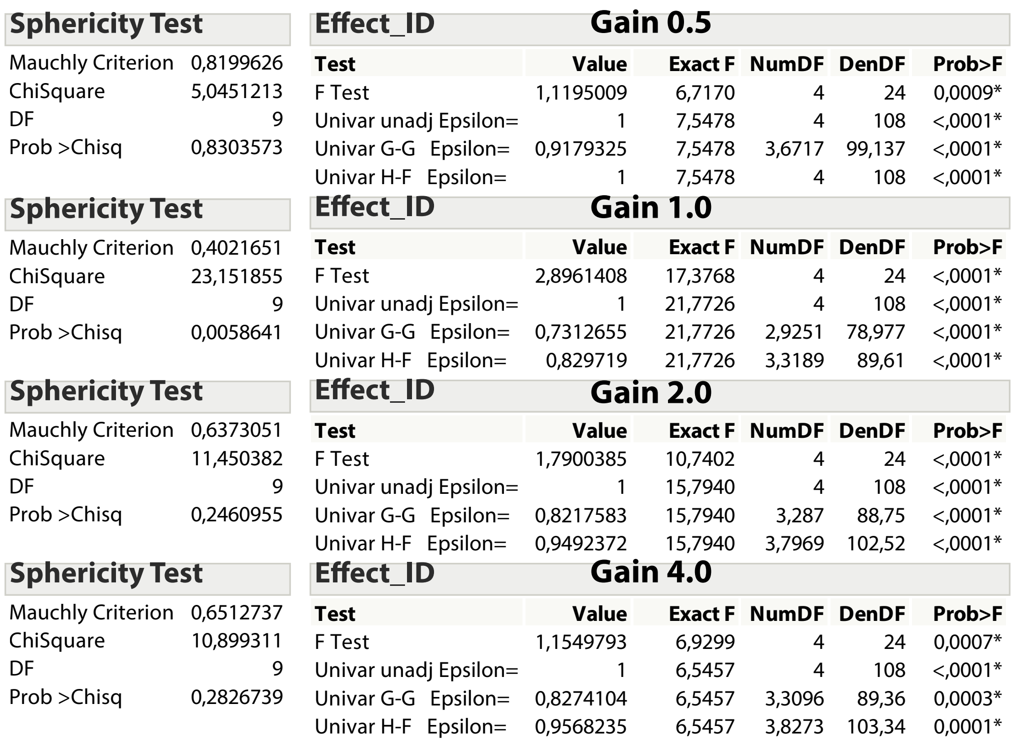
Detailed results are also shown in Figure 4.10.

Linear Regression: Fitts' Law Model for Mapping with Linear Gain.

I derived models for conditions with linear gain, just like for (id)-mapping (Shown in 4.1.3—"Linear Regression: Fitts' Law Model for (id)-Mapping"). Plotted graphs for the models are shown in Figure 4.11.

Model for linear gain factor **0.5**, **1.0** (which equals (id)-mapping, shown previously), **2.0** and **4.0**:

- $MT_{(linear),0.5} = 563 + 258 \cdot ID$



Sphericity Test (Gain 0.5)

Mauchly Criterion	0,8199626
ChiSquare	5,0451213
DF	9
Prob >Chisq	0,8303573

Effect_ID **Gain 0.5**

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	1,1195009	6,7170	4	24	0,0009*
Univar unadj Epsilon=	1	7,5478	4	108	<,0001*
Univar G-G Epsilon=	0,9179325	7,5478	3,6717	99,137	<,0001*
Univar H-F Epsilon=	1	7,5478	4	108	<,0001*

Sphericity Test (Gain 1.0)

Mauchly Criterion	0,4021651
ChiSquare	23,151855
DF	9
Prob >Chisq	0,0058641

Effect_ID **Gain 1.0**

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	2,8961408	17,3768	4	24	<,0001*
Univar unadj Epsilon=	1	21,7726	4	108	<,0001*
Univar G-G Epsilon=	0,7312655	21,7726	2,9251	78,977	<,0001*
Univar H-F Epsilon=	0,829719	21,7726	3,3189	89,61	<,0001*

Sphericity Test (Gain 2.0)

Mauchly Criterion	0,6373051
ChiSquare	11,450382
DF	9
Prob >Chisq	0,2460955

Effect_ID **Gain 2.0**

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	1,7900385	10,7402	4	24	<,0001*
Univar unadj Epsilon=	1	15,7940	4	108	<,0001*
Univar G-G Epsilon=	0,8217583	15,7940	3,287	88,75	<,0001*
Univar H-F Epsilon=	0,9492372	15,7940	3,7969	102,52	<,0001*

Sphericity Test (Gain 4.0)

Mauchly Criterion	0,6512737
ChiSquare	10,899311
DF	9
Prob >Chisq	0,2826739

Effect_ID **Gain 4.0**

Test	Value	Exact F	NumDF	DenDF	Prob>F
F Test	1,1549793	6,9299	4	24	0,0007*
Univar unadj Epsilon=	1	6,5457	4	108	<,0001*
Univar G-G Epsilon=	0,8274104	6,5457	3,3096	89,36	0,0003*
Univar H-F Epsilon=	0,9568235	6,5457	3,8273	103,34	0,0001*

Figure 4.10: Linear Gain: ANOVA: The effect of ID on MT was significant for linear gain factors.

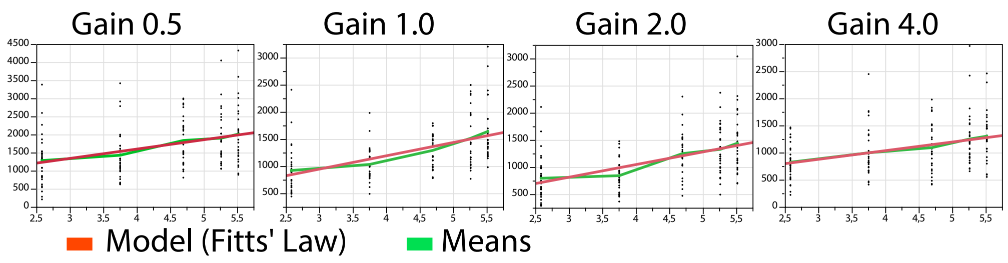


Figure 4.11: Parameters for a model based on Fitts' Law were derived by linear regression of the MT.

- $MT_{(linear),1.0} = 210 + 245 \cdot ID$
- $MT_{(linear),2.0} = 114 + 233 \cdot ID$
- $MT_{(linear),4.0} = 398 + 160 \cdot ID$

Detailed results for the goodness of the models are listed in

Table 4.5.

Gain Factor	Goodness of Model
0.5	R^2=0.131 ANOVA, whole model: $F[1, 138] = 20.840, p < 0.0001^*$ Parameter Estimates: a: t-ratio=2.22, Prob.$>$ $\|t\|$: $= 0.028$ b: t-ratio=4.57, Prob.$>$ $\|t\|$: $< 0.0001^*$
1.0	R^2=0.268 ANOVA, whole model: $F[1, 138] = 50.583, p < 0.0001^*$ Parameter Estimates: a: t-ratio=1.36, Prob.$>$ $\|t\|$: $= 0.177$ b: t-ratio=7.11, Prob.$>$ $\|t\|$: $< 0.0001^*$
2.0	R^2=0.250 ANOVA, whole model: $F[1, 138] = 46.129, p < 0.0001^*$ Parameter Estimates: a: t-ratio=0.74, Prob.$>$ $\|t\|$: $= 0.46$ b: t-ratio=6.79, Prob.$>$ $\|t\|$: $< 0.0001^*$
4.0	R^2=0.130 ANOVA, whole model: $F[1, 138] = 20.694, p < 0.0001^*$ Parameter Estimates: a: t-ratio=2.53, Prob.$>$ $\|t\|$: $= 0.013$ b: t-ratio=4.55, Prob.$>$ $\|t\|$: $< 0.0001^*$

Table 4.5: Linear Gain: Goodness of models derived from the data.

4.3 Results for Mapping with (id)-function

Figure 4.1 gives an overview on the data for conditions nonlinear mapping with power-functions.

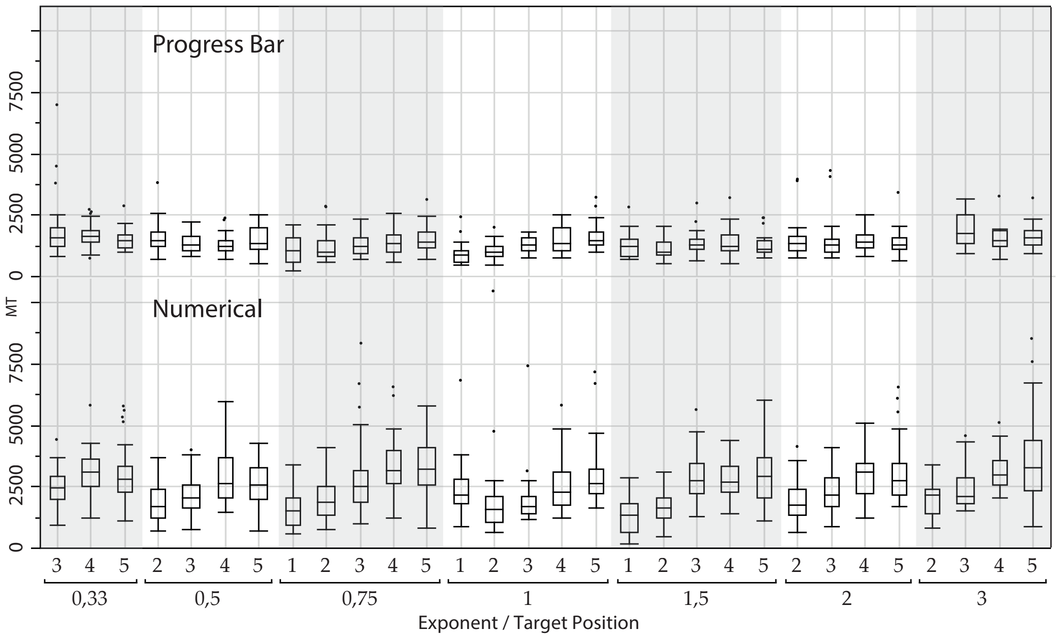


Figure 4.12: Nonlinear Gain, power functions: Boxplots for movement time (MT) measurements, comparing the data sets for task space visualization types.

4.3.1 Nonlinear Gain: The Standard Deviation is Significantly Higher for Numerical Task Space Conditions

Figure 4.1 suggests that the standard deviation is higher when nonlinear gain was applied with a numerical task space, compared to the progress bar task space. This could be confirmed with a oneway ANOVA test ($F[1, 57] = 56.836$, $p < 0.0001$*). I combined the data sets from different target positions and gain factors for this test, as Figure 4.1 does not indicate a significant interaction.

4.3.2 Nonlinear Gain: Movement Time is Significantly Higher for Numerical Task Space Conditions

Again, the next effect I examined was if the movement time increased with higher levels of ID, when nonlinear gain is applied. The effect is indicated by Figure 4.13, which

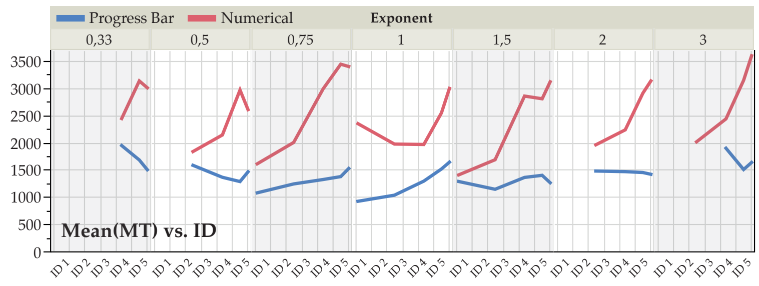


Figure 4.13: Nonlinear Gain, power functions: Comparing mean average MT for mapping with nonlinear gain, by power-functions. The movement time increased significantly for higher values of ID. Also, MT was significantly depending on the task space the task was performed in.

compares the mean average MT for different levels of ID, grouped by task space. A repeated measurements ANOVA was conducted for each level of ID separately, because Figure 4.13 also indicates that the effect interacts with the level of ID. The ANOVA reports, shown in Table 4.6, confirm that the movement times are significantly higher for conditions with numerical task space.

ID	F Statistic
ID 1	$F[1, 55] = 6.190, p = 0.0159^*$
ID 2	$F[1, 111] = 34.001, p < 0.0001^*$
ID 3	$F[1, 167] = 80.044, p < 0.0001^*$
ID 4	$F[1, 167] = 350.983, p < 0.0001^*$
ID 5	$F[1, 167] = 320.163, p < 0.0001^*$

Table 4.6: Nonlinear Gain, power functions: Reports for repeated measurements ANOVA, comparing the MT for conditions progress bar task space and numerical task space.

4.3.3 Nonlinear Gain: Effect of Difficulty on Movement Time

Figure 4.13 suggests that, for conditions with numerical task space, there is a significant increase in MT for higher levels of ID. This effect can not be observed for the progress bar task space conditions. I conducted a repeated measurements ANOVA test to confirm the effect in numerical task space. This test verified that movement times increased significantly for increased levels of ID. ($F[4, 52] = 29.001$, $p < 0.0001^*$, sphericity assumed).

4.4 Findings

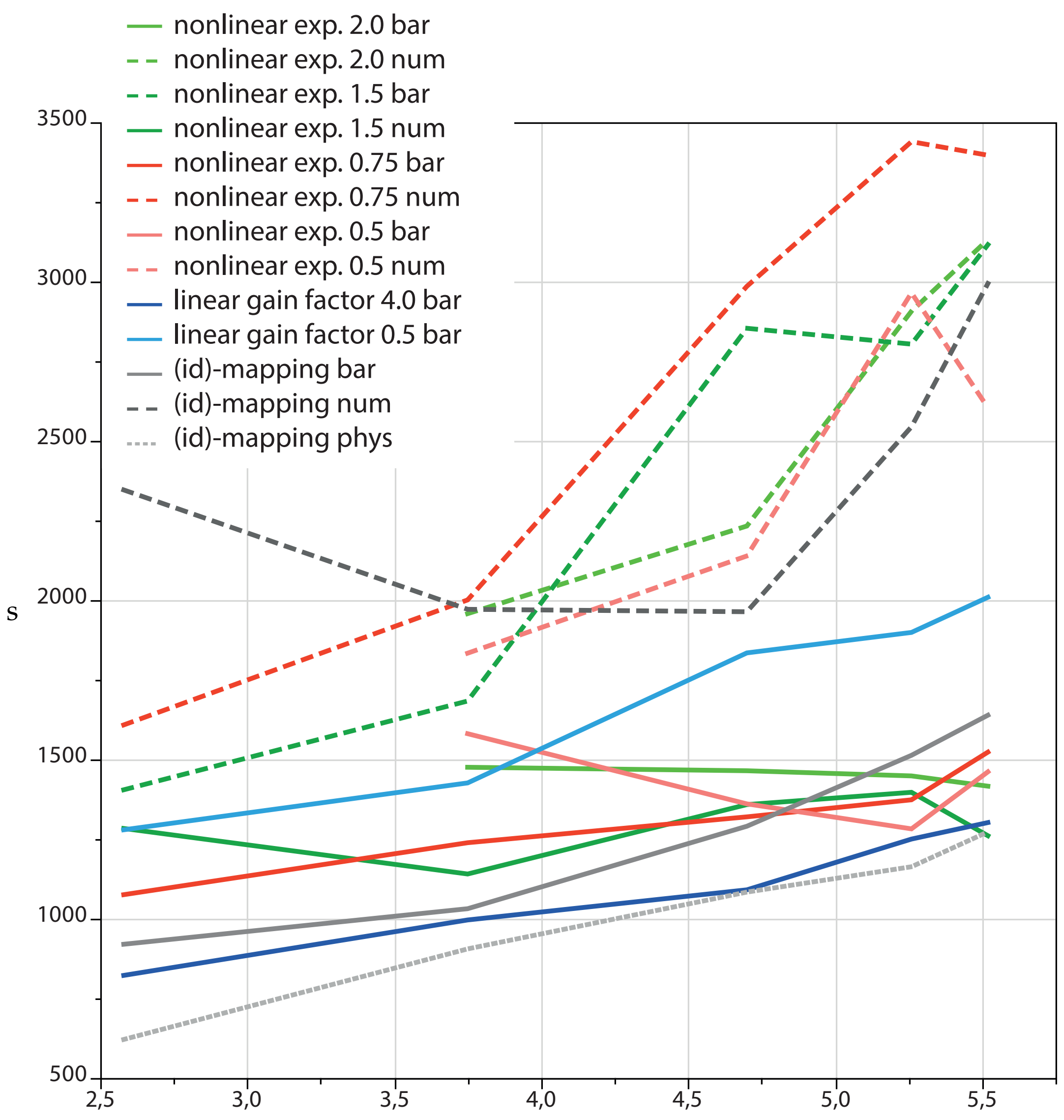


Figure 4.14: Plotted MT vs. ID, for a selection of conditions. The color indicates the mapping, the line-style indicates the task space.

Chapter 5

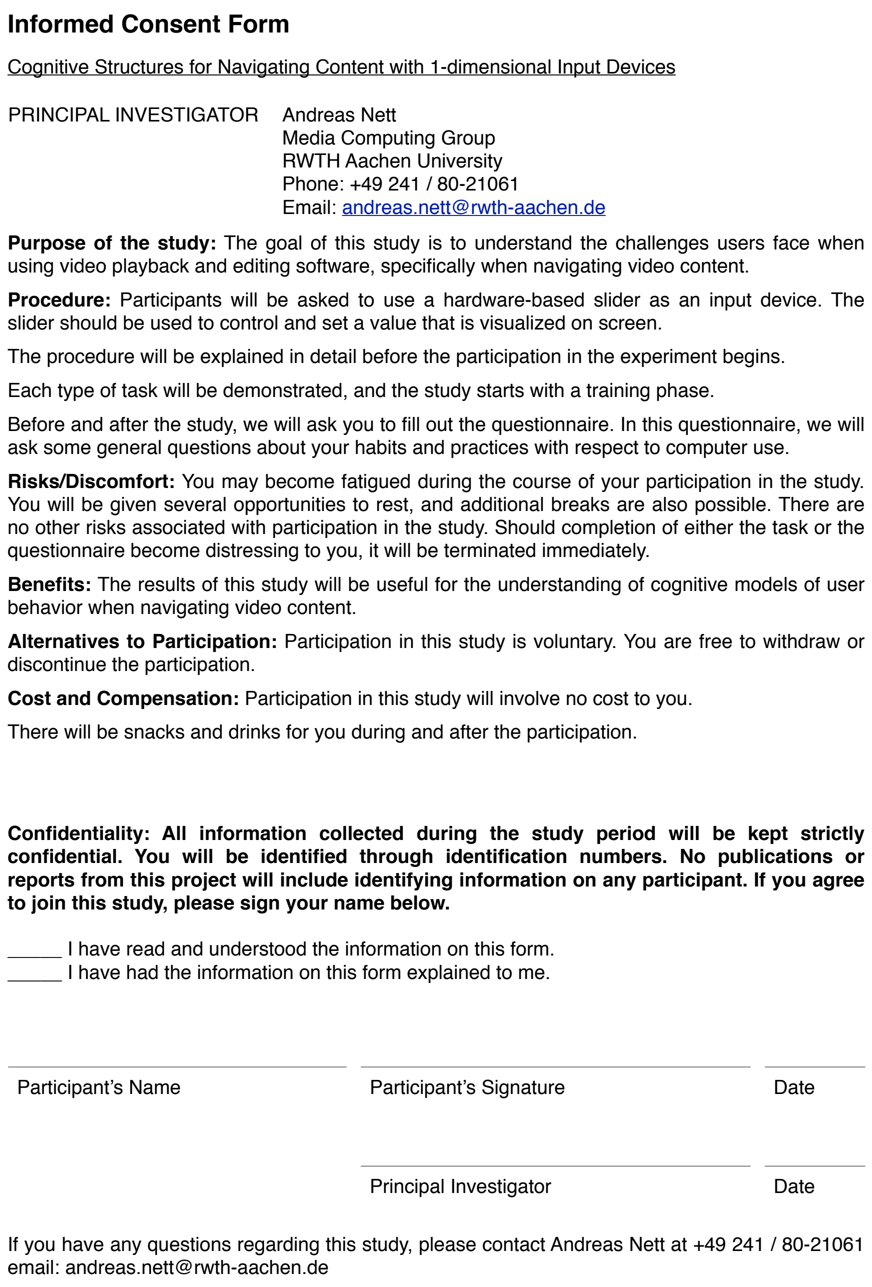
Summary and future work

5.1 Summary and contributions

5.2 Future work

Appendix A

Appendix - Material for the User Study



Informed Consent Form

Cognitive Structures for Navigating Content with 1-dimensional Input Devices

PRINCIPAL INVESTIGATOR	Andreas Nett Media Computing Group RWTH Aachen University Phone: +49 241 / 80-21061 Email: andreas.nett@rwth-aachen.de

Purpose of the study: The goal of this study is to understand the challenges users face when using video playback and editing software, specifically when navigating video content.

Procedure: Participants will be asked to use a hardware-based slider as an input device. The slider should be used to control and set a value that is visualized on screen.

The procedure will be explained in detail before the participation in the experiment begins.

Each type of task will be demonstrated, and the study starts with a training phase.

Before and after the study, we will ask you to fill out the questionnaire. In this questionnaire, we will ask some general questions about your habits and practices with respect to computer use.

Risks/Discomfort: You may become fatigued during the course of your participation in the study. You will be given several opportunities to rest, and additional breaks are also possible. There are no other risks associated with participation in the study. Should completion of either the task or the questionnaire become distressing to you, it will be terminated immediately.

Benefits: The results of this study will be useful for the understanding of cognitive models of user behavior when navigating video content.

Alternatives to Participation: Participation in this study is voluntary. You are free to withdraw or discontinue the participation.

S

Cost and Compensation: Participation in this study will involve no cost to you.

There will be snacks and drinks for you during and after the participation.

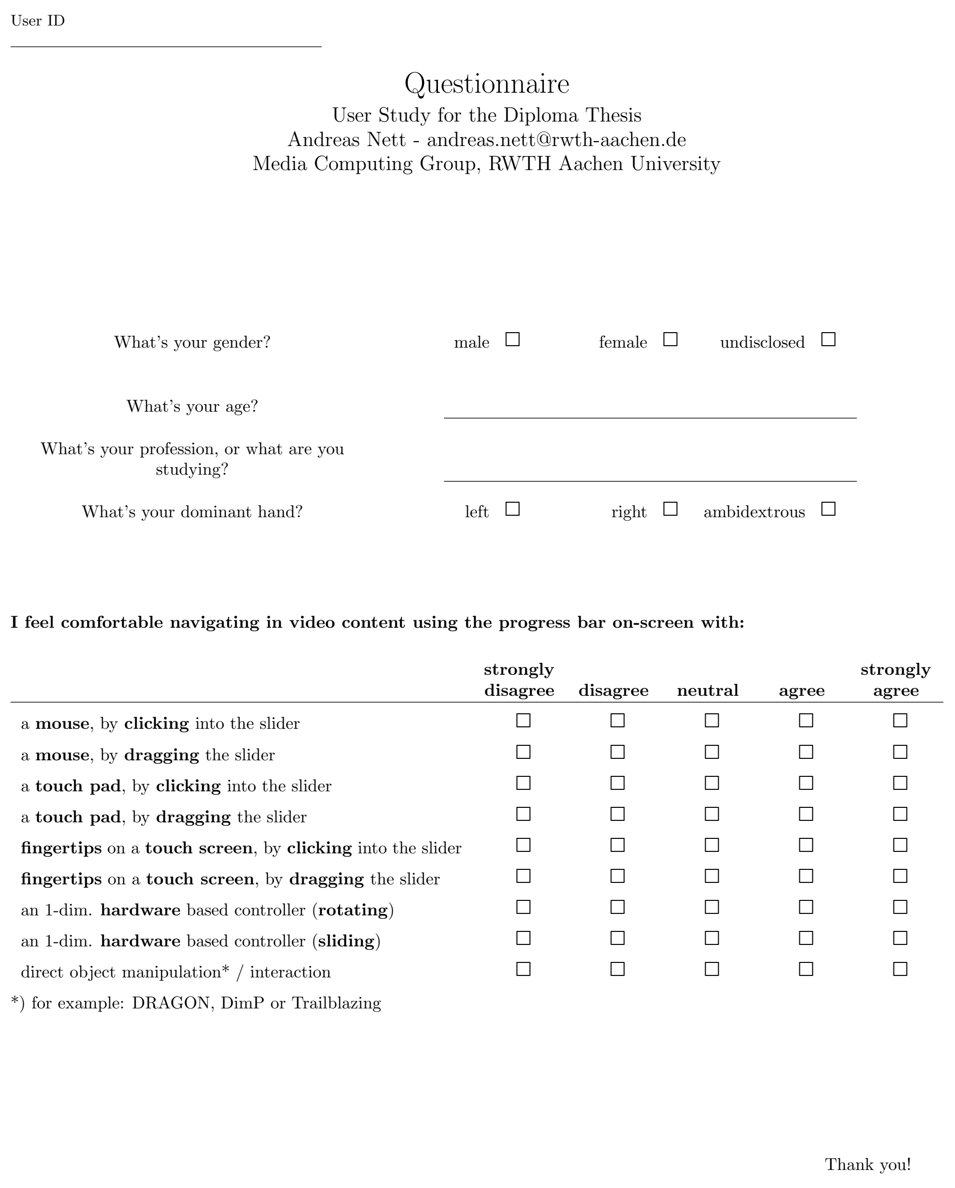
Confidentiality: All information collected during the study period will be kept strictly confidential. You will be identified through identification numbers. No publications or reports from this project will include identifying information on any participant. If you agree to join this study, please sign your name below.

_____ I have read and understood the information on this form.
_____ I have had the information on this form explained to me.

Participant's Name	Participant's Signature	Date
	Principal Investigator	Date

If you have any questions regarding this study, please contact Andreas Nett at +49 241 / 80-21061
email: andreas.nett@rwth-aachen.de

Figure A.1: The consent form for the user study.



User ID

____

Questionnaire

User Study for the Diploma Thesis
Andreas Nett - andreas.nett@rwth-aachen.de
Media Computing Group, RWTH Aachen University

What's your gender?	male ☐	female ☐	undisclosed ☐
What's your age?	______________________________		
What's your profession, or what are you studying?	______________________________		
What's your dominant hand?	left ☐	right ☐	ambidextrous ☐

I feel comfortable navigating in video content using the progress bar on-screen with:

	strongly disagree	**disagree**	**neutral**	**agree**	**strongly agree**
a **mouse**, by **clicking** into the slider	☐	☐	☐	☐	☐
a **mouse**, by **dragging** the slider	☐	☐	☐	☐	☐
a **touch pad**, by **clicking** into the slider	☐	☐	☐	☐	☐
a **touch pad**, by **dragging** the slider	☐	☐	☐	☐	☐
fingertips on a **touch screen**, by **clicking** into the slider	☐	☐	☐	☐	☐
fingertips on a **touch screen**, by **dragging** the slider	☐	☐	☐	☐	☐
an 1-dim. **hardware** based controller (**rotating**)	☐	☐	☐	☐	☐
an 1-dim. **hardware** based controller (**sliding**)	☐	☐	☐	☐	☐
direct object manipulation* / interaction	☐	☐	☐	☐	☐

*) for example: DRAGON, DimP or Trailblazing

Thank you!

Figure A.2: The questionnaire for the user study.

Bibliography

Johnny Accot and Shumin Zhai. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 295–302, New York, USA, 1997.

Johnny Accot and Shumin Zhai. Scale effects in steering law tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1–8, New York, USA, 2001.

Johnny Accot and Shumin Zhai. Refining Fitts' law models for bivariate pointing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 193–200, New York, USA, 2003.

Christopher Ahlberg and Ben Shneiderman. The alphaslider: a compact and rapid selector. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 365–371, New York, USA, 1994.

Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 519–526, New York, USA, 2004.

Reinoud J. Bootsma, Laure Fernandez, and Denis Mottet. Behind Fitts' law: kinematic patterns in goal-directed movements. *International Journal of Human-Computer Studies*, Vol. 61:811–821, December 2004.

Stuart K. Card, Thomas P. Moran, and Allen Newell. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, Vol. 23 (7):396–410, July 1980.

Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. The design space of input devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, pages 117–124, New York, USA, 1990.

Andy Cockburn and Andrew Firth. Improving the acquisition of small targets, 2003.

Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. Video browsing by direct manipulation. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 237–246, New York, USA, 2008.

Digby Elliott, Werner F. Helsen, and Romeo Chua. A century later: Woodworth's (1899) two-component model of goal-directed aiming. *Psychological Bulletin*, 127(3):342–57, May 2001.

Laure Fernandez and Reinoud J Bootsma. Non-linear gaining in precision aiming: Making Fitts' task a bit easier. *Acta Psychologica*, Vol. 129(2):217–227, October 2008.

Paul M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, Vol. 47:381–391, 1954.

Yves Guiard, Michel Beaudouin-Lafon, and Denis Mottet. Navigation as multiscale pointing: extending Fitts' model to very high precision tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, pages 450–457, New York, USA, 1999.

Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. Direct manipulation interfaces. *Human-Computer Interaction*, 1(4):311–338, December 1985.

Thorsten Karrer, Malte Weiss, Eric Lee, and Jan Borchers. DRAGON: a direct manipulation interface for frame-accurate in-scene video navigation. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 247–250, New York, USA, 2008.

David Kieras. Using the keystroke-level model to estimate execution times, 1993.

Don Kimber, Tony Dunnigan, Andreas Girgensohn, Frank Shipman, Thea Turner, and Tao Yang. Trailblazing: video playback control by direct object manipulation. ICME, 2007.

I. Scott MacKenzie. Human-computer interaction. chapter Movement time prediction in human-computer interfaces, pages 483–492. Morgan Kaufmann Publishers Inc., San Francisco, USA, 1995.

I. Scott MacKenzie. *Motor Behaviour Models for Human-Computer Interaction*, pages 27–54. Morgan Kaufmann Publishers Inc., 2003.

I. Scott Mackenzie, Abigail Seller, and William Buxton. A comparison of input devices in elemental pointing and dragging tasks. pages 161–166, 1991.

Karin Nieuwenhuizen, Dzmitry Aliakseyeu, and Jean-Bernard Martens. Insight into goal-directed movement strategies. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 883–886, New York, USA, 2010.

Donald A. Norman. *The Design of Everyday Things*. Basic Books, Inc., New York, USA, 1988.

Gonzalo Ramos and Ravin Balakrishnan. Fluid interaction techniques for the control and annotation of digital video. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, UIST '03, pages 105–114, New York, USA, 2003.

Claude Elwood Shannon and the Institute of Radio Engineers. *Communication in the Presence of Noise*. Institute of Radio Engineers, 1949.

Ben Shneiderman. *Designing the user interface: strategies for effective human-computer interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 1986.

Moritz Wittenhagen. Dragoneye - fast object tracking and camera motion estimation. Master's thesis, RWTH Aachen University, Aachen, Germany, October 2008.

Robert S. Woodworth. *The Accuracy of Voluntary Movement, by R. S. Woodworth,...* Macmillan, 1899.

Aileen Worden, Nef Walker, Krishna Bharat, and Scott Hudson. Making computers easier for older adults to use: area cursors and sticky icons. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, CHI '97, pages 266–271, New York, USA, 1997.

Index

Typeset December 24, 2012