# Smart Views in Smart Meeting Rooms

**Dissertation**
**zur**
**Erlangung des akademischen Grades**
**Doktor-Ingenieur (Dr.-Ing.)**
**der Fakultät für Informatik und Elektrotechnik**
**der Universität Rostock**

vorgelegt von

**Axel Radloff,** *geboren am 01. Januar 1983 in Rostock*
*aus Rostock*

Rostock, 2014

urn:nbn:de:gbv:28-diss2014-0132-9

| | |
|---|---|
| Gutachter | Prof. Dr. Heidrun Schumann, Universität Rostock |
| | Prof. Dr. Oliver Staadt, Universität Rostock |
| | Prof. Dr. Dieter Schmalstieg, Technische Universität Graz |
| Einreichung | 13.01.2014 |
| Verteidigung | 25.06.2014 |

**Zusammenfassung**

Die Präsentation von Informationen in Smart Meeting Rooms ist auf Grund der Heterogenität und Dynamik der Umgebung vor große Herausforderungen gestellt. Die Charakteristika der verschiedenen Geräte im Raum, die heterogenen Displayflächen und die unterschiedlichen Applikationen, sowie das Ziel smarter Umgebungen, den Nutzer bei seiner Arbeit zu unterstützen, stellen neue Anforderungen und Probleme für die Informationspräsentation dar.

Ziel dieser Dissertation ist es daher, eine Problemlösung für diese neuen Herausforderungen zu entwickeln. Neben der Generierung und Anzeige von Informationsrepräsentationen spielt dabei die Interaktion mit diesen angezeigten Informationen eine entscheidende Rolle. Für die Präsentation von Informationen wird in dieser Dissertation das Konzept des *Smart View Managements* vorgestellt. Das *Smart View Management* ermöglicht dabei eine Generierung, Verteilung und Anordnung von visuellen Ausgaben unterschiedlicher Nutzer und Applikationen, unterstützt durch automatische Ansätze. Für die Interaktion in Smart Meeting Rooms wird das Konzept des *Smart Interaction Managements* vorgestellt. Es ermöglicht allen Nutzern die Interaktion mit allen dargestellten Informationen auf allen Displayflächen. Auf Basis dieser beiden entwickelten und umgesetzten Konzepte werden Methoden vorgestellt, die die Anzeige und den Inhalt der Informationsrepräsentationen interaktiv manipulieren.

**Abstract**

Information presentation in Smart Meeting Rooms faces huge challenges due to the heterogeneity and dynamic of those environments. The characteristics of different devices, the heterogeneity of the display surfaces and the diverse applications, along with the aim of smart environments to support users in their tasks, pose new problems and challenges for the presentation of information.

The aim of this dissertation is to develop solutions for these new challenges. Along with the generation and displaying of information, the interaction with this displayed information plays a crucial role. To present information the concept of the *Smart View Management* is introduced. The *Smart View Management* enables the generation, distribution and arrangement of visual outputs of different applications and users, supported by automatic methods. To interact in Smart Meeting Rooms the concept of the *Smart Interaction Management* is presented. It enables all users to interact with all displayed Information representations on all displays. Furthermore, based on both developed and realized concepts, methods are presented to interactively manipulate the display and the content of information representations.

## Danksagung

An dieser Stelle möchte ich meinen Dank all denen gegenüber Ausdrücken, die diese Arbeit möglich gemacht haben.

Zu allererst möchte ich mich bei Frau Prof. Heidrun Schumann für die Betreuung bedanken. Durch ihre Unterstützung, Motivation und vor allem durch ihre konstruktive Kritik hat sie diese Arbeit erst möglich gemacht. Bedanken möchte ich mich weiterhin bei Prof. Oliver Staadt und Prof. Dieter Schmalstieg, für ihre Bereitschaft diese Arbeit zu begutachten.

Ebenso geht mein ausdrücklicher Dank an bei Martin Luboschik. Seine außerordentliche Betreuung schon während meiner Studien- und Diplomarbeit haben den Grundstein für mein Interesse gelegt, meine Promotion in Angriff zu nehmen. Darüber hinaus haben die konstruktiven Gespräche, Vorschläge und die hervorragende Zusammenarbeit maßgeblich zu dieser Arbeit beigetragen.

Auch möchte ich mich für die Zusammenarbeit bei Albert Pritzkau, Georg Fuchs, Mike Sips, Christian Tominski und vor allem Anke Lehmann bedanken. Ein weiterer Dank geht an alle weiteren Mitarbeitern des Lehrstuhls Computergrafik und des Lehrstuhl Visual Computing für das großartige kollegiale Umfeld. Gleiches gilt natürlich auch für alle Stipendiaten, Kollegiaten und Professoren des Graduiertenkollegs MuSAMA, dessen Finanzierung diese Arbeit ermöglichte.

Mein Dank richtet sich auch an alle, die diese Arbeit Korrektur gelesen haben und damit viele Stunden ihrer Freizeit geopfert haben.

Ein besonderer Dank gebührt meiner Ehefrau Jeanette, meinem Sohn Christopher Neo und meinen Eltern. Ohne ihren Rückhalt, ihre Unterstützung und ihr Verständnis für meine Arbeit wäre diese Dissertation nicht möglich gewesen.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1. Einleitung

## 1.1. Motivation

Smart Meeting Rooms fokussieren auf die Zusammenarbeit heterogener Geräte durch das Erzeugen eines gemeinsamen Geräteensembles. Dabei sind Smart Meeting Rooms Multi-Display-Umgebungen, die durch diverse zusätzliche Aspekte ergänzt werden. So sind in der Regel verschiedenste Sensoren vorhanden, um Daten über die Nutzer und die Umgebung zu erfassen, und Software, um die aufgenommenen Daten auszuwerten. Diese Umgebung wird zusätzlich ergänzt durch die vom Nutzer mitgebrachten persönlichen Geräte, die sich in das Gesamtensemble integrieren.

Ziel solcher Smarten Umgebungen ist es, Nutzer bei ihrer Arbeit in diesen Multi-Display-Umgebungen zu unterstützen. Dafür werden Daten über Umgebung und Nutzer gesammelt und ausgewertet, um die Umgebung und ihr Verhalten an die Wünsche der Nutzer anzupassen.

Anwendungsgebiete für Smart Meeting Rooms sind unter anderem, Nutzer bei der Präsentation oder Diskussion von Daten zu unterstützen, wobei ein wesentlicher Bestandteil das Präsentieren und Teilen von Informationen ist. Präsentieren und Teilen von Informationen bedeutet hierbei, diese auf großen Displayflächen anzuzeigen.

Bisher wurde dies durch das direkte Verbinden von persönlichen Geräten mit einem Projektor oder einem großen Bildschirm erreicht. Das Problem bei dieser Art der Informationsanzeige ist, dass sie wenig flexibel ist. Von einem Gerät wird eine spezifische Anzeige generiert und auf einer Displayfläche angezeigt, wobei die Position der präsentierten Information fix oder ein zusätzlicher manueller Aufwand erforderlich ist, um sie zu ändern. Werden außerdem Informationen von verschiedenen Quellen bereitgestellt, müssen sie zwangsläufig auf verschiedenen Displayflächen dargestellt

werden. Dadurch können zusammengehörende Informationen nicht nahe beieinander angezeigt werden, was z.B. ein Vergleichen erheblich erschwert.

In der vorliegenden Arbeit wird daher ein Konzept für die Informationspräsentation in Smart Meeting Rooms entwickelt. Ziel ist es, in einer Multi-Display-Umgebung alle Displayflächen für die Anzeige von Informationen nutzen zu können, die persönlichen Geräte der Nutzer für das Bereitstellen der Informationen zu verwenden und die Nutzer bei der Konfiguration der Anzeige automatisch zu unterstützen.

Für die Präsentation und Diskussion von Informationen reicht allerdings die Anzeige nicht aus. Um der Dynamik der Umgebung und den wechselnden Schwerpunkten von Präsentationen und Diskussion gerecht zu werden, müssen die Nutzer in die Lage versetzt werden, mit den dargestellten Informationen zu interagieren und die Anzeige entsprechend anzupassen.

Deshalb ist ein weiteres Ziel dieser Arbeit, ein Konzept für die Interaktion in Smart Meeting Rooms zu entwickeln. Dies soll ebenfalls den grundsätzlichen Eigenschaften und Zielen von Smart Meeting Rooms genügen. Dabei soll sowohl eine Nutzung verschiedenster persönlicher Interaktionsgeräte möglich sein als auch die Interaktion mit diesen persönlichen Interaktionsgeräten auf allen Displayflächen.

## 1.2. Herausforderungen und Ziele

Die grundlegenden Herausforderungen dafür lassen sich wie folgt zusammenfassen:

*Welche Informationen werden auf welcher Displayfläche für wen, wie dargestellt?*

Die erste Herausforderung ist hierbei das Erzeugen und Bereitstellen der Informationsrepräsentationen. Dabei sollen verschiedene persönliche Geräte verschiedener Nutzer mit unterschiedlichen Applikationen in die Lage versetzt werden, Informationsrepräsentationen zu erzeugen und für die Anzeige bereitzustellen. Bisherige Ansätze, die über ein direktes Verbinden von persönlichem Gerät und Projektor hinausgehen, sind dabei in der Regel auf bestimmte Betriebssysteme oder Applikationen eingegrenzt. Lediglich bildbasierte Ansätze sind in der Lage, die benötigte Flexibilität

zu gewährleisten. Diese erlaubten aber bisher keine Beschreibung der Informationsrepräsentationen.

Eine Beschreibung der anzuzeigenden Informationsrepräsentationen ist notwendig, um die zweite Herausforderung, die Verteilung der Informationsrepräsentationen auf die verschiedenen Displayflächen, zu adressieren. Das Ziel ist hierbei, automatisch eine Konfiguration der Anzeige zu ermöglichen. Um dies zu erreichen, müssen sowohl die Eigenschaften der Nutzer und der Displayflächen als auch die Eigenschaften der Informationsrepräsentationen berücksichtigt werden. Dabei sollte auch die Zusammengehörigkeit verschiedener Informationen beachtet werden.

Eine weitere Herausforderung ist, dass die Anzahl der Informationsrepräsentationen, die gleichzeitig angezeigt werden können, prinzipiell nicht beschränkt sein sollte. Multiple Informationsrepräsentationen von verschiedenen Quellen sollen auf einer Displayfläche angezeigt werden können. Damit wird im Gegensatz zu bisherigen Ansätzen nicht nur ermöglicht, mehr Informationsrepräsentationen gleichzeitig anzuzeigen, sondern es wird ebenso eine automatische Anzeige zusammengehörender Informationsrepräsentationen auf einer Displayfläche unterstützt.

Daraus ergibt sich wiederum die Aufgabe, diese multiplen Informationsrepräsentationen geeignet auf einer Displayfläche anzuzeigen. Ein Problem dabei ist, dass die Eigenschaften der anzuzeigenden Informationsrepräsentationen stark variieren können. Hier ist das Ziel, eine automatische Anordnung entsprechend der Eigenschaften von Displayfläche, Informationsrepräsentationen und Nutzer zu ermöglichen.

Smart Meeting Rooms werden in der Regel dafür verwendet, Informationen zu vermitteln oder zu diskutieren. Solche Präsentationen bzw. Diskussionen erfordern es, dynamisch neue Ansichten zu generieren, Informationen interaktiv in Beziehung zu setzen und dadurch zu weiteren Einsichten zu gelangen.

Daher ist neben der Anzeige der Informationen, eine weitere große Herausforderung, die Interaktion mit den dargestellten Informationsrepräsentationen. Hier ist das Ziel, dass alle Nutzer mit verschiedenen persönlichen Geräten mit allen dargestellten Informationen auf allen Displayflächen interagieren können. Eine Anforderung dabei ist, dass neben den klassischen Interaktionsgeräten wie Maus und Tastatur auch neue Geräte wie Wiimotes gleichzeitig verwendet werden können.

Neben den technischen Grundlagen für die Interaktion, stellt die Manipulation der Anzeige der Informationsrepräsentationen eine weitere Problemstellung dar. Ziel ist es daher, Methoden zur Verfügung zu stellen, die es den Nutzern erlauben, die Anzeige vollständig interaktiv zu modifizieren unabhängig davon, welches Gerät, welcher Nutzer oder welche Applikation die Informationsrepräsentationen erzeugt oder bereitstellt.

## 1.3. Ergebnisse und Struktur der Arbeit

In Kapitel 2 werden die Grundlagen für die Informationspräsentation in Smart Meeting Rooms erläutert. Dazu wird der Forschungsgegenstand vorgestellt und eine begriffliche Basis geschaffen. Danach wird der aktuelle Stand der Forschung zur Informationspräsentation in Multi-Display-Umgebungen vorgestellt.

In Kapitel 3 wird das *Smart View Management* vorgestellt, ein Konzept zur Informationspräsentation in Smart Meeting Rooms, das den Schwerpunkt dieser Arbeit bildet und in [RLS11] publiziert wurde. Zunächst wird in Abschnitt 3.1 eine spezifische Problembeschreibung erarbeitet. In Abschnitt 3.2 wird dann das Konzept der *Views* eingeführt. *Views* beschreiben visuelle Ausgaben von Applikationen und ermöglichen es, von der Art der anzuzeigenden Information, unabhängig von der zugrunde liegenden Applikation, zu abstrahieren. Auf dieser Basis wird der prinzipielle Lösungsansatz erläutert. Dazu wird die Generierung von *Views* durch unterschiedliche Nutzer, Geräte und Applikationen in Abschnitt 3.3 beschrieben. Die Zusammengehörigkeit von *Views* wird dann in Abschnitt 3.4 thematisiert und liefert die Grundlage für die automatische Anzeige der *Views*. Die automatische Verteilung der *Views* auf die Displayflächen wird durch das *Smart Display Mapping* in Abschnitt 3.5 beschrieben, gefolgt vom automatischen Layout in Abschnitt 3.6. Eine Anwendung des *Smart View Managements*, die in [RNS11] gezeigt wurde, wird anschließend in Abschnitt 3.7 beschrieben.

In Kapitel 4 wird die Interaktion thematisiert. Dafür wird zunächst ein Überblick über aktuelle Ansätze für die Interaktion in Smart Meeting Rooms gegeben (siehe Abschnitt 4.1). Danach wird das Smart Interaction Management, ein Konzept zur Interaktion in Smart Meeting Rooms, vorgestellt (siehe Abschnitt 4.2), das in [RLSS12] pu-

bliziert wurde. Zunächst wird das grundlegende Prinzip vorgestellt und danach, wie die Interaktionen erfasst, verteilt und ausgewertet werden.

Nach der Thematisierung der technischen Basis wird in den folgenden beiden Abschnitten die Interaktion mit der Anzeige der *Views* (Abschnitt 4.3) und mit dem Inhalt der *Views* (Abschnitt 4.4) behandelt. Ein Lösungsansatz für die Interaktion mit der Anzeige von *Views* wurde in [LRS10a] publiziert. Für die Interaktion mit dem Inhalt von *Views* wurden zwei Arbeiten vorgestellt [RLSS11, PRSB10].

Kapitel 5 liefert eine Zusammenfassung des Inhaltes der Dissertation und gibt einen Ausblick auf Themenstellungen, die in weiteren Arbeiten adressiert werden können.

# 2. Stand der Forschung

In diesem Kapitel wird der aktuelle Stand der Forschung zum Thema der Informationsanzeige in Smart Meeting Rooms dargestellt.

Um zunächst den Forschungsgegenstand zu charakterisieren, werden in Abschnitt 2.1 Smart Environments und in Abschnitt 2.1.1 Smart Meeting Rooms im Allgemeinen sowie im Abschnitt 2.1.2 das Smart Lab Rostock im Speziellen vorgestellt.

Anschließend wird der aktuelle Stand der Forschung im Bereich der Informationsanzeige in Smart Meeting Rooms in Abschnitt 2.2 beleuchtet. Dafür werden Problemstellungen identifiziert, die bei der Informationspräsentation in solchen Umgebungen auftreten. Die vorgestellten aktuellen Ansätze werden dann anhand der Problemstellungen bewertet. Es werden Arbeiten vorgestellt, die einzelne der Problemstellungen adressieren, allerdings ohne den Anspruch, ein umfangreiches System zur Informationspräsentation darzustellen.

In den nachfolgenden Kapiteln wird jeweils separat der Stand der Forschung zu den einzelnen, im Rahmen dieser Dissertation erarbeiteten Lösungen präsentiert.

## 2.1. Smart Environments

Die Idee von *Smart Environments* beginnt mit dem Begriff des *Ubiquitous Computing* (*dem allgegenwärtigen Rechnen*), den Mark Weiser 1991 in [Wei91] begründete. Er beschreibt dort "eine physikalische Welt die stark und unsichtbar verwoben ist mit Sensoren, Aktuatoren, Displays und rechnenden Elementen eingebettet in Objekte des täglichen Gebrauchs und verbunden mit einem durchgehenden Netzwerk." (Übersetzung aus [Wei91]).

Damit wird im Ubiquitous Computing schon die Vision des *Invisible Computer* von Donald Norman [Nor99] mit einbezogen. Dort beschreibt Norman, dass sich "Computer so stark ändern werden, dass sie aus dem Blick verschwinden werden" (Übersetzung aus [Nor99]).

Durch die Vielzahl verschiedener Geräte, die Vielzahl ihrer Operationen und vor allem der Unsichtbarkeit der Geräte wird es nach Encarnação und Kirste erforderlich, Strategien, Modelle und Technologien für die Selbstorganisation zu entwickeln. Dadurch kann dieses Geräteensemble an die Bedürfnisse und Wünsche des Nutzers angepasst werden [EaK05].

Auch Tennhouse setzt auf diesem Gedanken auf und formuliert das Prinzip des *Proactive Computing* [Ten00, WPT03]. Basierend auf der Allgegenwart vieler verschiedener Geräte stellt er die These auf, dass diese Geräte nicht mehr sinnvoll vom Menschen bedient werden können. Allerdings soll der Mensch aber in der Lage sein alle Prozesse der Geräte zu beeinflussen. Daher definiert er das *Proaktive Computing* als Rollenverschiebung des Menschen vom Observer und Controller hin zum Supervisor, der in der Lage ist, das Gesamtsystem zu konfigurieren und in jedem Schritt manuell einzugreifen. Dabei wird er aber von einer Vielzahl von Automatismen unterstützt.

Basierend auf dieser Allgegenwart der Geräte, der Unsichtbarkeit der Geräte, den darauf definierten Automatismen, Strategien und Modellen und dem *Proactive Computing* definiert sich das Ziel der Smart Environments nach Cook und Das als "eine kleine Welt, wo alle Arten von smarten Geräten kontinuierlich arbeiten um das Leben der Nutzer komfortabler zu machen." (Übersetzung aus [CD04]).

Die Definition der Smart Environments nach Youngblood et al. [YHHC05] fordert zusätzlich noch explizit die ständige Beobachtung der Nutzer und der Umgebung durch Sensoren, um die Nutzbarkeit der Umgebung zu verbessern.

Damit ergibt sich eine Definition von Smart Environments, die das Ziel haben, die Nutzer bei ihrer Arbeit und selbst bei alltäglichen Aktivitäten zu unterstützen [C+98, AEa06]. Dafür sammeln sie, wie in der Definition schon verankert, Informationen über die Umgebung und über die sich darin befindlichen Nutzer. Diese Informationen werden dann verwendet, um die vom Nutzer angestrebte Situation zu ermitteln und die Umgebung den Wünschen entsprechend anzupassen [YHHC05, CD07].

Aus dem allgemeinen Begriff der Smart Environments entstanden eine ganze Reihe verschiedener spezifischer intelligenter Umgebungen, die jeweils einen konkreten Anwendungsfokus verfolgen. Beispiele hierfür sind Smart Meeting Rooms [AEa06, HK05b, HK05a], Smart Conference Room [HK05a], Smart Office [AHF+02, LGMLC01, RMSF10] und Smart Class Room [Abo99, FFH00, SXX+03].

### 2.1.1. Smart Meeting Rooms

Diese Arbeit ist im Rahmen des Graduiertenkollegs *MuSAMA* (Multimodal Smart Appliance Ensembles for Mobile Applications) entstanden. Das Ziel von *MuSAMA* ist es, in einem ad-hoc Geräteensemble eine kohärente Assistenz des Menschen zu verwirklichen. Dafür sollen Konzepte und Verfahren entwickelt werden, die auf Basis von Sensordaten und Interaktionsereignissen eine möglichst sinnvolle Unterstützung liefern [hom13].

Das Leitszenario des Graduiertenkollegs MuSAMA ist der Smart Meeting Room. Die Hauptaufgabe von Smart Meeting Rooms ist es, ein Meeting von mehreren Nutzern zu unterstützen. Szenarien für Smart Meeting Rooms sind in der Regel Präsentationen (bzw. Vorträge) und wissenschaftliche Diskussionen [EaK05]. Bei solchen Szenarien ist es wünschenswert, dass Nutzer ihre persönlichen Geräte ohne großen Konfigurationsaufwand verwenden können [HK05a, AEa06]. Es sind aber auch stationäre Geräte notwendig, da z.B. Trackingsysteme oder große hochauflösende Beamer sich nicht schnell vom Nutzer mitbringen lassen, sondern aufwändig installiert und gegebenenfalls vorher kalibriert werden müssen.

Um die Verwendung persönlicher Geräte zu unterstützen, sind Smart Meeting Rooms in der Regel ad-hoc Geräteumgebungen [AEa06, BMK+00, HK05a], die zusätzlich durch vorhandene Geräte ergänzt werden [Thi10]. Darüber hinaus ist es besonders hilfreich, wenn eine *walk-up-and-use* Funktionalität der Umgebung bereitgestellt werden kann [SER03, WJF+09]. *Walk-up-and-use* Funktionalität heißt, dass Nutzer ihre persönlichen Geräte spontan ohne vorherige Installationen von Software verwenden können.

Smart Meeting Rooms sind in der Regel Multi-Display-Umgebungen [BRH+08, WLSS09, HK05a, HNC+09, JFW04]. Das bedeutet, dass im Smart Meeting Room bereits multiple Displayflächen vorhanden sind, die auch von mehreren Nutzern verwendet werden.

Durch die Einbindung der persönlichen Geräte sind Smart Meeting Rooms darüber hinaus heterogene Multi-Display-Umgebungen. Neben den bereits vorhandenen Displayflächen reichen die Displays der persönlichen Geräte von Smart Phones über Tablet Computer bis zu mitgebrachten Beamern [AEa06].

Dabei unterscheiden sich die Displayflächen nicht nur in ihren offensichtlichen Eigenschaften wie z.B. Größe, Auflösung und Pixeldichte, sondern auch in der Art und Weise der Nutzung. So werden Desktop Displays in der Regel von einer Entfernung von 70 cm betrachtet und große TVs oder Beamer von 3 m Entfernung [TQD09].

Weiterhin wurde von Tan et al. gezeigt, dass die Effektivität bei der Nutzung verschiedener Displayflächen von der zu erledigenden Aufgabe abhängt. So sind z.B. große Displayflächen für räumliche Aufgaben besser geeignet als kleine Displayflächen, obwohl dessen technische Eigenschaften gleich sind [TGSP06].

### 2.1.2. Smart Lab Rostock

Die konkrete Forschungsumgebung des Graduiertenkollegs MuSAMA und auch dieser Arbeit ist das *Smart Lab Rostock*. Dieses Lab ist ein Smart Meeting Room, der grundsätzlich darauf abzielt, alle der oben genannten Eigenschaften zu erfüllen.

Im Folgenden wird das *Smart Lab Rostock* als spezifische Umgebung beschrieben. Dazu wird zuerst der grundsätzliche Aufbau skizziert und anschließend die installierte Hardware und die Softwareinfrastruktur dargestellt.

Das Smart Lab hat eine Größe von ca. 60 Quadratmetern und eine Raumhöhe von etwa vier Metern. In der Mitte des Raums befinden sich rechteckig angeordnete Tische mit jeweils zwei Stühlen (Bild 2.1). Diese Anordnung eignet sich besonders gut für das Leitszenario des Meetings, da die beteiligten Nutzer sich von Angesicht zu Angesicht unterhalten können. Im Raum sind an drei der vier Wände Leinwände

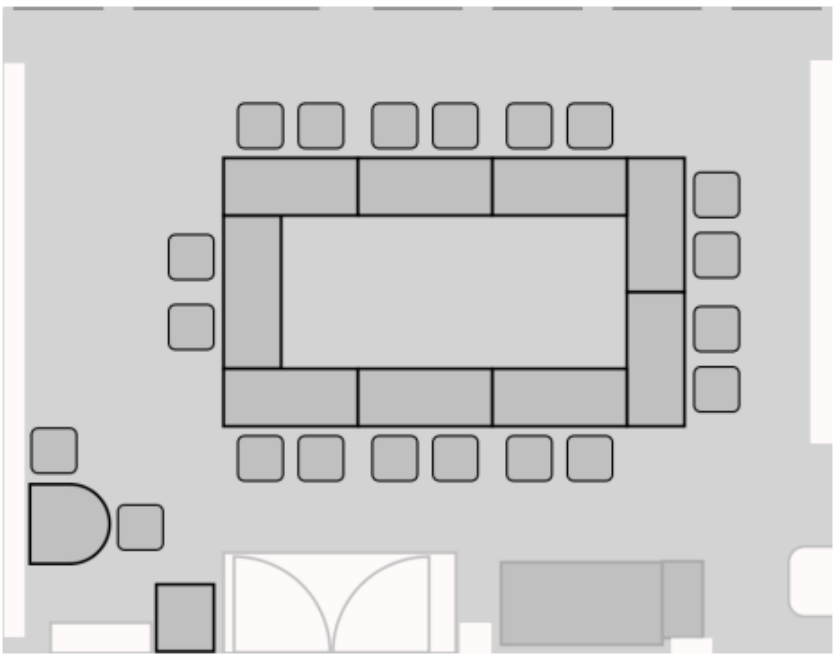


Abbildung 2.1.: Smart Lab Rostock im Grundriss.

für Beamer-Projektionen angebracht. Damit lassen sich bis zu sieben verschiedene Beamer-Projektionen gleichzeitig anzeigen. Für die Positionserkennung der Nutzer werden zwei unabhängige Systeme verwendet. Zum einen das *Ubisense* Tracking System [Ubi13a, Ubi13b] bei dem kleine portable *Tags* durch Ultrabreitband-Signale erkannt werden. Dies ermöglicht eine Positionserkennung mit Identifikation des Nutzers. Zum anderen wurde der Fußboden mit einem *SenseFloor* [Gmb10, Sha13] versehen, welches eine weitaus genauere Lokalisation der Nutzer ermöglicht, ohne den Nutzer allerdings direkt identifizieren zu können.

Die Softwareinfrastruktur wird durch die Middleware namens *Helferlein* gebildet [BN12, BRK10]. Diese Middleware basiert auf einer verteilten Infrastruktur, die ein *publish/subscribe* System verwendet, um *events* zu verteilen. Die Daten der Middleware werden dabei in Tupeln gespeichert.

Ziel dieser Dissertation ist es, neue Konzepte für die Informationsanzeige für das Smart Lab zu entwickeln. Dazu soll im Folgenden der aktuelle Stand der Forschung vorgestellt werden.

## 2.2. Informationspräsentation in Smart Meeting Rooms

Bei der Informationspräsentation in Smart Meeting Rooms besteht die Herausforderung darin, Probleme der Informationspräsentation mit den Ansprüchen von Smart Environments, insbesondere Smart Meeting Rooms, in Übereinstimmung zu bringen.

Daraus ergeben sich Fragestellungen für die Informationspräsentation in Smart Meeting Rooms, anhand derer im Weiteren die Arbeiten aus der Literatur beleuchtet und eingeordnet werden.

### 2.2.1. Grundlegende Fragestellungen

Bei der Anzeige von Information gibt es eine Quelle und eine Senke der anzuzeigenden Information [MLM+12]. Daher steht am Anfang jeder Präsentation die Frage der *Quelle der Informationsrepräsentation*. Da nur aktuell relevante Informationen angezeigt werden sollen, muss die *Auswahl* der anzuzeigenden Information getroffen werden. Bei der Senke, der Anzeige der Informationen, stellt sich die Frage der *Anordnung* der Informationen, also des Layouts auf der Ausgabefläche [Ali09]. Außerdem ist die Manipulation der Darstellung notwendig, um eine Anpassung an die Bedürfnisse des Nutzers zu erlauben. Dafür ist die *Interaktion* von großem Interesse [LSST11].

Eine grundlegende Eigenschaft von Smart Meeting Rooms ist, dass die beteiligten Geräte *ad-hoc* ein gemeinsames Ensemble bilden [AEa06]. Diese Eigenschaft gilt es auch bei der Informationspräsentation zu berücksichtigen. Smart Meeting Rooms wurden zudem als Multi-Display-Umgebungen charakterisiert [BRH+08]. Deshalb muss für die Informationsanzeige die *Zuordnung* der Informationsrepräsentationen zu einer Displayfläche geklärt werden, d.h. auf welcher Displayfläche die Information angezeigt werden soll [Hei09]. Bei der Anzeige von Informationen ist hier im Speziellen die Frage nach der *Zusammengehörigkeit* zu beantworten, da diese Informationen auch zusammen dargestellt werden sollten. Ziel der Smart Meeting Rooms ist es, den *Nutzer* bei seiner Arbeit zu unterstützen [CD04]. Dies gilt auch für die Informationsanzeige, weshalb die *Nutzer* bei der automatischen *Assistenz* für die Konfiguration der Anzeige, berücksichtigt werden muss.

Daraus ergeben sich für die Informationspräsentation in Smart Meeting Rooms folgende Problemstellungen:

**Bereitstellen der anzuzeigenden Informationen** Was ist die Quelle der Informationsrepräsentationen? Ist es dabei möglich, private Geräte zu nutzen? Lassen sich diese ad-hoc für die Informationsanzeige nutzen? Welche technischen Voraussetzungen werden an die Geräte gestellt? Wie wird eine Informationsrepräsentation zur Anzeige zur Verfügung gestellt? Wie erfolgt die Auswahl, welche der zur Verfügung stehenden Informationsrepräsentationen auf großen Displayflächen angezeigt werden? Wer wählt die anzuzeigenden Informationen aus?

**Anzeige der Informationsrepräsentationen** Welche Mechanismen stehen zur Verfügung, um die Zusammengehörigkeit von Informationsrepräsentationen zu definieren? Welcher Displayfläche werden die Informationsrepräsentationen zugeordnet? Wie wird diese Zuordnung gesteuert? Gibt es dafür Assistenzfunktionen? Können mehrere Informationsrepräsentationen auf einer Displayfläche dargestellt werden und wie werden diese dann angeordnet?

**Interaktion** Wie wird die Anzeige der Informationsrepräsentationen angepasst? Wie werden dabei die Eigenschaften der Nutzer berücksichtigt? Welche Assistenz wird dem Nutzer bei der Informationsanzeige geboten? Welche automatischen Konfigurationen werden dem Nutzer angeboten? Wie kann mit den dargestellten Informationsrepräsentationen interagiert werden? Welche Geräte können dafür verwendet werden? Wer darf mit welchen Darstellungen interagieren?

### 2.2.2. Bisherige Lösungsansätze

In diesem Abschnitt werden nun verschiedene Konzepte und Systeme zur Informationsdarstellung in Smart Meeting Rooms und heterogenen Multi-Display-Umgebungen vorgestellt. Diese Systeme werden hinsichtlich der oben aufgelisteten Problemstellungen beschrieben. Zu jedem System wird das grundsätzliche Setup der Umgebung beschrieben, soweit dies aus den Publikationen bekannt ist. Zusätzlich wird der Fokus der jeweiligen Arbeit benannt, um das damit verbundene Ziel der Informationsanzeige einordnen zu können. Abschließend erfolgt eine Zusammenfassung.

Für die Informationsanzeige in heterogenen Multi-Display-Umgebungen bzw. in Smart Meeting Rooms existieren einige wenige Arbeiten in der Literatur. Grundsätzlich lassen sich diese Systeme in drei Kategorien einteilen:

**Ausnutzung von Betriebssystemspezifischen Eigenschaften** Bei der betriebssystembasierten Informationsanzeige werden grundsätzliche Funktionen des Betriebssystems ausgenutzt, um eine verteilte Informationsanzeige zu erreichen. Grundsätzlich werden dafür die Betriebssystemeigenen Techniken zum Rendering verwendet. Dazu gehören z.B. XWindow bei Linux [SG86] oder GDI unter Windows [GDI13].

**Datenbasierte Informationsanzeige** Bei der datenbasierten Informationsanzeige werden Daten in Form von Datenbanken oder Dateien vom Quellrechner an den Zielrechner versendet. Dort werden die Informationsrepräsentationen der Daten generiert und angezeigt.

**Bildbasierte Informationsanzeige** Bei der bildbasierten Informationsanzeige werden am Quellrechner die Informationsrepräsentationen erzeugt. Diese werden dann als Bilddaten zum Zielrechner gesendet, wo sie dann angezeigt werden.

Die folgend vorgestellten Systeme sind jeweils nach diesen drei Kategorien sortiert.

#### 2.2.2.1. Ausnutzung von Betriebssystemspezifischen Eigenschaften

**Deskotheque**

Ein sehr umfassendes Framework zur Informationspräsentation in heterogenen Multi-Display-Umgebungen ist die **Deskotheque**. Im Rahmen der Dissertation von Manuela Waldner [Wal11] wurde ein System entwickelt, das es multiplen Nutzern erlaubt, Informationsrepräsentationen auf multiplen Displayflächen anzuzeigen.

Das Basissetup (siehe Abbildung 2.2) beinhaltet drei Projektoren, die frontal vor zwei Arbeitsplätzen angeordnet sind. Diese Arbeitsplätze haben jeweils einen privaten Monitor mit Maus und Tastatur. Zwischen den beiden Arbeitsplätzen ist ein Tisch aufgestellt, auf dem ein weiterer Projektor projiziert. Die Projektoren an der Frontwand überlagern sich, wodurch eine nahtlose Anzeigefläche erzeugt werden kann.

Die Grundlage der *Deskotheque* bildet ein Window Management. Dies ermöglicht es, multiple Displayflächen zu einer großen nahtlosen Anzeigefläche zu vereinen [WPS08], wobei durch das *Composing Window Management* die geometrischen Verzerrungen und die Helligkeitsunterschiede bei der Überlagerung verschiedener Projektoren kamerabasiert erkannt und ausgeglichen werden.

Basierend auf dieser Multi-Display-Anzeige wurde das *Deskotheque* Framework entworfen [WLSS09]. Der Fokus dieses Frameworks ist es, eine Informationsanzeige in einer Multi-Display-Umgebung zu realisieren, um die kooperative Arbeit von zwei Nutzern zu unterstützen. Neben dem *Composing Window Management* wurden im Rahmen der Deskotheque auch die Themen Interaktion und Verknüpfung von Informationen thematisiert.

Quelle der Informationsrepräsentationen, die angezeigt werden sollen, sind die stationären Rechner an den beiden Arbeitsplätzen und ggf. vom Nutzer mitgebrachte Laptops. Das *Deskotheque* Framework fällt in die Klasse der Betriebssystemspezifischen Informationsanzeigesysteme. Hier wird als Betriebssystem Linux vorausgesetzt mit einem spezifischen XWindow Client. Dieser XWindow Client ist mit einem zentralen XWindow Server verbunden und ist somit in der Lage, visuelle Inhalte per Netzwerk zu verschicken und auf anderen Displayflächen darzustellen.

Durch diese betriebssystemabhängige Erzeugung der Informationsrepräsentationen ist evtl. ein langwieriger Installationsprozess notwendig, um ein privates Gerät in dieser Umgebung nutzen zu können. Nach dieser Installation kann dieses private Gerät allerdings ad-hoc verwendet werden. Für die Nutzbarkeit von privater Software sind die gleichen Probleme zu lösen. Ist die Applikation allerdings unter Linux lauffähig, so ist eine ad-hoc Nutzung möglich.

Die Informationsrepräsentationen, die auf der großen Anzeigefläche dargestellt werden sollen, sind Fenster von Applikationen, die auf einem Arbeitsplatzrechner oder Laptop gestartet wurden. Der Nutzer ist in der Lage, dieses Fenster auf die große Displayfläche zu ziehen und es so dort anzuzeigen.

Die Zusammengehörigkeit der Informationsrepräsentationen, also der dargestellten Fenster, wird nicht thematisiert. Die Zusammengehörigkeit von dargestellter Information in diesen Fenstern wird allerdings sehr umfassend durch das visual Linking

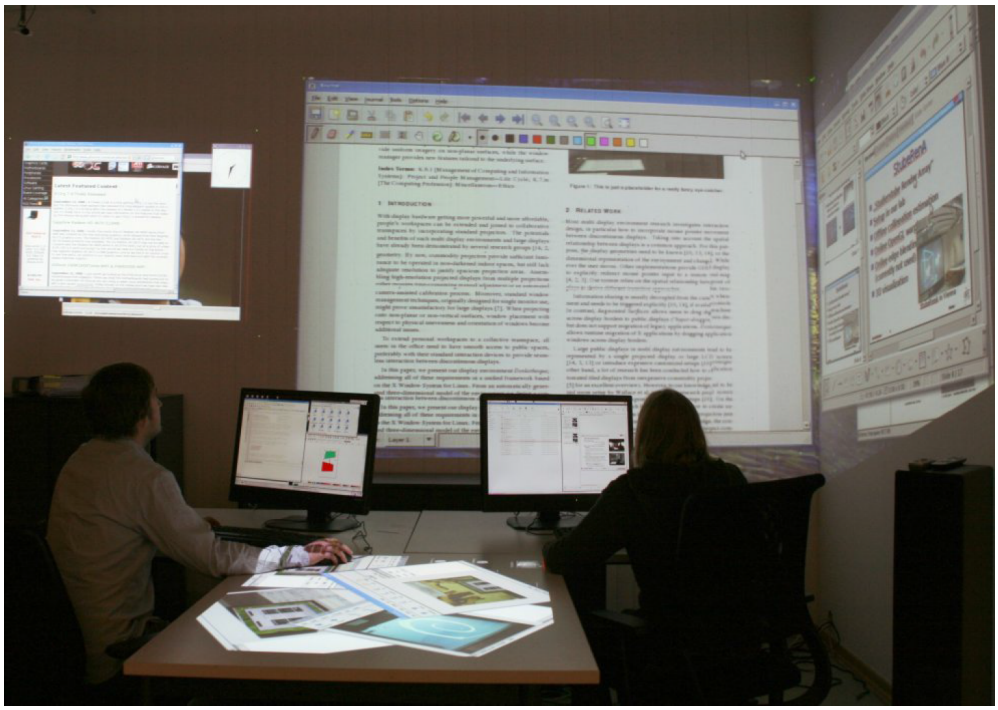


Abbildung 2.2.: Deskotheque. Bild aus [PWS09]

thematisiert. In [WPL+10] stellen Waldner et al. einen Mechanismus vor, um Informationen, dargestellt von verschiedenen Applikationen, zu verlinken. Steinberger et al. erweitern diesen Ansatz und präsentieren in [SWS+11] die *context preserving visual links*. Hier wird eine Analyse der visuellen Aufmerksamkeit (visual salience) vorgenommen, wodurch sich die Links so zeichnen lassen, dass dargestellte Informationen nicht überdeckt werden.

Die Zuordnung der Fenster zu den Displayflächen erfolgt interaktiv durch den Nutzer, indem er in die Lage versetzt ist, Fenster von seinem privaten Display auf die große Anzeigefläche zu verschieben. Damit wird gleichzeitig die Interaktion des anderen Nutzers mit dem Fenster freigegeben. Eine automatische Zuordnung erfolgt nicht.

Die Anordnung der Fenster auf dem privaten Display und auf der großen Anzeigefläche kann ebenfalls manuell vorgenommen werden. Darüber hinaus stellen Waldner et al. in [Wal11] einen Layout Mechanismus vor, der es ermöglicht, Fenster an die geometrischen Grenzen der einzelnen Projektionen anzupassen.

Eine weitere Layout Komponente stellen Waldner et al. in [WSGS11] vor. Hier wird die Sichtbarkeit von Informationen in verdeckten Fenstern hergestellt. Dafür werden wiederum mit Hilfe einer Analyse der visuellen Aufmerksamkeit (visual salience) die einzelnen Fenster untersucht. Dadurch wird dann bestimmt, welche Teile eines Fensters Informationen enthalten und somit nicht verdeckt werden dürfen und welche keine Informationen enthalten. In den unwichtigen Teilen des Fensters werden dann mittels Blending die darunter liegenden Informationen dargestellt.

Ein großer Fokus des *Deskotheque* Frameworks ist die Interaktion mit den dargestellten Informationen. Dafür wurden in [WPKS10] zwei grundlegende Techniken für die mausbasierte Navigation in Multi-Display-Umgebungen entwickelt: *Free Navigation* und *Path Navigation*. Beide Techniken nutzen das räumliche Modell der Displayanordnung, die im Vorfeld detektiert wurde. Basierend darauf und ausgehend von der angenommenen Nutzerposition an einem der zwei Arbeitsplätze wird dann perspektivisch korrekt die Position des Mauszeigers angepasst. Diese beiden Techniken wurden dann in [WS10] evaluiert. Um die Navigation bei der Überbrückung großer Distanzen zusätzlich zu erleichtern, wurde in [WKS10] ein *Pointer Warping* für Multi-Display-Umgebungen vorgestellt. Eine Einbindung oder Unterstützung weiterer Interaktionsgeräte zur Navigation ist nicht vorhanden.

Der Nutzer wird im Rahmen des *Deskotheque* Frameworks nicht spezifisch beachtet. Lediglich die angenommene Position des Nutzers an einem der Arbeitsplätze wird bei der Mausnavigation mit berücksichtigt. Eine Berücksichtigung für die Adaption der Darstellungen findet nicht statt.

Zusammenfassend lässt sich feststellen, dass das *Deskotheque* Framework ein sehr umfangreiches und sehr leistungsstarkes System zur Informationsanzeige in heterogenen Multi-Display-Umgebungen darstellt. Neben dem Layout der Fenster steht vor allem die Mausnavigation in dieser Umgebung im Mittelpunkt.

Für den Einsatz in Smart Meeting Rooms sind aber noch weitere Probleme zu lösen. Es wird vorausgesetzt, dass als Quelle Informationsrepräsentationen ein Rechner mit einem spezifischen Betriebssystem verwendet wird. Dies schließt die Nutzung von mobilen Geräten wie etwa Smart Phones und Tablets ebenso aus wie die Nutzung von Geräten mit alternativen Betriebssystemen. Diese Einschränkung macht eine ad-

hoc Nutzung sehr schwierig. Der Einsatz beliebiger Software wird erheblich erschwert und die *Deskotheque* bietet nur ein eingeschränktes Maß an Assistenz. Der Nutzer wird bei der Darstellung der Fenster insoweit unterstützt, als dass diese an den geometrischen Grenzen der Beamer ausgerichtet werden können. Weiterhin werden Verdeckungen der Fenster untereinander automatisch aufgelöst. Das initiale Layout wird allerdings weiterhin vom Nutzer interaktiv vorgenommen.

**WinCuts**

Ein weiteres System für eine Informationsdarstellung in heterogenen Multi-Display-Umgebungen ist **WinCuts**. In [TMC04] stellen Tan et al. ein System vor, um die private Bildschirmfläche besser ausnutzen zu können und um Teile von Fenstern auf einer großen Displayfläche für die Zusammenarbeit mehrerer Nutzer darzustellen (siehe Abbildung 2.3).

Der Fokus war hier ursprünglich darauf gerichtet, die wichtigsten Teile einer Applikation zu selektieren und nur diesen Inhalt separat darzustellen. So soll die Displayfläche auf dem privaten Gerät besser ausgenutzt werden können. Dieser Ansatz wurde weiterentwickelt, um auf einer großen Displayfläche auch diese Fensterausschnitte darstellen zu können.

Die Quelle der Informationsrepräsentationen sind private Geräte der Nutzer, auf denen Ausschnitte eines geöffneten Fensters vom Nutzer selektiert werden. Diese Ausschnitte (genannt *WinCuts*) stellen die Informationsrepräsentationen dar, die im Raum angezeigt werden sollen. Auch *WinCuts* ist in der Klasse der Betriebssystemabhängigen Systeme zur Informationsdarstellung anzusiedeln. Als Betriebssystem wird Windows vorausgesetzt. Der Fensterinhalt wird mit Hilfe der *Win32 Graphics Device Interface (GDI) API* abgegriffen und per Netzwerk verschickt. Vorteil hierbei ist, dass der Inhalt selbst von minimierten, also nicht sichtbaren Fenstern, angezeigt werden kann.

Eine ad-hoc Nutzung dieses Systems ist nur eingeschränkt möglich. Zum einen wird Windows als Betriebssystem vorausgesetzt, zum anderen muss die verwendete Software die GDI API implementieren.

Die Auswahl der *winCuts* erfolgt manuell durch den Nutzer. Dieser gibt außerdem an, auf welchem anderen Gerät der von ihm definierte *winCut* angezeigt werden soll.

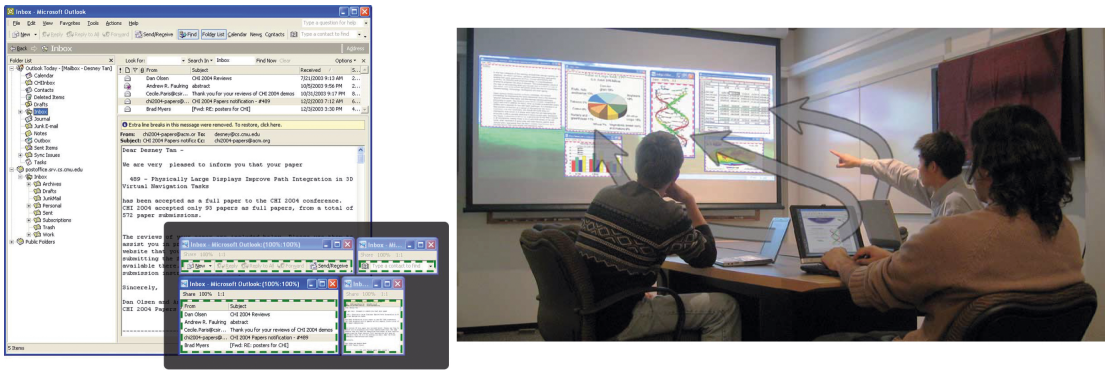


Abbildung 2.3.: WinCuts. Bild aus [TMC04]

Dabei steht kein Mechanismus für die Definition der Zusammengehörigkeit von *winCuts* zur Verfügung.

Für die Interaktion wird ein System namens *Visitor* verwendet welches es ermöglicht, Mausinteraktionen auf einen anderen Rechner zu versenden.

Zusammenfassend bietet *WinCuts* eine einfache Möglichkeit, ausgewählte Inhalte mit anderen Nutzern zum Zwecke der Zusammenarbeit zu teilen. Dabei ist auf Grund der Betriebssystemabhängigkeit die spontane Nutzung eingeschränkt. Auch eine Assistenz bei der Anordnung oder Anzeige der *WinCuts* ist nicht vorhanden.

#### 2.2.2.2. Datenbasierte Informationsanzeige

**iRoom – Stanford Interactive Workspace**

Ein prominenter Vertreter der Datenbasierten Informationsanzeige ist der *iRoom* [JFW02]. Das Stanford Interactive Workspace Projekt wurde initiiert, um die Interaktion mit hochauflösenden Displayflächen zu untersuchen. Der Fokus besteht hier in der Einbettung von großen Displayflächen in eine *ubiquitous computing* Umgebung.

Das Setup des *iRooms* beinhaltet drei touchsensitive große Displayflächen an einer Wand, eine große Displayfläche mit Stiftinteraktion in der Frontwand und ein Tisch mit eingebauter großer Displayfläche in der Mitte des Raums. Zusätzlich sind diverse

Abbildung 2.4.: iRoom – Stanford Interactive Workspace. Bild aus [JFW02]

Mikrophone, Interaktionsgeräte und Netzwerkschnittstellen im Raum verbaut (siehe Abbildung 2.4). Dies macht den Raum von den Möglichkeiten der Ausstattung her zu einem Smart Room.

Fokus war hier die Softwareinfrastruktur und das damit einhergehende Modell der Interaktion, die hervorgehobene Nutzung von großen interaktiven Displayflächen, zum Teil mit touch Interaction, und die Kollaboration mit anderen Forschergruppen, um konkrete Anwendungen zu schaffen. Für die Informationsanzeige wurden drei Ziele definiert: (1) Daten zwischen Geräten und Applikationen zu verschieben, die an verschiedene Displayflächen angeschlossen sind, (2) die Interaktion von jedem Nutzer auf jeder Displayfläche zu erlauben und (3) die automatische Koordination heterogener Applikationen zu ermöglichen.

Die Quelle der Informationsrepräsentationen sind im *iRoom* die Geräte, die direkt an den Displayflächen angeschlossen sind. Der Ansatz ist hier, dass Daten von einem persönlichen Gerät mit Hilfe einer Middleware (iROS) [BRTF02] an eine Applikation übertragen werden, wo sie dann gerendert und angezeigt werden. Die *iROS* Middle-

ware besteht dabei grundsätzlich aus *DataHeap* und *EventHeap*, mit deren Hilfe Interaktionen und Daten übertragen werden können.

Ist eine Applikation an die Middleware angepasst worden, so ist eine ad-hoc Nutzung persönlicher Geräte möglich. Die vorherige Anpassung der Software ist allerdings notwendig und kann nicht spontan durchgeführt werden. Durch die Nutzung einer plattformunabhängigen Middleware ist die Nutzung von persönlichen Geräten nicht an bestimmte Betriebssysteme gebunden.

Die anzuzeigenden Informationsrepräsentationen werden von den Nutzern ausgewählt. Die Zuordnung zu den Displayflächen erfolgt durch das Starten der Applikation auf dem zugehörigen Computer.

Um eine nahtlose Interaktion mit den dargestellten Informationsrepräsentationen zu ermöglichen, wurde *PointRight* entwickelt [JHW00, JFW02]. Dafür wird ein logisches Mapping der Displayflächen zueinander erstellt. Durch den *EventHeap* [JF02] werden Interaktionen von persönlichen Interaktionsgeräten an die großen Displayflächen weitergeleitet. Somit ist es möglich, dass jeder Nutzer mit den Informationsrepräsentationen auf jeder großen Displayfläche interagieren kann.

Im *iRoom* wird der Fokus auf Interaktion und Koordination der Applikationen durch eine spezifische Middleware gerichtet. Deshalb ist hier keine weitere automatisierte Assistenz vorhanden, die den Nutzer bei der Informationsdarstellung unterstützen kann. Um die Konfiguration des Raums aber dennoch zu vereinfachen, wurde ein Konfigurationstool namens *iCrafter* entwickelt. Damit lassen sich z.B. Lampen und Beamer interaktiv steuern. Weiterhin wurde der *SmartPresenter* geschaffen [JFW04], womit es auf Basis der *iROS* Middleware möglich ist, einen Foliensatz flexibel auf mehreren Displayflächen anzuzeigen.

Zusammenfassend fokussiert der *iRoom* auf die Koordination von Applikationen und die Interaktion mit diesen Applikationen. Statt fertig gerenderter Bilder werden die zugrunde liegenden Daten versendet und an den jeweiligen Displayflächen gerendert. Durch dieses Verfahren ist die ad-hoc Informationsanzeige sehr einfach möglich, wenn die verwendete Software für den Raum angepasst wurde. Die Anpassung der Software ist dabei allerdings sehr aufwendig. Die Heterogenität der Displayflächen und der Nutzer wird nicht adressiert, ebenso wenig die Assistenz.

**Adapting a Single-user, Single-display Molecular Visualization Application for Use in a Multi-user, Multi-Display Environment**

Ein weiterer Vertreter der Datenbasierten Informationsanzeige ist die Arbeit von Forlines et al. Hier wurde eine Applikation, die für einen Nutzer und eine Displayfläche konzipiert wurde, für die Nutzung von mehreren Nutzern in einer Multi-Display-Umgebung angepasst [FES+06, FL08].

Die Multi-Display-Umgebung besteht hier aus einem Tisch mit touchsensitiver Displayfläche, zwei großen Displayflächen und einem TabletPC.

Das Hauptaugenmerk besteht hierbei darin, die Applikation so anzupassen, dass automatisch auf den verschiedenen Displayflächen andere Ansichten desselben Datensatzes zu sehen sind. Der entwickelte Ansatz wird an zwei verschiedenen Applikationen demonstriert. Zum einen an einer Geographischen Anwendung [FES+06] (Google Earth [Goo13]) und zum anderen an einer Applikation zur 3D Repräsentation von Molekülstrukturen [FL08] (Jmol [Jmo13]).

Der Fokus der genannten Arbeiten liegt darin, auf verschiedenen Displayflächen jeweils andere Ansichten anzeigen zu können und die Interaktionen der Nutzer dabei mit den anderen Ansichten zu koordinieren. Für die verschiedenen Displayflächen werden dafür jeweils verschiedene koordinierte Instanzen der gleichen Applikation gestartet.

Die jeweiligen Displayflächen zeigen dabei voreingestellte Ansichten an. Durch die Interaktion auf dem Touchtable werden die Ansichten auf allen Displayflächen manipuliert. Die Nutzung persönlicher Geräte ist hier nicht vorgesehen. Auch eine dynamische Zuordnung von Ansicht zu Displayfläche ist ausgeschlossen.

Dieses System ist für konkrete Anwendungen konzipiert worden. Durch die Anpassung der Applikationen für die Koordination untereinander ist ein hoher Aufwand bei der Implementierung notwendig. Auch sind die verschiedenen Ansichten jeweils auf die konkrete Software zugeschnitten. Dieser Ansatz ist deshalb nicht ohne weiteres auf andere Probleme und Applikationen übertragbar.

Abbildung 2.5.: Multi-User Multi-Display Setting. Bild aus [FES+06]

**Dynamo**

*Dynamo* ist ein Vertreter der Datenbasierten Informationsanzeige [IBR+03]. Der Fokus liegt hier auf dem schnellen Austausch von Medieninhalten. Es wird eine *walk-up-and-use* Funktionalität bereitgestellt, die es erlaubt, ohne jegliche Vorbereitung die gewünschten Informationen anzuzeigen.

Ein konkretes Hardware Setup gibt es für *Dynamo* nicht. Die Vorgabe ist, dass ein Computer eine große Displayfläche oder mehrere Displayflächen bis hin zu einer Display-Wall betreibt.

Die Quellen der Informationsrepräsentationen sind hier keine persönlichen Geräte. Es werden mobile Datenträger (USB-Stick, mobile Festplatte, etc.) an den zentralen Computer angeschlossen, der die Displayflächen betreibt. Die Medien (Videos, Bilder, etc.) können dann durch *Dynamo* eigene Applikationen dargestellt werden. Zusätzlich können im *Dynamo* System zusätzliche Applikationen wie z.B. Browser oder Notizblock gestartet werden.


Abbildung 2.6.: *Dynamo*. Bild aus [IBR+03]

Die angeschlossenen Datenträger werden dann für jeden Nutzer in einem persönlichen Bereich auf der großen Displayfläche dargestellt, wo der Nutzer interaktiv die anzuzeigenden Informationen auswählen kann.

Um die Zusammengehörigkeit von dargestellten Informationen anzuzeigen, gibt es die Möglichkeit, spezifische Bereiche (sogenannte *carved regions*) zu definieren, in denen Anzeigen gruppiert werden. Diese Bereiche werden ebenfalls dazu verwendet, um zu definieren, wer der beteiligten Nutzer mit der Darstellung in diesem Bereich interagieren darf. Dazu kann der erstellende Nutzer weitere Nutzer hinzufügen.

Eine Zuordnung von Information zu Displayflächen existiert nicht. Es wird auch kein spezifischer Layout Mechanismus angeboten, um die dargestellten Informationen anzuordnen. Eine Adaption der Informationsrepräsentationen oder eine spezifische Berücksichtigung des Nutzers bei der Darstellung sind ebenfalls nicht vorhanden.

Für die Interaktion der Nutzer mit den angezeigten Informationen stehen mehrere Mäuse und Tastaturen zur Verfügung. Damit wird es ermöglicht, dass mehrere Nutzer

simultan mit den Darstellungen interagieren können. Durch die bereits erwähnten *carved regions* wird zudem garantiert, dass andere Nutzer keine Manipulationen begehen können, die nicht erwünscht sind.

Zusammenfassend bietet Dynamo ein System, um ad-hoc bzw. durch eine *walk-up-and-use* Funktionalität mediale Inhalte gemeinsam mit anderen Nutzern anzuzeigen. Das Problem ist hier allerdings, dass die anzuzeigenden Informationen bereits auf einem mobilen Datenträger vorhanden sein müssen, wodurch die Anzeige wenig flexibel ist. Auch die Notwendigkeit, *Dynamo* eigene Applikationen für die Anzeige verwenden zu müssen, schränkt die Nutzbarkeit ein. Ist ein Datenformat dem System nicht bekannt, so ist eine Anzeige nicht möglich.

#### 2.2.2.3. Bildbasierte Informationsanzeige

**WeSpace**

Ein Vertreter der Bildbasierten Informationsanzeige ist der *WeSpace* [WJF+09]. Dieser ist ein System, um die Kollaboration zwischen Astrophysikern bei der Exploration und Visualisierung von Daten zu unterstützen. *WeSpace* wurde diesem konkreten Anwendungsziel folgend entwickelt und in mehreren Iterationen an die Bedürfnisse der Nutzer angepasst.

Der Raum besteht aus einem zentralen Tisch mit multitouch Display und einer großen Displayfläche vor diesem Tisch. An drei Seiten des Tisches sind Sitzplätze für die Nutzer.

Quellen der Informationsrepräsentationen sind persönliche Laptops der Nutzer. Dabei wird die vollständige Displayfläche des Laptops mit Hilfe eines modifizierten VNC (Virtual Network Computing ) Clients dem *WeSpace* Server zur Verfügung gestellt. Dieser Server stellt die Anzeigen auf der großen Displayfläche und dem touchtable gleichermaßen dar. Auf dem *WeSpace* Server können auch spezifische Applikationen gestartet werden, die dann ohne ein persönliches Gerät Informationsrepräsentationen erzeugen und darstellen. Der VNC Client ist sowohl MacOS X als auch Windows kompatibel.

Abbildung 2.7.: *WeSpace*. Bild aus [WJF+09]

Die ad-hoc Nutzung ist hier sehr gut unterstützt. Ist ein persönliches Gerät im Netzwerk angemeldet, kann durch das Starten des *WeSpace* Clients der Rechner für die Informationsanzeige genutzt werden.

Nach der Freigabe des Laptops kann der Nutzer die Wichtigkeit der Informationen festlegen. Diese kann *private* (soll nicht auf der großen Displayfläche angezeigt werden), *public* (soll auf der großen Displayfläche angezeigt werden) oder *important* (soll prominent auf der großen Displayfläche angezeigt werden) sein, was nicht nur die Auswahl der anzuzeigenden Informationen sondern auch das Layout der Anzeige bestimmt. Die als *important* markierten Informationen werden groß in der Mitte angezeigt, während die *public*-Anzeigen klein übereinander am Rand präsentiert werden. Eine gespiegelte Ansicht wird ebenfalls auf dem tabletop angezeigt. Zudem wurde das Layout erweitert, so dass auch Ansichten transparent über die Displayfläche verschoben werden können, um besser vergleichen zu können.

Die Interaktionen finden im *WeSpace* nur auf dem Touchtable statt und werden durch die gespiegelte Ansicht automatisch auf der großen Displayfläche angezeigt.

*WeSpace* wurde für die Kooperation kleiner Gruppen in einem bestimmten Anwendungsfeld entworfen. Es unterstützt sehr gut die strukturierte ad-hoc Anzeige von Informationen selbst von persönlichen Geräten. Auch die Nutzung von spezifischen nativen Applikationen wird ermöglicht. Allerdings wird das Arbeiten in diesem Umfeld mit zunehmender Anzahl der Teilnehmer schwierig, da der Winkel, mit dem die Nutzer auf die Informationen blicken, nicht berücksichtigt wird.

Im Folgenden werden Arbeiten vorgestellt, die einzelne Probleme der Informationspräsentation in Smart Environments adressieren.

### 2.2.3. Spezielle Ansätze

#### 2.2.3.1. Bereitstellen der anzuzeigenden Informationen

Für die Bereitstellung von Informationsrepräsentationen für große Displayflächen, für multiple Displayflächen oder für das Übertragen von Ansichten zu anderen Computern existieren eine Reihe weiterer Arbeiten. Hier ist ebenfalls die Unterteilung in (1) Ausnutzung von Betriebssystemspezifischen Eigenschaften, (2) Datenbasierte Informationsanzeige und (3) Bildbasierte Informationsanzeige zu beobachten.

*Betriebssystemspezifischen Informationspräsentation*

Bei der Betriebssystemspezifischen Informationsanzeige werden Eigenschaften des Betriebssystems bei der visuellen Ausgabe gezielt ausgenutzt. Dabei wird bei Linux das *XWindow* System [SG86] verwendet und das Rendering der Informationsrepräsentationen auf einen anderen Computer umgeleitet (z.B. [BKN05]). Bei Windows hingegen wird GDI (Graphics Device Interface) [GDI13] verwendet. Auch hier werden graphische Ausgaben zu anderen Computern versendet (z.B. [BBB+08, VDGR99]).

*Datenbasierte Informationspräsentation*

Bei der Datenbasierten Informationsanzeige werden die Informationsrepräsentationen auf dem jeweiligen Computer gerendert, an den die darstellende Displayfläche

angeschlossen ist. Die Informationsanzeige durch andere Geräte im Netzwerk basiert in der Regel darauf, dass Daten an den anzeigenden Computer verschickt werden und dort mit Hilfe einer passenden Applikation angezeigt wird. Neben dem bereits vorgestellten *iRoom* [JFW02] gibt es nur wenige Vertreter dieser Art der Informationsanzeige. Hier sind vor allem ARIS [BB04] und Gaia [RHC02b, RHC+02a] und EasyLiving [BMK+00] zu nennen.

*Bildbasierte Informationspräsentation*

Bei der Bildbasierten Informationsanzeige werden fertig gerenderte Bilddaten durch eine spezielle Applikation abgegriffen und über ein Netzwerk an einen anderen Computer versendet, wo die Bilddaten dann dargestellt werden. Hier ist vor allem die Nutzung von VNC Clients [RSFWH98] weit verbreitet. Neben OpenSource Projekten und kommerziellen Produkten zur Fernwartung (z.B. TightVNC [Tig13], UltraVNC [Ult13], RealVNC [Rea13], TeamViewer [Tea13]) werden VNC Clients auch für die Informationsanzeige in Multi-Display-Umgebungen eingesetzt [WL07, NSY+07, BFLA02, RSFWH98]).

Für die Auswahl von anzuzeigenden Informationen wird in der Regel versucht, die Absicht der Nutzer zu berechnen. Grundlegend werden dabei verschiedene Ansätze verfolgt um zu bestimmen, welche Informationsrepräsentationen bzw. welche Dokumente für die Nutzer von Interesse sind.

Ein Ansatz ist es, die Interessen der Nutzer implizit durch die Überwachung der Absichten zu erkennen [CLWB01]. Ein anderer Ansatz versucht, das Interesse an einem Dokument anhand eines Profil Managers zu errechnen [BBM+06]. Weiterhin wird auf Basis einer Agenda das Interesse an Dokumenten erkannt [BWB+11, GK07].

#### 2.2.3.2. Anzeige

Bei der Zuordnung von Informationsrepräsentationen zu einer Displayfläche gibt es grundsätzlich zwei Möglichkeiten: Interaktiv durch den Nutzer oder automatisch.

Bei der interaktiven Zuordnung wird in der Regel eine Applikation verwendet, mit deren Hilfe die Zuordnung vorgenommen werden kann. Dabei gibt es drei Arten, diese Applikationen zu gestalten: textuell, als Übersichtskarte oder ikonifiziert. Die dabei

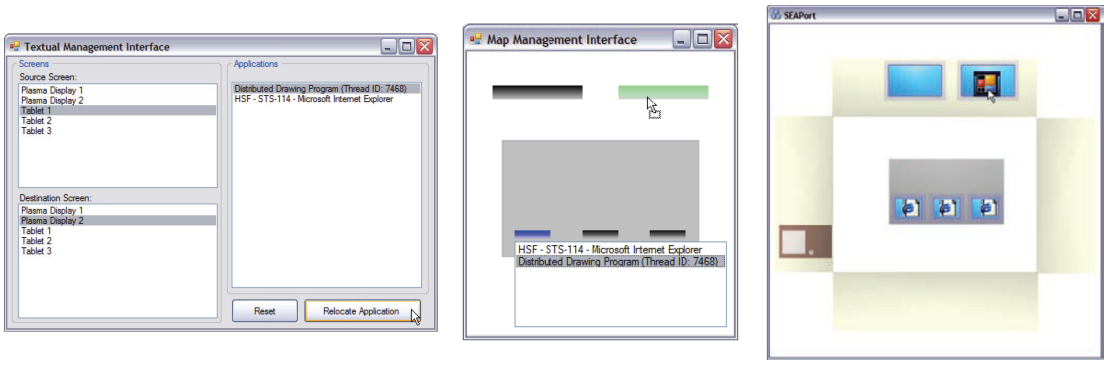



Abbildung 2.8.: Arten der Zuweisung von Informationsrepräsentationen zu Displayflächen, aus [BB06].

am weitesten verbreitete Art ist die ikonifizierte Darstellung wie z.B. in [BB04]. Diese ikonifizierte Darstellung (siehe Abbildung 2.8) wurde in einer Studie von Biehl et al. [BB06] mit den anderen Arten verglichen und als effektivste Art der Zuweisung evaluiert. Hier waren sowohl die Geschwindigkeit als auch die kognitive Belastung am geringsten.

Die automatische Zuordnung von Informationsrepräsentationen zu Displayflächen wurde in der Literatur bisher wenig betrachtet. Einen ersten Ansatz liefert der *Display Mapper* von Heider et al. [Hei09, HK08]. Durch eine Qualitätsfunktion wird die Zuordnung von einem Dokument zu einer Displayfläche berechnet. Die Qualitätsfunktion basiert dabei auf der Position und Blickrichtung der Nutzer, die Position und Ausrichtung der Displayflächen und dem Interesse eines Nutzers an den Dokumenten. Eine Einschränkung ist hier, dass der Suchraum für das optimale Mapping sehr schnell sehr groß wird. Zudem ist es nur möglich, ein Dokument pro Displayfläche anzuzeigen.

Für die Anordnung von Informationsrepräsentationen auf einer Displayfläche kann das allgemeine Layout Problem zugrunde gelegt werden. Das allgemeine Layout Problem ist das Bestimmen einer Anordnung einer beliebigen Menge von nicht überlappenden Rechtecken in einem minimalen rechteckigen Raum. Beach [Bea85] zeigte, dass dieses Problem NP-vollständig ist. Da damit kein effektiver Algorithmus zur Lösung dieses Problems existiert, ist die Herausforderung, Methoden zu entwickeln, die effektive Layouts entsprechend gewählter Evaluationskriterien generieren [Ali09].

Die Anordnung (Layout) von verschiedenen Informationsrepräsentationen auf einer Displayfläche wird in Abschnitt 3.6 genauer behandelt. Dort wird ein problemspezifischer Stand der Forschung vorgestellt.

### 2.2.4. Interaktion mit den dargestellten Informationen

Grundsätzlich wird in Smart Meeting Rooms der Ansatz verfolgt, eine Interaktion auf allen Displayflächen und mit allen Darstellungen zu ermöglichen. In der Regel werden die Maus- und Tastatureingaben per Netzwerk versendet und ausgeführt (z.B. [Wal11, WJF+09]). Dazu ist es meist notwendig, ein Modell der Anordnung der Displayflächen zu generieren, wodurch es ermöglicht wird, die räumliche Relation der Displayflächen zu bestimmen.

Ein weiterer Aspekt der Interaktion ist die direkte Manipulation der Informationsrepräsentationen. Es werden vor allem Themen wie die Annotation von Folien oder die Umsetzung eines digitalen Whiteboards thematisiert [HBR+09, HBL+06, KI07].

Der Stand der Forschung zur Interaktion in Smart Meeting Rooms wird in Kapitel 4.1 separat behandelt.

### 2.2.5. Assistenz für die Konfiguration der Anzeige

Im Rahmen von Smart Meeting Rooms werden Assistenzfunktionen eingesetzt, um den Nutzer bei der Konfiguration der Umgebung zu unterstützen. Im Rahmen der Informationsanzeige wird in der Regel versucht, den Nutzer bei der interaktiven Konfiguration mit intuitiven und leicht zu bedienenden Interfaces zu unterstützen [BB06].

Eine automatische Unterstützung für die Auswahl, Zuordnung und Anordnung von Informationsrepräsentationen wird dabei in der Literatur nicht sehr stark berücksichtigt.

Bei der Auswahl der anzuzeigenden Informationen existieren vor allem Ansätze, die auf Basis einer vorherigen Planung eine Assistenz während der Ausführung bereitstellen. Dabei fokussieren diese Arbeiten in der Regel auf die Unterstützung von Präsentationen. Hier kann nicht nur die zeitliche Abfolge der Vortragsfolien sondern

auch die räumliche Verteilung auf verschiedene Displayflächen vorher definiert werden [ZLL+04, KI07]. Es werden auch zusätzliche Funktionalitäten wie die Annotation von Vortragsfolien [CLB+03] oder die Verbesserung der Lesbarkeit [LKZH05] mit angeboten. Eine dynamische Auswahl während z.B. einer Präsentation wird hier allerdings nicht thematisiert.

Für eine automatische Zuordnung der Informationsrepräsentationen zu Displayflächen wurde der Display Mapper als erster Ansatz von Heider et al. [Hei09, HK08] vorgestellt.

Für die Anordnung gibt es eine ganze Reihe von Lösungen, mit deren Hilfe ein Layout verschiedenster Informationsrepräsentationen automatisch erstellt werden kann. Allerdings wird dies im Rahmen von großen Displayflächen in Smart Meeting Rooms oder Multi-Display-Umgebungen nur spärlich betrachtet. Einen ersten Ansatz liefert hier Waldner et al. [WSGS11, Wal11]. In Abschnitt 3.6 werden weitere Ansätze vorgestellt, die sich für solche Umgebungen übertragen lassen.

### 2.2.6. Zusammenfassung und konkretisierte Aufgabenstellung

Für die Informationsdarstellung in Multi-Display-Umgebungen gibt es drei grundlegende Klassen von Ansätzen: (1) Ausnutzung von Betriebssystemspezifischen Eigenschaften, (2) Datenbasierte Informationsanzeige und (3) Bildbasierte Informationsanzeige. Diese haben jeweils ihre spezifischen Vorteile bei der Informationspräsentation.

Bei der Ausnutzung von Betriebssystemspezifischen Eigenschaften ist eine höhere Flexibilität und Mächtigkeit der Möglichkeiten gegeben. Durch das Eingreifen in das Betriebssystem können z.B. auch verdeckte oder minimierte Inhalte angezeigt werden.

Die Betriebssystemspezifische Informationsdarstellung hat den Nachteil, dass die Einbindung von persönlichen und mobilen Geräten nur sehr schwer zu erreichen ist. In einem typischen Szenario wie es z.B. Encanação und Kirste [EaK05] beschreiben, in dem alle Nutzer ihren Laptop und ein Nutzer einen Projektor mitbringt, ist eine Betriebssystemspezifische Informationspräsentation nicht praktikabel. Typischerweise sind die Geräte der Nutzer heterogen, nicht nur bzgl. der Hardware sondern auch zunehmend bzgl. des verwendeten Betriebssystems. Eine Installation eines neuen Be-

triebssystems ist in diesem Fall nicht angebracht. Auch die Nutzung mobiler Geräte wird durch diese Form der Informationspräsentation eingeschränkt. Zum einen sind im Bereich der Smart Phones und Tablets die Unterschiede in den Betriebssystemen noch stärker, zum anderen lassen sich die Methoden, die für die Informationspräsentation ausgenutzt wurden, nicht zwangsläufig direkt übertragen.

Bei der Datenbasierten Informationsanzeige werden die jeweiligen Informationsrepräsentationen direkt für eine spezifische Displayfläche erzeugt. Die erzeugende Applikation ist direkt an die Displayfläche angebunden und kann so an die Displayfläche angepasste Darstellungen erzeugen.

Datenbasierte Informationsdarstellungen haben hingegen den Nachteil der Heterogenität der verwendeten Applikationen und Daten. Für eine Datenbasierte Informationsdarstellung ist die Anpassung der Applikation notwendig, mit deren Hilfe die Informationsrepräsentationen generiert werden sollen. Diese Applikationen müssen sowohl in die Lage versetzt werden Daten zur Darstellung durch eine andere Applikation zu senden, als auch Daten zu empfangen, um Darstellungen erzeugen zu können. Damit ist es nicht möglich, proprietäre Software zu verwenden. Auch müssen Applikationen zur Verfügung stehen, die verschiedene Daten verarbeiten können. Somit ist ebenso eine Anpassung bzgl. der Daten notwendig. Damit ist nicht nur die Anzahl der nutzbaren Applikationen eingeschränkt sondern es ist auch notwendig, im Vorfeld die richtige Version auf den Geräten der Nutzer zu installieren.

Die Bildbasierte Informationsdarstellung lässt sich dagegen für die Informationspräsentation im Smart Meeting Room einsetzen. Durch das Versenden von Bildinformationen wird eine Betriebssystemunabhängigkeit ermöglicht. Weiterhin kann dadurch auch die Nutzung von proprietärer Software gewährleistet werden. Zudem besteht eine hohe Flexibilität, die eine ad-hoc Nutzung möglich macht. Ein Eingriff in die Applikationen ist außerdem nicht notwendig.

Ansätze aus der Literatur, die eine Bildbasierte Informationsdarstellung thematisieren (wie z.B. [BFLA02, WJF+09, WL07], fokussieren allerdings hauptsächlich auf das zur Verfügung Stellen der technischen Basis. So werden Bildschirminhalte abgegriffen, verschickt und auf einer Displayfläche dargestellt. Eine weitere konzeptionelle Betrachtung der Informationsrepräsentation findet zumeist nicht statt. Somit werden

grundsätzliche Fragen wie z.B. das Bereitstellen der anzuzeigenden Informationen thematisiert, die automatische Assistenz für die Konfiguration der Anzeige der dargestellten Information werden hingegen nur sehr spärlich adressiert.

Auch beschränken sich die Ansätze aus der Literatur auf das Abgreifen der Bilddaten durch z.B. modifizierte VNC Clients. Eine weitergehende Integration der Applikationen über z.B. eine API zur Erzeugung der Bilddaten direkt durch die Applikation wird nicht thematisiert.

Ein weiterer Punkt ist, dass es kaum automatische Assistenz für die Nutzer bei der Informationsdarstellung gibt. Es wird zumeist eine technische Basis für die Interaktion mit den Darstellungen oder der Konfiguration der Anzeige geschaffen. Durch den Einsatz von ausgefeilten graphischen Interfaces wird dem Nutzer die manuelle Konfiguration dann erleichtert [BB06, BB04, PLF+01]. Eine automatische Assistenz, eingebettet in ein Gesamtkonzept für die Informationspräsentation findet sich allerdings nicht.

Auch eine Anpassung der Anzeige oder der Informationsrepräsentationen selbst wird nur wenig betrachtet. So steht die Ausnutzung der Informationen, die über den Nutzer zur Verfügung stehen (z.B. Position, Blickrichtung) erst am Anfang. Ein erster Ansatz findet sich dafür in [NSY+07]. Hier wird in einer Multi-Display-Umgebung basierend auf den Positionsdaten eines Nutzers eine perspektivische Entzerrung der Darstellungen vorgenommen. Ein weiterer Ansatz findet sich in [DKQ13]. Hier wird auf Basis der Blickrichtung eine spezifische Anpassung der Displayflächen vorgenommen, die im peripheren Sichtfeld liegen. Allerdings sind dies einzelne Ansätze, die nicht konzeptionell im Rahmen eines Gesamtsystems behandelt wurden.

Ziel der vorliegenden Dissertation ist es deshalb, ein Konzept für die Informationsanzeige in Smart Meeting Rooms zu entwickeln. Dieses Konzept soll dabei insbesondere das Bereitstellen der anzuzeigenden Informationen, die Anzeige der Informationsrepräsentationen, die Interaktion mit den Informationsrepräsentationen und die Anpassung sowohl der Anzeige als auch des Angezeigten adressieren.

# 3. Smart View Management

In diesem Kapitel wird das *Smart View Management* vorgestellt. Das *Smart View Management* ist ein Konzept zur Informationsdarstellung in Smart Meeting Rooms, in dem insbesondere die Bereitstellung der anzuzeigenden Informationen und die Anzeige der Informationsrepräsentationen thematisiert werden.

Im Abschnitt 3.1 werden zunächst die zu adressierenden Probleme beschrieben. Darauf basierend wird der prinzipielle Lösungsansatz in Abschnitt 3.2 vorgestellt. Dem folgt eine generelle Übersicht über die Architektur des *Smart View Managements*, wobei die funktionalen Komponenten genannt werden und gezeigt wird, wie diese Komponenten im *Smart View Management* zusammenarbeiten. In den darauf folgenden Abschnitten 3.3 bis 3.6 werden die einzelnen funktionalen Komponenten im Detail besprochen. In Abschnitt 3.7 wird dann eine konkrete Anwendung des *Smart View Managements* vorgestellt. Eine Zusammenfassung des Kapitels erfolgt in Abschnitt 3.8.

Das *Smart View Management* wurde auf der Smart Graphics publiziert [RLS11].

## 3.1. Problembeschreibung

Für die Informationspräsentation in Smart Meeting Rooms gilt es, zwei grundlegende Problemstellungen zu adressieren:

- **Bereitstellen** der Informationen.
- **Anzeigen** der Informationen.

Das **Bereitstellen** der Informationen bildet die Basis für die Informationspräsentation in Smart Meeting Rooms. Das grundlegende Problem besteht darin, die Infor-

mationsrepräsentationen für die Anzeige im Smart Meeting Room zur Verfügung zu stellen. Dafür lassen sich aus dem Stand der Forschung (vgl. Abschnitt 2.2) drei grundlegende Kategorien identifizieren: (a) Ausnutzung von Betriebssystemspezifischen Eigenschaften, (b) Datenbasierte Informationsanzeige und (c) Bildbasierte Informationsanzeige.

Eine Voraussetzung für die Informationspräsentation in Smart Meeting Rooms ist die ad-hoc Nutzung von privaten Geräten. Dafür ist es notwendig, Informationsrepräsentationen unabhängig vom Betriebssystem oder anderen Anforderungen an Hard- und Software an persönlichen Geräten zu erzeugen und diese für die Anzeige bereitzustellen. Aus diesem Grund wird in der vorliegenden Arbeit die Bildbasierte Informationsdarstellung verwendet, da sie nicht spezifische Software wie bei der Datenbasierten Informationsanzeige oder ein spezielles Betriebssystem wie bei der Betriebssystemspezifischen Informationsdarstellung voraussetzt (vgl. Abschnitt 2.2.6). Deshalb ist es notwendig, in Abschnitt 3.3 das Problem zu behandeln, wie Informationsrepräsentationen dem Smart Meeting Room zur Anzeige zur Verfügung gestellt werden.

Für die Übertragung der Informationsrepräsentationen ist es erforderlich, diese zu komprimieren. Die Komprimierung dient dem Ziel, Bandbreite im Netzwerk zu sparen und damit die Übertragungsgeschwindigkeit zu erhöhen bei gleichzeitigem Erhalt der Darstellungsqualität. Da im Rahmen dieser Arbeit die Bildbasierte Informationsdarstellung verwendet wird, ist es für die Übertragung notwendig, Bilddaten zu komprimieren. Zur Komprimierung von Bilddaten existieren verschiedenste Verfahren und Standards. Durch den Einsatz von Komprimierungsverfahren soll in der Bildbasierten Informationsdarstellung erreicht werden, dass Informationen bei geringstmöglicher Netzwerklast schnell angezeigt werden. Deshalb empfiehlt es sich, nicht nur die Kompressionsrate als Kriterium heranzuziehen, sondern ein Verfahren zu nutzen, dessen inhärente Eigenschaften diese Anforderung unterstützt. Darum wird im Abschnitt 3.3 ebenfalls das Problem der Kompression der Informationsrepräsentationen diskutiert.

Ein weiteres Problem beim Bereitstellen der Informationen ist die Beschreibung der bereitgestellten Informationsrepräsentationen. Diese Beschreibung stellt die Grundlage für die automatische Anzeige der Informationen dar. Um entscheiden zu können,

wie die Informationen angezeigt werden sollen, ist konkretes Wissen notwendig, um individuelle Eigenschaften bei der Anzeige berücksichtigen zu können, z.B. um zu entscheiden, welche Informationen gemeinsam angezeigt werden sollen. Aus diesem Grund wird in Abschnitt 3.3 das Problem der Beschreibung von Informationen diskutiert und in Abschnitt 3.4 das Zusammenfassen von Informationen als Grundlage für eine gemeinsame Anzeige beschrieben.

Die **Anzeige** von Informationen darin, eine Informationsrepräsentation auf einer Displayfläche auszugeben. In der klassischen Desktop Umgebung stellt dies kein Problem dar, da eine direkte Kopplung zwischen erzeugendem Gerät und Displayfläche besteht. Im Smart Meeting Room hingegen wird dies zu einem Problem.

In einer Multi-Display-Umgebung muss eine Displayfläche ausgewählt werden, auf dem die Informationen angezeigt werden sollen. In der bisherigen Literatur werden für die Auswahl der Displayflächen im Allgemeinen interaktive Methoden verwendet. Ein integraler Bestandteil von Smart Meeting Rooms ist es aber, die Nutzer bei ihrer Arbeit zu unterstützen [AEa06, C+98]. Dafür ist es wünschenswert, den Nutzern den Aufwand bei der Konfiguration abzunehmen und durch automatische Assistenz die manuelle Konfiguration zu ersetzen oder zumindest wesentlich zu erleichtern.

Das Problem, das es an dieser Stelle zu lösen gilt, ist eine automatische Zuordnung der Informationsrepräsentationen zu den verschiedenen Displayflächen zu erreichen. Die Herausforderung, auf Basis der Beschreibung der Informationen und der Eigenschaften von Raum (z.B. Positionen der Displayflächen, Größe der Displayflächen) und Nutzer (z.B. Position, Blickrichtung) eine automatische Zuordnung zu erreichen, wird in Abschnitt 3.5 thematisiert.

Eine weitere zu lösende Aufgabe bei der Anzeige ist, dass die Anzahl der anzuzeigenden Informationen im Smart Meeting Room nicht durch die Anzahl der vorhandenen Displayflächen beschränkt ist. Es kommt häufig vor, dass mehr Darstellungen angezeigt werden sollen, als Displayflächen zur Verfügung stehen. Darum ist es erforderlich, multiple Informationen auf einer Displayfläche anzuzeigen und diese außerdem automatisch auf den Displayflächen anzuordnen. Das ist auch der Fall, wenn multiple Informationen verschiedenster Applikationen dicht beieinander und damit auf einer Displayfläche angezeigt werden sollen, selbst wenn mehr Displayflächen zur

Verfügung stehen. So kann die Zusammengehörigkeit von Informationen berücksichtigt und damit z.B. das Vergleichen von Informationen erleichtert werden. In Abschnitt 3.6 wird deshalb diskutiert, wie verschiedene Informationen auf einer Displayfläche angezeigt werden und mittels automatischem Layout Mechanismus angeordnet werden können.

Für die Informationspräsentation müssen also zwei grundlegende Probleme gelöst werden: (1) Bereitstellen und (2) automatisches Anzeigen von Informationen. Dafür bedarf es geeigneter Strategien, um Informationsrepräsentationen zur Verfügung zu stellen, zu beschreiben und zusammenzufassen. Um eine automatische Anzeige der Informationen zu erreichen, werden Lösungen für die automatische Konfiguration der Anzeige, also der automatischen Zuweisung der Informationen zu den Displayflächen und der automatischen Anordnung multipler Informationen auf einer Displayfläche, benötigt.

## 3.2. Prinzipieller Lösungsansatz

Die grundsätzliche Lösungsidee für die Informationspräsentation in Smart Meeting Rooms ist, die visuellen Ausgaben von Applikationen, also die Informationsrepräsentationen, die von den jeweiligen Applikationen erzeugt werden, dynamisch zu verknüpfen und die Anzeige dieser Informationsrepräsentationen automatisch zu konfigurieren. Damit kann eine einheitliche Informationsanzeige verschiedenster Applikationen erreicht werden, ohne die Applikationen auf Daten- oder Funktionslevel verschneiden zu müssen.

Zur Beschreibung der visuellen Ausgaben, die von Applikationen erzeugt werden, soll im Rahmen dieser Arbeit der Begriff **View** definiert werden. Eine *View* bezeichnet dabei jegliche darstellbare Informationsrepräsentation (Text, Visualisierung, Bild, Präsentation, ...).

Um die Probleme der Informationspräsentation zu adressieren (vgl. Abschnitt 3.1), wurde das *Smart View Management* entwickelt. Durch das *Smart View Management* werden die vier grundsätzlichen Teilprobleme für die Informationspräsentation in Smart Meeting Rooms adressiert: (1) Generierung und Beschreibung von *Views* als Grund-

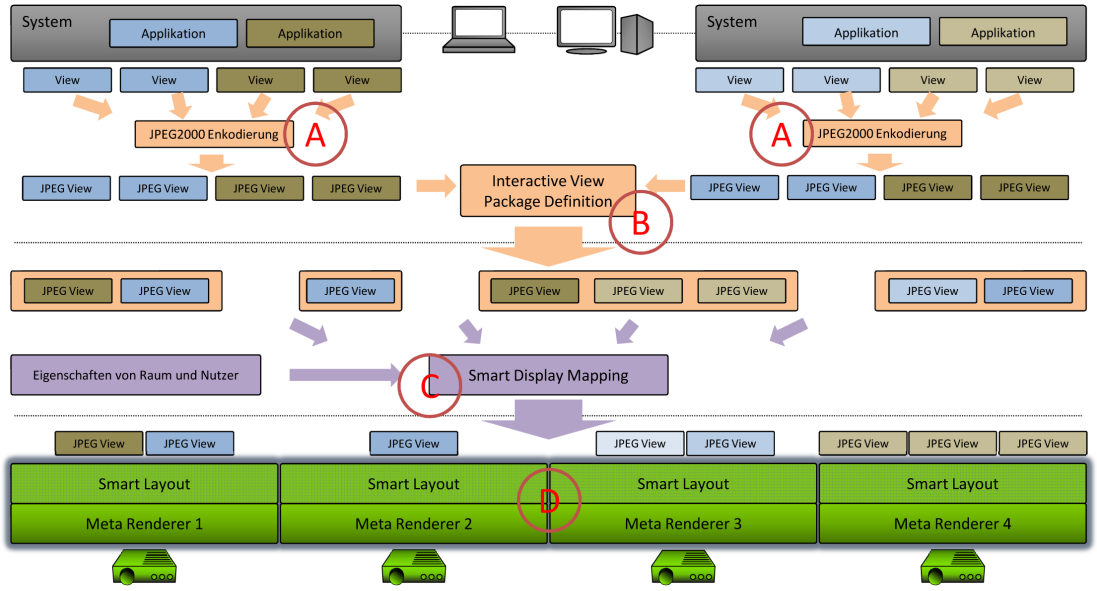



Abbildung 3.1.: Grundlegende Architektur des *Smart View Managements*. *Views* werden von multiplen Applikationen auf verschiedenen Geräten generiert. Durch die *Interaktive View Package Generierung* wird der Zusammenhang der *Views* mittels *View Packages* definiert. Auf Basis der *Views*, ihrer *View Description*, der *View Packages* und ihrer *View Package Description* und unter Zuhilfenahme von Informationen über die Nutzer (Position, Blickrichtung) sowie Informationen über den Raum (z.B. Position der Leinwände) werden die *Views* automatisch verschiedenen Displayflächen mittels *Smart Display Mapping* zugewiesen. Diese *Views* werden dann mittels *Metarenderer* und *Smart View Layout* automatisch auf den jeweiligen Displayflächen angeordnet und angezeigt.

lage für das Bereitstellen der Informationen, (2) Definition und Beschreibung der Zusammengehörigkeit von *Views*, (3) automatische Zuweisung von *Views* zu den Displayflächen und (4) automatisches Anordnen von *Views* auf einer Displayfläche.

Die grundlegende Architektur des *Smart View Managements* und damit der Zusammenhang bei der Adressierung der Teilprobleme ist in Abbildung 3.1 dargestellt. Im Folgenden werden die einzelnen Teilprobleme beschrieben und der Zusammenhang an der Architektur gezeigt.

**(1) Generierung von *Views*:** Hierzu gehören drei grundsätzliche Schritte:

1. Erzeugung der *View* interaktiv durch den Nutzer am persönlichen Gerät. Dafür werden sowohl Applikationen in die Lage versetzt, *Views* direkt zu generieren, als auch die Möglichkeit geschaffen, proprietäre Applikationen zur Erzeugung von *Views* zu nutzen.

2. Erstellen der *View Description* für die automatische Steuerung der Anzeige. Diese *View Descriptions* können sowohl automatisch durch die Applikationen abgeleitet als auch interaktiv durch den Nutzer festgelegt werden.

3. Kodierung der *Views* mit *JPEG2000*. Durch die automatische Kodierung der *Views* mit *JPEG2000* können zum einen die zu übertragenden Bilddaten reduziert und zum anderen eine flexible Dekodierung verschiedener Auflösungen ermöglicht werden.

Durch diese drei Schritte werden *Views* für die Anzeige bereitgestellt und durch *View Descriptions* so beschrieben, dass dies die Basis für die automatische Konfiguration der Anzeige liefert (siehe Abbildung 3.1 Markierung (A)). Die spezifische Auseinandersetzung mit dem Problem der *View* Generierung findet sich in Abschnitt 3.3.

**(2) Definition der Zusammengehörigkeit von *Views*:** Um die Zusammengehörigkeit von *Views* zu definieren, wird es dem Nutzer ermöglicht, *Views* interaktiv zu *View Packages* zusammenzufassen. *View Packages* repräsentieren eine Menge von *Views* und erlauben es, eine gemeinsame Anzeige dieser *Views* zu gewährleisten. Die View Packages werden durch *View Package Descriptions* beschrieben.

Die *View Packages*, die darin enthaltenen *Views* und die *View Package Descriptions* bilden die Basis für die automatische Anzeige der Informationen (siehe Abbildung 3.1 Markierung (B)). Dies wird in Abschnitt 3.4 genauer behandelt.

**(3) Zuweisung von *Views* zu den Displayflächen:** Dieser Schritt erfolgt automatisch, d.h. es wird automatisch bestimmt, auf welcher Displayfläche welche *Views* angezeigt werden sollen. Dazu wurde das sogenannte *Smart Display Mapping* entwickelt. Auf Basis der *Views*, ihrer jeweiligen *View Description*, der Zusammenfassung von *Views* zu *View Packages*, der jeweiligen *View Package Description*, sowie der Eigenschaften der Nutzer (Position, Blickrichtung) und des Smart Meeting Rooms (z.B. welcher Beamer ist aktiv, welche Leinwand steht zur Verfügung) wird eine automatische Zuordnung von *Views* zu Displayflächen vorgenommen (siehe Abbildung 3.1 Markierung (C)). Eine Ausführliche Betrachtung des *Display Mappings* findet sich in Abschnitt 3.5.

**(4) Anordnen von *Views* auf einer Displayfläche:** Zur Anzeige multipler *Views* auf einer Displayfläche ist ein Mechanismus erforderlich, mit dessen Hilfe *Views* von unterschiedlichen Quellen gemeinsam angezeigt werden können. Dafür wird ein sogenannter *Metarenderer* verwendet. Hier werden die Bilddaten der anzuzeigenden *Views* gesammelt und durch automatische Layoutmechanismen auf der Displayfläche angeordnet (siehe Abbildung 3.1 Markierung (D)). Die Anordnung von *Views* wird in Abschnitt 3.6 näher erläutert.

Im Folgenden (Abschnitte 3.3 bis 3.6) werden die Lösungen der Teilprobleme der Informationspräsentation in Smart Meeting Rooms behandelt, die im Rahmen des *Smart View Managements* entwickelt wurden. Jeder Abschnitt endet dabei mit einer Zusammenfassung der vorangegangenen Schritte. Die Zusammenfassung wird jeweils mit einem durchgängigen Beispiel illustriert (Abbildungen 3.2, 3.3, 3.5, 3.9 und 3.14).

## 3.3. View Generierung

Die Generierung von *Views* besteht aus drei grundsätzlichen Schritten: (1) Erzeugung der *Views* am persönlichen Gerät, (2) Kodierung der *Views* mit *JPEG2000* und (3) Erstellen von *View Descriptions* zum Beschreiben der *Views*.

Die **Erzeugung der Views** in Smart Meeting Rooms muss verschiedenste Anforderungen erfüllen. *Views* müssen ad-hoc auf privaten Geräten erzeugt werden können. Die Erzeugung der *Views* sollte betriebssystemunabhängig erfolgen, damit eine Vielzahl verschiedenster Geräte unterstützt werden kann. Die *View* Erzeugung sollte zudem weitestgehend unabhängig von Hard- und Software sein und somit auch proprietäre Applikationen mit einschließen.

Weiterhin ist es wünschenswert, *Views* erzeugen zu können, selbst wenn sie auf dem persönlichen Gerät minimiert oder verdeckt sind. Dadurch wird es ermöglicht, die private Displayfläche weiterhin zu nutzen, ohne Fläche für die Erzeugung von *Views* reservieren zu müssen. Das Arbeiten der Nutzer an den privaten Geräten würde so nicht beeinflusst werden.

Außerdem ist es von Vorteil, *Views* nativ für die Anzeige zu erzeugen. So kann z.B. die Auflösung einer *View* so angepasst werden, dass sie der Darstellung bei der Anzeige entspricht und keine Skalierung vorgenommen werden muss.

Um diese Anforderungen erfüllen zu können, wurden im Rahmen dieser Arbeit zwei Optionen für die Erzeugung von *Views* entwickelt: (1) die *ad-hoc View Erzeugung*, mit der eine unmittelbare Erzeugung von *Views* selbst mit proprietären Applikationen gewährleistet wird und (2) die *API View Erzeugung*, mit der die verdeckte Erzeugung von *Views* und die native Erzeugung bzgl. der Anforderungen der Anzeige realisiert werden können. Diese beiden Optionen werden im Folgenden genauer beschrieben.

**Ad-hoc View Erzeugung:** Durch die *ad-hoc View Erzeugung* werden *Views* auf Bildebene bereitgestellt, ohne Applikationen modifizieren zu müssen. Auf diese Art und Weise kann selbst proprietäre Software ohne Anpassung für die Generierung von *Views* verwendet werden. Dafür markiert der Nutzer mit Hilfe eines sogenannten *View Grabber* einen rechteckigen Ausschnitt auf der Bildschirmfläche eines persönlichen Gerätes. Dieser Bereich wird dann als *View* bereitgestellt, indem die Bilddaten dieses Ausschnitts der persönlichen Displayfläche als Bildstrom über das Netzwerk versendet werden. Von einem Gerät kann mittels *ad-hoc View Erzeugung* eine beliebige Anzahl verschiedener *Views* erzeugt werden. Diese Option entspricht der Bildbasierten Informationsanzeige, ermöglicht es allerdings, nur Ausschnitte der privaten Displayfläche als *Views* bereitzustellen

**API View Erzeugung:** Die *API View Erzeugung* benötigt eine Anpassung der informationsanzeigenden Applikation. Mittels einer bereitgestellten API (Schnittstelle zur Anwendungsprogrammierung) erzeugt die Applikation die Bilddaten der *View* selbstständig, ohne dass diese von einem *View Grabber* zusätzlich erfasst werden müssen. Damit wird die *View* Erzeugung direkt an die Generierung der Informationsrepräsentation durch eine Applikation gekoppelt.

Diese Art der *View* Erzeugung benötigt im Gegensatz zur *ad-hoc View Erzeugung* einen geringen zusätzlichen Implementationsaufwand. Die zur Verfügung gestellte API muss in der Applikation eingebunden werden. Dazu müssen lediglich die zur *View* gehörenden Bilddaten an eine vorher initialisierte Schnittstelle übermittelt werden. Die API bietet dann die Funktionalität, aus diesen automatischen Bilddaten eine *View* zu erzeugen. Das ermöglicht es der Applikation, eine *View* zu generieren, ohne diese zusätzlich auf der Displayfläche des privaten Gerätes anzeigen zu müssen. Weiterhin wird durch die Nutzung der API die Möglichkeit für eine Applikation geschaffen, multiple native Auflösungen einer *View* zu erzeugen und diese der Anzeige bereitzustellen.

Die *API View Erzeugung* ermöglicht die Erzeugung von *Views*, die nicht auf dem persönlichen Gerät angezeigt werden müssen. Somit kann mit einem persönlichen Gerät eine Vielzahl verschiedener *Views* von verschiedenen Applikationen erzeugt werden, ohne dass alle *Views* auch auf diesem Gerät angezeigt werden müssen. Der Nutzer kann dadurch mit seinem Gerät weiterarbeiten, ohne durch die *View* Erzeugung eingeschränkt zu werden. Darüber hinaus können durch die *API View Erzeugung Views* in multiplen Auflösungen erzeugt werden, die dann für die Anzeige dynamisch ausgewählt werden können.

Das Ergebnis beider Optionen der *View* Erzeugung ist ein *View*, der dem *Smart View Management* für die Anzeige zur Verfügung gestellt wird. Beide Optionen der *View* Erzeugung können dabei simultan verwendet werden, d.h. es lassen sich *Views* mit beiden Optionen simultan auf multiplen persönlichen Geräten generieren.

Im nächsten Schritt werden die erzeugten *Views* mit dem *JPEG2000* Standard komprimiert. Durch den Einsatz eines Komprimierungsverfahrens wird Bandbreite im Netzwerk gespart und damit die Übertragungsgeschwindigkeit erhöht.

Der *JPEG2000* Standard bietet die entscheidende Funktionalität, eine hierarchische Multilevel Repräsentation der *View* erzeugen zu können [Ros06] (siehe Abbildung 3.3a). Damit liefert *JPEG2000* nicht nur eine gute Kompression der Bilddaten, sondern bringt zusätzliche Vorteile für die Informationspräsentation: (1) eine flexible Skalierung (in 2er Potenzen), (2) eine schnelle Vorschau und (3) eine interaktive Verfeinerung von Bereichen von Interesse.

Der *JPEG2000* Standard ermöglicht eine flexible Dekodierung der Bilddaten. Dadurch lassen sich multiple Repräsentationen mit verschiedenen Auflösungen (in 2er Potenzen der ursprünglichen Auflösung) aus einer enkodierten *View* extrahieren, so dass die Größe der *Views* bei der Anordnung auf der Displayfläche entsprechend angepasst werden kann. Neben der guten Kompression, kann auch Bandbreite bei der Übertragung gespart werden. Für eine neue Auflösung muss die *View* nicht neu generiert, übertragen und angezeigt werden, sondern lässt sich aus der ursprünglich enkodierten *View* extrahieren.

Eine Enkodierung der *Views* mit *JPEG2000* erlaubt es weiterhin, *Views* progressiv zu übertragen. Dadurch kann eine Vorschau der *View* angezeigt werden, bevor die vollständigen Bilddaten übertragen wurden. Damit lässt sich eine schnelle Anzeige selbst bei sehr großen Bildern ermöglichen.

Die Nutzung von *JPEG2000* erlaubt darüber hinaus die interaktive Manipulation der *Views*. So kann z.B. ein spezifischer Bereich von Interesse selektiert und in höherer Auflösung angezeigt werden. Auch dazu ist es nicht notwendig, eine neue *View* zu erzeugen und zu übertragen.

Wegen dieser Möglichkeit der flexiblen Dekodierung eignet sich *JPEG2000* als Kompressionsstandard für den Einsatz im Rahmen des *Smart View Management*. Durch seine zusätzlichen Eigenschaften lassen sich Übertragungen von *Views* einsparen bzw. *Views* flexibel anzeigen. Darüber hinaus bereitet *JPEG2000* auch die Grundlage für interaktive Anpassungen der Anzeige (Abschnitt 4.3) und des Angezeigten (vgl. Abschnitt 4.4).

Das **Erstellen der View Descriptions** zur Beschreibung der *Views* ist der nächste Schritt der *View Generierung*. Mit Hilfe der *View Descriptions* werden die *Views* für die weitere Verwendung beschrieben. Eine *View Description* hat zwei wesentliche Funk-

tionen: (1) Bereitstellen von Informationen für die Konfiguration der Anzeige und (2) Unterstützen technischer Anforderungen der Anzeige.

Um diese beiden Funktionen der *View Description* zu unterstützen, wurden zwei Kategorien der Beschreibung definiert: (1) Informationen über Eigenschaften der *View* und (2) Informationen über die Erzeugung der *View*. Die Aspekte dieser Kategorien werden im Folgenden beschrieben.

**Eigenschaften:** Die Eigenschaften von *Views* werden für die spätere Anzeige verwendet, insbesondere die Anordnung der *Views* auf den Displayflächen (vgl. Abschnitt 3.6). Zur Unterstützung der Generierung eines automatischen Layouts wurden drei grundlegende Eigenschaften identifiziert:

**Quellauflösung** Die Quellauflösung ist die Auflösung einer *View*, in der sie erzeugt wurde. Bei der Anzeige entspricht diese Auflösung der bevorzugten Auflösung für die Darstellung.

**Zielauflösungen** Die Zielauflösungen sind eine Menge von Auflösungen, die aus der *JPEG2000* enkodierten *View* dekodiert werden können. Damit geben die Zielauflösungen die darstellbaren Auflösungen der *View* an.

**Seitenverhältnis** Das Seitenverhältnis gibt das Verhältnis von Breite zu Höhe einer *View* an.

Mit Hilfe dieser drei Eigenschaften lassen sich die möglichen Darstellungsformen einer *View* ableiten, die für eine automatische Anordnung erforderlich sind.

**Erzeugung:** Informationen über die Erzeugung der *View* sind Voraussetzung für die technische Umsetzung der Anzeige. So ist es für die Anzeige notwendig zu wissen, welches Gerät die *View* generiert hat und wie die *View* über das Netzwerk abgerufen werden kann. Diese Informationen sind auch für die Interaktion mit den *Views* notwendig (siehe Abschnitt 4). Folgende Informationen über die Erzeugung werden dafür verwendet:

**ID** Jeder *View* wird eine eindeutige Identifikationsnummer zugeordnet. Durch diese Nummer lässt sich eine *View* stets eindeutig identifizieren.

**Gerät** Hiermit wird die Netzwerkschnittstelle des *View* generierenden Geräts angegeben. Dazu gehören die IP (Internet Protokoll) Adresse und der Netzwerk Port, auf dem die *View* versendet wird.

**Option der Erzeugung** Die Option der Erzeugung gibt an, mit welcher Option der *View* Erzeugung (*ad-hoc View Erzeugung* oder *API View Erzeugung*) die *View* generiert wurde.

Diese Informationen über die Erzeugung der *View* werden verwendet, um stets auf die *View* technisch zugreifen zu können. Die *View Description* für alle *Views* besteht aus den beiden Informationen über Eigenschaften und Erzeugung und werden stets automatisch erstellt.

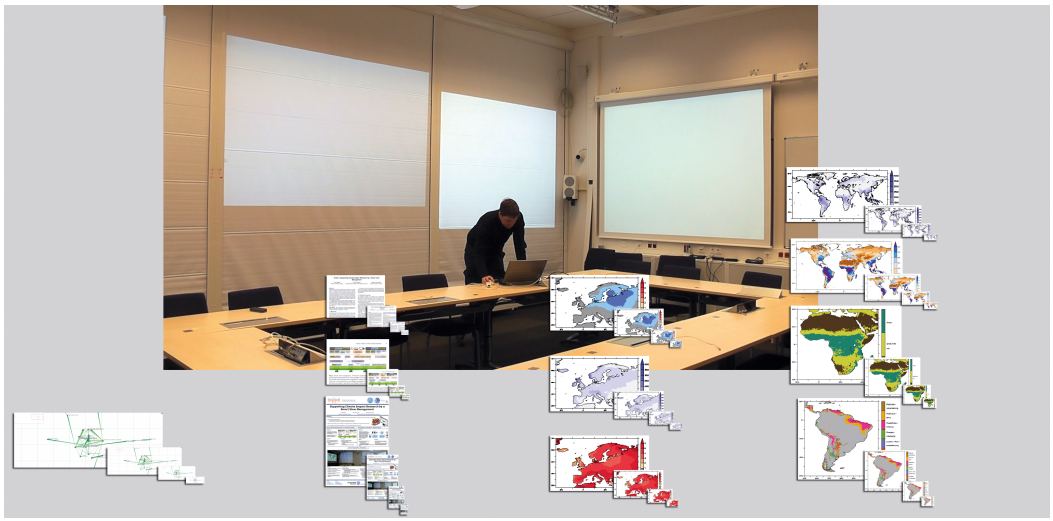
Darüber hinaus liefert die *API View Erzeugung* die Möglichkeit, das Generieren der Bilddaten und das Erzeugen der *Views* direkt miteinander zu koppeln. Diese Option erlaubt es, weitere Informationen den *View Descriptions* hinzuzufügen. Auf diese Art und Weise kann auch der Inhalt der *View* beschrieben werden. Im Rahmen der Visualisierung können somit z.B. der zugrunde liegende Datensatz, die angezeigten Attribute und die verwendete Visualisierungstechnik als Beschreibung mit angefügt werden.

Mittels *View Generierung* werden *Views*, erzeugt von verschiedenen Geräten und Applikationen, bereitgestellt (siehe Abbildung 3.2). Diese *Views* sind mit *JPEG2000* enkodiert und mittels *View Descriptions* beschrieben (siehe Abbildung 3.3), was zum einen eine schnelle Übertragung über das Netzwerk und zum anderen eine flexible Skalierung durch die Dekodierung ermöglicht.

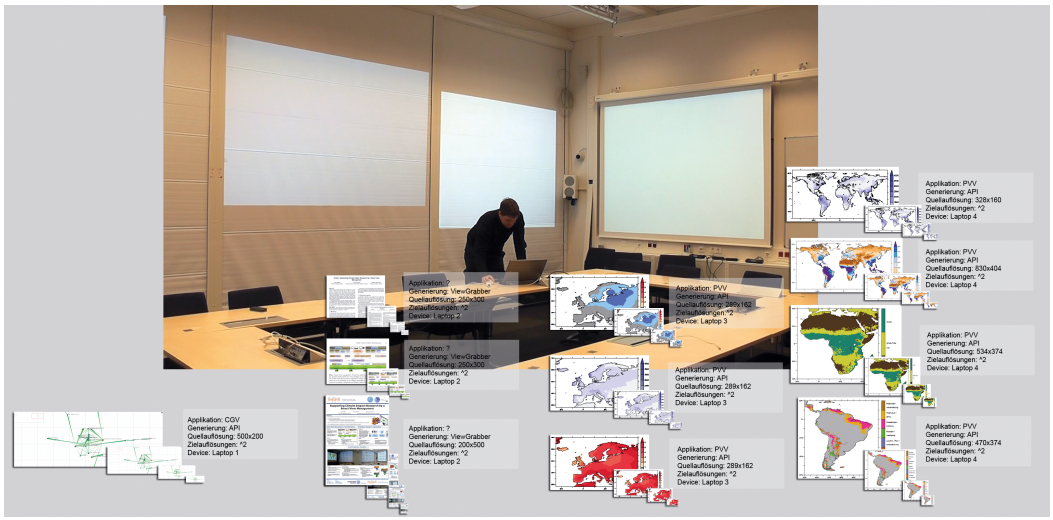
(a) Multiple Geräte mit multiplen Applikationen von verschiedenen Nutzern.

(b) Durch multiple Geräte, Applikationen und Nutzer generierte *Views*.

Abbildung 3.2.: Abbildung (a) zeigt eine Ausgangssituation im Smart Meeting Room. Multiple Geräte mit multiplen Applikationen von verschiedenen Nutzern sollen zur Bereitstellung von *Views* verwendet werden. Abbildung (b) zeigt die Erzeugung der *Views* von multiplen Geräten und Applikationen.



(a) *JPEG2000* kodierte *Views* mit hierarchischer Multilevel Repräsentation.




(b) Durch *View Descriptions* angereicherte Views.

Abbildung 3.3.: Abbildung (a) zeigt die Erzeugung der hierarchischen Multilevel Beschreibung durch die Kodierung mit *JPEG2000* und Abbildung (b), die Anreicherung der *Views* mit *View Descriptions*.

## 3.4. View Package Generierung

*View Packages* beschreiben eine Menge von zusammengehörenden *Views*. Das Wissen über diese Zusammengehörigkeit und die Beschreibung der *View Packages* mittels *View Package Descriptions* bilden die Grundlage für die automatische Anzeige der *Views*, spezifisch die Zuweisung der *Views* zu den Displayflächen (siehe Abschnitt 3.5).

Für die Beschreibung der Zusammengehörigkeit von *Views* wäre eine automatisierte Generierung von *View Packages* und *View Package Descriptions* wünschenswert. Entsprechende Einflussfaktoren können aktuell allerdings nicht in Echtzeit durch ein automatisches System berücksichtigt werden. Die automatische Generierung ist etwa abhängig vom Szenario, von der Anwendung und nicht zuletzt von den darzustellenden Informationen. Diese Faktoren lassen sich alle im Vornherein nicht immer so beschreiben, dass eine automatische Auswertung erfolgen kann. Deshalb wird im Rahmen dieser Arbeit ein interaktiver Ansatz genutzt, der dem Nutzer die volle Kontrolle über die Generierung von *View Packages* und der *View Package Descriptions* gewährt.

Der im Rahmen des *Smart View Managements* verwendete interaktive Ansatz bietet dem Nutzer ein graphisches Interface, welches es erlaubt, die *View Packages* zusammenzustellen (siehe Abbildung 3.4). Dieses Interface zeigt auf der linken Seite eine Vorschau auf die generierten *Views* (Abbildung 3.4 (A)). Diese *Views* können dann per drag-and-drop Interaktion zu den rechts dargestellten *View Packages* hinzugefügt oder neue *View Packages* können erzeugt werden (Abbildung 3.4 (B)). Weiterhin können die aktuellen *Views* und *View Packages* geladen und die aktuell zusammengestellte Konfiguration angewendet werden (Abbildung 3.4 (C)).

Zur Beschreibung der *View Packages* werden *View Package Descriptions* verwendet. Basierend auf den Anforderungen der Anzeige werden die *View Package Descriptions* in zwei Kategorien geteilt. Die erste Anforderung ist, dass der Ort der Anzeige den Bedürfnissen der Nutzer in ihrem aktuellen Szenario entspricht. Die zweite Anforderung ist, in der Anzeige die Aufgaben zu unterstützen, die mit den *Views* des *View Packages* erledigt werden sollen. Daher ergeben sich die beiden Kategorien: (1) Szenario und (2) Aufgabe. Diese Kategorien erheben nicht den Anspruch auf Vollständigkeit, spiegeln aber die typischen Aspekte eines Smart Meeting Rooms wider.

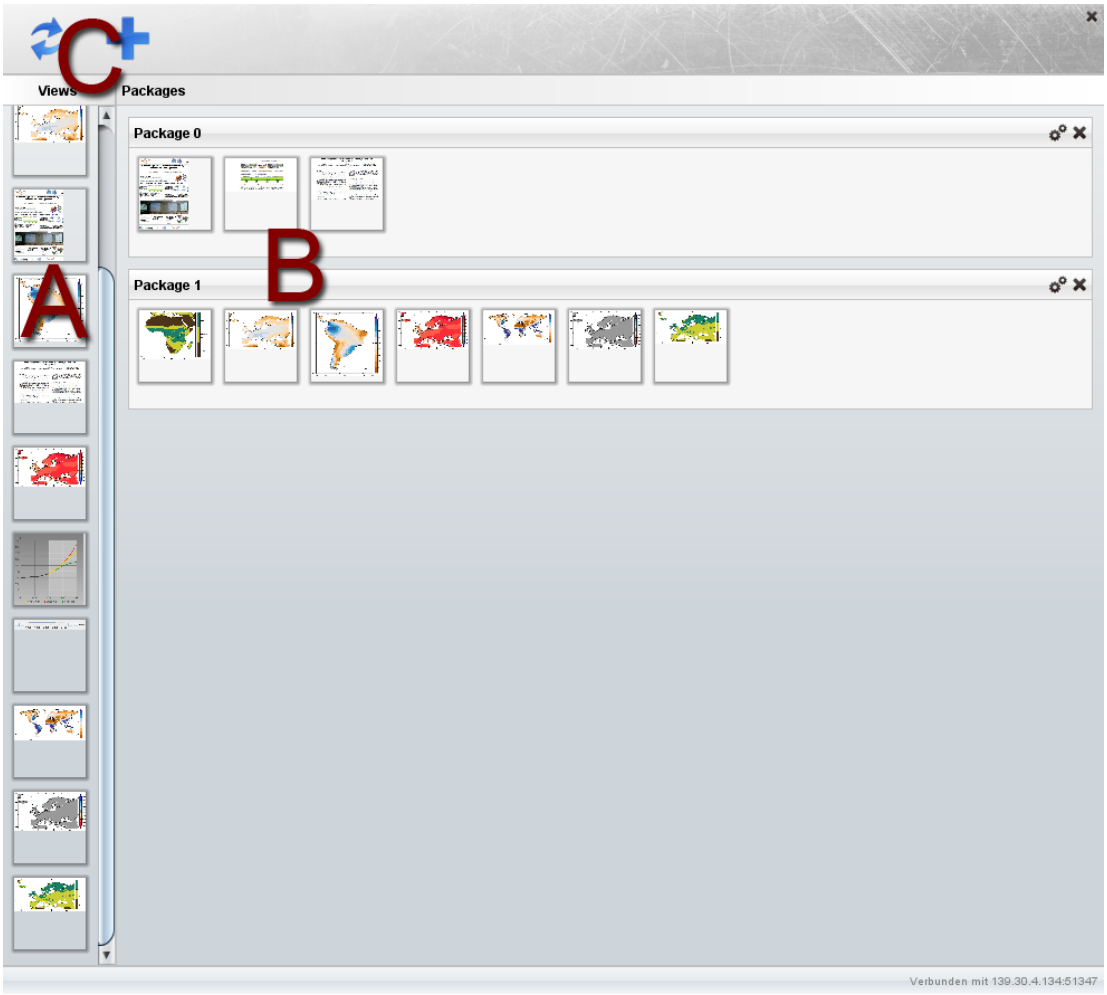



Abbildung 3.4.: Das grafische User Interface (GUI) für die Erzeugung der *View Packages*. Links (Markierung A) werden die zur Verfügung stehenden *Views* als Miniatur angezeigt. Auf der rechten Seite (Markierung B) werden die *View Packages* angezeigt. Diese können mittels drag-and-drop Interaktion erstellt und manipuliert werden. Mittels Schaltflächen an der oberen rechten Ecke jedes *View Packages* lassen sich die *View Package Descriptions* bearbeiten bzw. *View Packages* löschen. Oben (Markierung C) kann die Ansicht der *Views* und der *View Packages* aktualisiert bzw. angewendet werden.

**Szenario** (Präsentation, Diskussion) Das Szenario, in dem Nutzer eines Smart Meeting Rooms arbeiten, ist wichtig für die Konfiguration der Anzeige. Grundsätzlich lassen sich hier zwei verschiedene Arten des Arbeitens unterscheiden. Einerseits kann ein Nutzer im Mittelpunkt des Interesses stehen. Dieser Nutzer vermittelt als Presenter Informationen oder leitet einen Erkenntnis- oder Meinungsaustausch. Dieses Szenario lässt sich mit dem Begriff *Präsentation* beschreiben. Andererseits kann es sein, dass kein spezifischer Nutzer im Mittelpunkt steht, sondern ein gleichberechtigter Austausch stattfindet. Dieses Szenario lässt sich mit dem Begriff *Diskussion* beschreiben. Das entsprechende Szenario hat dabei jeweils Anforderungen an die Anzeige von *Views*. Bei einer Präsentation ist es wichtig, dass die *Views* dicht beim Presenter angezeigt werden und nicht beliebig im Raum verteilt sind. Bei der Diskussion hingegen ist es lediglich notwendig, dass alle *Views* für alle Nutzer gut sichtbar sind.

**Aufgabe** (präsentieren, vergleichen) Die Aufgabe, für die ein *View Package* verwendet wird, ist ebenfalls wichtig für die Anzeige. Es ist vor allem entscheidend, ob die *Views* eines *View Packages* einzeln genutzt werden sollen, um z.B. relevante Informationen aus der Darstellung zu extrahieren, oder ob *Views* gemeinsam genutzt werden sollen, z.B. beim Vergleichen der Inhalte von *Views*. Bei der einzelnen Nutzung wird an die Anzeige lediglich die Anforderung gestellt, die *View* auf einer Displayfläche darzustellen. Diese Aufgabe wird als *Präsentieren* bezeichnet. Bei der Aufgabe des *Vergleichens* hingegen, wird durch die inhaltliche Nähe dieser *Views* auch eine räumliche Nähe bei der Anzeige gefordert, da dies den Vergleich erheblich erleichtert.

Das Resultat nach der *View Package* Erstellung ist die vollständige Bereitstellung der *Views* für die Anzeige (siehe Abbildung 3.5). Die *Views* wurden durch *API View Erzeugung* von Applikationen oder durch die *ad-hoc View Erzeugung* mittels *View Grabber* erzeugt und durch *View Descriptions* beschrieben. Die Zusammengehörigkeit von *Views* wurde durch *View Packages* und den dazugehörenden *View Package Descriptions* beschrieben. Darauf aufbauend wird nun in den folgenden Abschnitten die automatische Anzeige thematisiert.

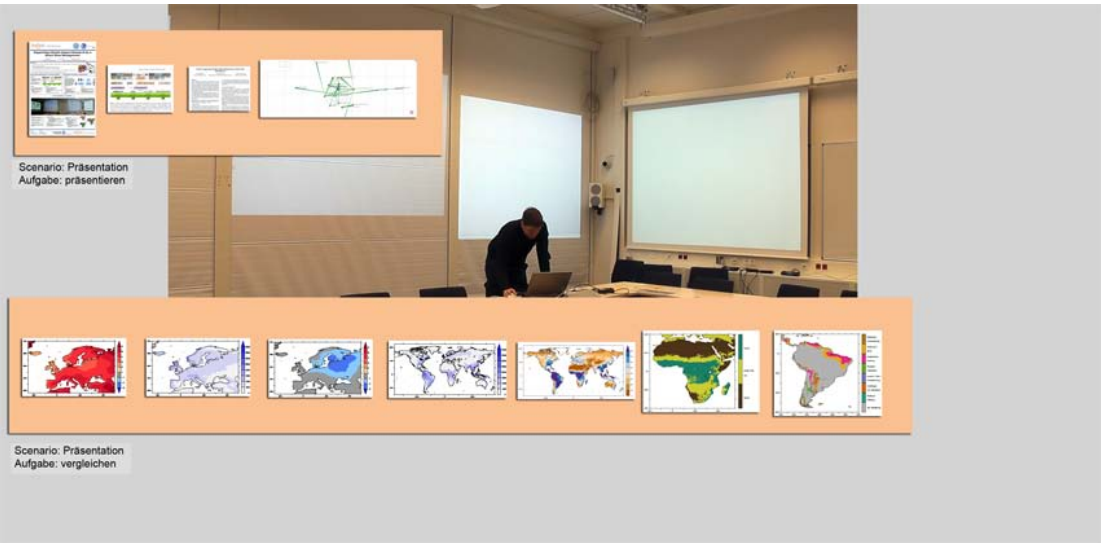



Abbildung 3.5.: Diese Abbildung zeigt die Gruppierung der vorher generierten *Views* (siehe Abbildungen 3.2 und 3.3) in *View Packages* und die Definition der *View Package Descriptions*. Die *View Packages* sind dabei in dem Szenario Präsentation. Das obere *View Package* ist für das Präsentieren konfiguriert, während das untere *View Package* für den Vergleich konfiguriert wurde.

## 3.5. Smart Display Mapping

Der erste Schritt bei der Anzeige der *Views* ist die Zuweisung der *Views* zu den vorhandenen Displayflächen. In der gängigen Literatur wird dies in der Regel durch die interaktive Zuweisung mittels einer graphischen Nutzerschnittstelle realisiert (vgl. Kapitel 2.2.3.2). Eine automatische Zuordnung der *Views* zu den Displayflächen ermöglicht es hingegen, den Nutzer zu unterstützen, indem ihm die Arbeit für die Konfiguration abgenommen wird.

Die Problemstellung ist nun, auf Basis des Wissens über die anzuzeigenden *Views*, über die Nutzer und den Smart Meeting Room eine Zuweisung der *Views* zu den Displayflächen zu finden, um die Arbeit der Nutzer zu unterstützen.

Um dieses Problem zu adressieren, wurde das *Smart Display Mapping* entwickelt. Es basiert auf dem in [HK08] und [Hei09] vorgeschlagenen *Display Mapping*, erweitert dieses jedoch durch entscheidende Punkte in Bezug auf (1) die Nutzung multipler *Views* auf einer Displayfläche, (2) die Umsetzung der *View Package Description* für die Konfiguration der Anzeige und (3) die Qualitätsberechnung eines *Display Mappings*. In den folgenden Abschnitten wird zuerst der grundlegende Ansatz von Heider et al. vorgestellt. Anschließend werden die im Rahmen dieser Arbeit entwickelten Erweiterungen beschrieben, die teilweise auf Arbeiten von Martin Luboschik basieren.

### 3.5.1. Display Mapping - grundlegender Ansatz

Heider definiert das *Display Mapping* in [Hei09] als die Zuordnung von Dokumenten zu Displayflächen. Seien dabei $D$ und $Y$ die Mengen von Dokumenten und Displayflächen, dann ist ein *Display Mapping* $m$ definiert als Funktion $m : D \rightarrow \mathcal{P}(Y)$. Diese Funktion weist jeweils ein Dokument der Menge $D$ einer Menge von Displayflächen $\mathcal{P}(Y)$ zu. Das heißt, für ein gegebenes Dokument $d \in D$ ergibt $m(d) \in \mathcal{P}(Y)$ die Menge von Displayflächen, denen $d$ zugeordnet ist.

Hier ist allerdings zu beachten, dass beim Ansatz von Heider zwar ein Dokument multiplen Displayflächen zugeordnet werden kann, die Zuordnung von multiplen Dokumenten zu nur einer Displayfläche jedoch nicht möglich ist.

Das Finden eines Mappings mit einer hohen "Qualität" kann nun als Optimierungsproblem aufgefasst werden. Die Qualität eines Mappings $m$ bei gegebenem vorherigen Mapping $m_0$ wird dann mit Hilfe der Funktion $q(m, m_0)$ berechnet:

$$q(m, m_0) = \alpha q_s(m) + \beta q_t(m, m_0) + \gamma q_p(m) \tag{3.1}$$

Mit Hilfe der von Heider vorgeschlagenen Mappingfunktion wird die Gesamtqualität des *Display Mappings* mittels Linearkombination dreier heuristischer Qualitätsmaße berechnet: der räumlichen Qualität ($q_s$), der Qualität der zeitlichen Kontinuität ($q_t$) und der Qualität der semantischen Nähe ($q_p$). Die verschiedenen Qualitätsmaße sind dabei gewichtet, um den Einfluss der drei Komponenten zu balancieren ($\alpha, \beta, \gamma \epsilon [0..1]$).

Hierbei bezeichnet die *räumliche Qualität* ($q_s$) die Sichtbarkeit der angezeigten Dokumente für alle Nutzer. Die *zeitliche Kontinuität* ($q_t$) wird verwendet, um keine schnellen Änderungen in den Mappings zu erhalten, denn ein Wechsel der Mappings in sehr kurzen Abständen mit allerdings marginalen Qualitätsverbesserungen ist für die Nutzer sehr verwirrend. Die *Qualität der semantischen Nähe* ($q_p$) bezeichnet dabei, dass Dokumente, die inhaltlich zusammen gehören, auch in räumlicher Nähe zueinander angezeigt werden sollten. Heider hat in seinen Arbeiten dafür keine explizite Lösung erarbeitet und nutzt die semantische Nähe als Platzhalter für eine spätere Umsetzung.

Heider schlägt für die Gewichte folgende Werte vor: $\alpha = 1$ und $\beta = 0,1$. Für $\gamma$ wird kein Wert vorgeschlagen, da die semantischen Nähe ($q_p$) der Dokumente nicht bekannt war.

Diese Qualitätsfunktion gilt es im *Display Mapping* zu maximieren. Heider zeigt in [Hei09], dass es sich hierbei um ein NP-hartes Problem handelt und nutzt daher einen verteilten Greedy-Algorithmus, um zufällige Mappings zu erzeugen. Diese werden dann mit Hilfe der Qualitätsfunktion bewertet, um das Optimum zu finden.

### 3.5.2. Erweiterungen

Im Folgenden wird die Anzeige von Dokumenten verallgemeinert und auf die Anzeige von *Views* bezogen, weshalb folgend die Bezeichnung *Views* anstelle von Dokument

verwendet wird. Dies gewährleistet nicht nur eine konstante Bezeichnung innerhalb der gesamten Arbeit, sondern verdeutlicht auch die Zunahme der Dynamik bei der Anzeige von Informationen. Während im ursprünglichen Ansatz von Heider der Begriff Dokument verwendet wird, stellt eine *View* ein sehr viel umfangreicheres Konzept dar. *Views* symbolisieren nicht nur statische, vorberechnete Ansichten, sondern schließen auch dynamisch erzeugten Inhalt mit ein.

Im Rahmen des *Smart View Managements* wurden drei grundsätzliche Erweiterungen des ursprünglichen *Display Mappings* vorgenommen:

1. Realisierung multipler *Views* auf einer Displayfläche.
2. Umsetzung des semantischen Zusammenhangs durch Berücksichtigung der *View Packages* und deren *View Package Descriptions*.
3. Verbesserte Qualitätsberechnung bei der räumlichen Qualität und zeitlichen Kontinuität.

Diese drei Erweiterungen werden nun genauer beleuchtet.

#### 3.5.2.1. Multiple Views auf einer Displayfläche

Der *Display Mapper* von Heider et al. ist lediglich in der Lage, einer Displayfläche eine einzelne *View* zuzuordnen. Dies ist eine starke Einschränkung in einem Smart Meeting Room Szenario. Die Anzahl der darstellbaren *Views* würde demnach von den Displayflächen der Umgebung abhängen und sich nicht aus der Notwendigkeit einer Anwendung ergeben. Deshalb ist es erforderlich, den Ansatz so zu erweitern, dass auch multiple *Views* einer Displayfläche zugeordnet werden können.

Dies wird durch die Teilung der physikalischen Displayflächen in logische Displayflächen erreicht. Sei $m$ die Anzahl der anzuzeigenden *Views*, dann wird jede Displayfläche in $n$ logische Displayflächen geteilt. Damit kann die Eigenschaft erhalten bleiben, dass einer (jetzt logischen) Displayfläche nur eine *View* zugeordnet werden kann. Hierbei muss lediglich eine redundante Anzeige identischer *Views* auf einer physischen Displayfläche ausgeschlossen werden. Ein Mapping mit dieser Eigenschaft wird grundsätzlich nicht weiter betrachtet.

#### 3.5.2.2. Umsetzung des semantischen Zusammenhangs

Bei der Anzeige multipler *Views* von multiplen Nutzern und Applikationen auf verschiedenen Displayflächen ist es notwendig, zusammengehörende *Views* in räumlicher Nähe zueinander anzuzeigen, da sonst ihre Verbindung untereinander verloren geht. Sind außerdem diese *Views* im Raum weit voneinander entfernt, ist es schwierig, mit diesen *Views* zu arbeiten. Daher ist es für das *Smart View Management* von Bedeutung, den Zusammenhang der *Views* beim *Display Mapping* zu berücksichtigen. Aber auch die Eigenschaften der *Views* sind beim *Display Mapping* von Bedeutung.

In [Hei09] wird der semantische Zusammenhang der Dokumente für die Berechnung der Qualität eines *Display Mappings* durch die Funktion $q_p$ definiert. Da der Zusammenhang der Dokumente in [Hei09] nicht bekannt ist, wird keine konkrete Umsetzung angegeben. Dafür wurde eine allgemeine Definition für den semantischen Zusammenhang zwischen Dokumenten eines *Display Mappings* vorgeschlagen:

$$q_p(m) = -\sum_{\substack{u \in U \\ d,d' \in D}} \rho(d,d') \cdot \delta(\pi(m,u,d), \pi(m,u,d')) \cdot input(d,u) \cdot input(d',u) \qquad (3.2)$$

Diese allgemeine Definition basiert auf zwei Funktionen: dem semantischen Zusammenhang zwischen zwei Dokumenten $\rho(d,d')$ und dem physikalischen Abstand der Displayflächen, die diese beiden Dokumente anzeigen $\delta(y,y')$, wobei $\pi(m,u,d)$ bzw. $\pi(m,u,d')$ die physikalische Position der anzeigenden Displayfläche im Raum darstellt bei aktuellem Mapping $m$, Nutzer $u$ und Dokument $d$ bzw. $d'$. Für die Ausgestaltung des physikalischen Abstandes empfiehlt Heider z.B. die Nutzung der euklidischen Distanz.

Weiterhin gibt Heider die Platzhalterfunktion input(d,u) bzw. input(d',u) für eventuelle zusätzliche Informationen über die Dokumente an, wie z.B. eine Agenda einer Präsentation.

Da weiterhin die Summe für die gesamte Qualitätsfunktion $(q(m))$ einen Malus darstellt, wird sie negiert.

Die Zusammenhänge zwischen den *Views* sind im *Smart View Management* durch die *View Packages* gegeben. Dadurch kann die allgemeine Definition von Heider nun konkret umgesetzt werden. Sei $V$ die Menge aller *Views*, $v, v' \in V$, $P$ die Menge aller *View Packages* und $p \in P$ ein *View Package*, dann lässt sich $\rho(v, v')$ nun sehr einfach definieren:

$$\rho(v, v') := \begin{cases} 1 & \text{,wenn } v \in p \wedge v' \in p \\ 0 & \text{,sonst} \end{cases} \tag{3.3}$$

Für den Abstand der *Views* wird die euklidische Distanz verwendet.

Durch diese Konkretisierung des semantischen Zusammenhangs lässt sich die inhaltliche Nähe der *Views* in *View Packages* in eine räumliche Nähe bei der Anzeige umsetzen. Dabei wird allerdings die Art der Zusammengehörigkeit bisher noch nicht betrachtet.

Im Folgenden wird daher die Anwendung der *View Package Description* zur Beschreibung der *View Packages* erläutert. Durch die *Aufgabe* wird die Zusammengehörigkeit der *Views* quantifiziert und durch das *Szenario* wird der Ort der Anzeige manipuliert. Der Aspekt der *Skalierung* spielt beim *Display Mapping* keine Rolle. Dieser kommt beim Layout zum Tragen, wo die Anordnung und damit auch Skalierung thematisiert wird.

**Umsetzung der kategorisierten Beschreibung von View Packages**

Der Aspekt der Aufgabe ist bereits in der Qualitätsfunktion des *Display Mappings* durch die Definition des semantischen Zusammenhangs der *Views* eingeflossen. Folgend werden die Umsetzungen der Aspekte *Aufgabe* und *Szenario* beschrieben.

**Aufgabe** Wie bereits dargestellt, wird im *Smart View Management* die semantische Nähe durch die Zusammengehörigkeit von *Views* in *View Packages* in physikalische Nähe umgesetzt. Um eine differenzierte Betrachtung dieser Zusammengehörigkeit zu erreichen, wird der *View Package* beschreibende Aspekt *Aufgabe* verwendet. Es werden zwei Arten von Aufgaben betrachtet: *Präsentieren* und *Vergleichen* (vgl. Abschnitt 3.4).

Dabei stellt *Vergleichen* die Forderung nach größerer räumlicher Nähe als *Präsentieren*. Sei $V$ wiederum die Menge aller *Views*, $v, v' \in V$, $P$ die Menge aller *View Packages* und $p \in P$ ein *View Package*. Sei weiterhin $A$ die Menge aller Aufgaben ( $A = \{Vergleichen, Prsentieren\}$) mit $a \in A$, dann wird zur Umsetzung dieses Aspektes die Funktion der semantischen Nähe $\rho(v, v')$ wie folgt erweitert:

$$\rho(d, d', a) := \begin{cases} 1 & \text{,wenn } v \in p \wedge v' \in p \wedge a = Vergleichen \\ 0,5 & \text{,wenn } v \in p \wedge v' \in p \wedge a = Prsentieren \\ 0 & \text{,sonst} \end{cases} \tag{3.4}$$

Durch die Erweiterung der Funktion der semantischen Nähe durch den Aspekt der *Aufgabe* kann bei der Zuweisung der *Views* zu den Displayflächen die mit den *Views* eines *View Packages* zu erfüllende Aufgabe mit einbezogen werden.

**Szenario** Die Szenarien *Präsentation* und *Diskussion* werden ebenfalls bei der Berechnung der Qualität des Display Mappings berücksichtigt.

Bei der *Präsentation* übt ein Nutzer stets die Rolle des Presenters aus. Das bedeutet, dass im Allgemeinen alle anderen Nutzer zum Presenter schauen. Für das Display Mapping heißt dies, dass die *Views* eines *View Packages* mit dem Aspekt der *Präsentation* auch in der Nähe des Presenters angezeigt werden sollen. Ziel bei der Umsetzung der *Präsentation* ist es nun, dass die Qualität eines Mappings dann am größten ist, wenn die *Views* auf den Displayflächen neben dem Presenter angezeigt werden.

Um dies zu erreichen, muss keine Funktion des *Display Mappings* modifiziert oder ausgetauscht werden. Vielmehr wird die *räumlichen Qualität* ausgenutzt. Es werden die Position und die Blickrichtung der Nutzer für die Berechnung verwendet, wie gut eine *View* für einen Nutzer sichtbar ist. Dabei wird die Blickrichtung der jeweiligen Nutzer durch Sensoren im Raum ermittelt. Zur Umsetzung der *Präsentation* werden die Blickrichtungen der Nutzer im Auditorium verändert. Zunächst wird der jeweilige Vektor zwischen der Position des Nutzers und des Presenters gebildet. Dieser Vektor wird dann genutzt, um die Blickrichtung des jeweiligen Nutzers zu approximieren (siehe Abbildung 3.6).

Abbildung 3.6c zeigt deutlich, dass die räumliche Qualität der *Views* automatisch an den Displayflächen am größten ist, die sich in unmittelbarer Nähe des Presenters befinden.

Im Szenario der *Diskussion* sollte die Sichtbarkeit der *Views* für alle Nutzer gleich sein. Daher wird die real gemessene Blickrichtung der Nutzer für die Berechnung der räumlichen Qualität genutzt (siehe Abbildung 3.6d).

Die semantischen Nähe ($q_p$) im *Smart Display Mapping* wird mit dem Wert $\gamma = 1$ gewichtet.

#### 3.5.2.3. Verbesserte Qualitätsberechnung

Eine nächste Erweiterung des *Smart Display Mappings* ist die verbesserte Qualitätsberechnung. So wurde zum Ersten die Berechnung der *räumlichen Qualität* erweitert. Zum Zweiten wurde die Berechnung der *zeitlichen Kontinuität* ( $q_t$ ) reduziert.

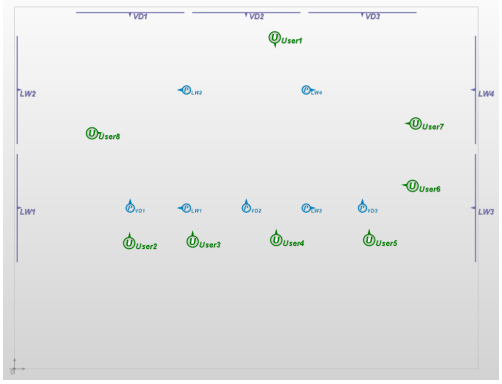
**Verbesserte räumliche Qualität**

Die räumliche Qualität beeinflusst die Zuordnung von *Views* zu den verfügbaren Displayflächen. Heider definiert in [Hei09] die räumliche Qualität eines Mappings $q_s(m)$ folgendermaßen:

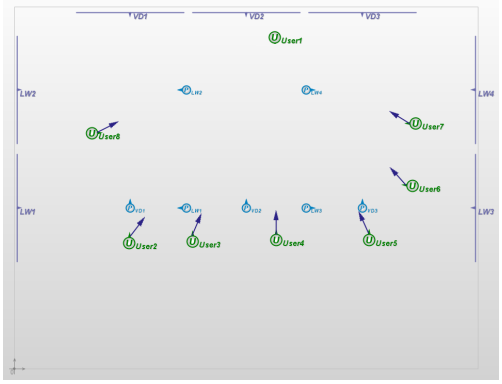
$$q_s(m) = \sum_{\substack{u \in U \\ d \in D}} input(d,u) \cdot \max_{y \in m(d)} vis(y,u) \tag{3.5}$$

Dabei sind $input(d,u) \in [0..1]$ die Wichtigkeit eines Dokuments $d$ für einen Nutzer $u$ und $vis(y,u) \in [0..1]$ die Sichtbarkeit einer Displayfläche $y$ vom Nutzer $u$.

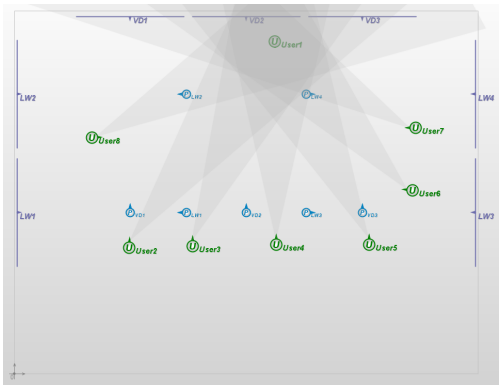
Zur Berechnung von *vis* wird hier lediglich der Winkel zwischen Oberflächennormale (der Displayfläche) und Nutzer bzw. Projektor verwendet. Allerdings werden zwei entscheidende Faktoren bei der Berechnung nicht berücksichtigt. So wird zum einen der Abstand der Nutzer von der Displayfläche nicht berücksichtigt. Zum anderen wird der Einfluss des primären Gesichtsfeldes auf die Erkennbarkeit von Information nicht betrachtet.



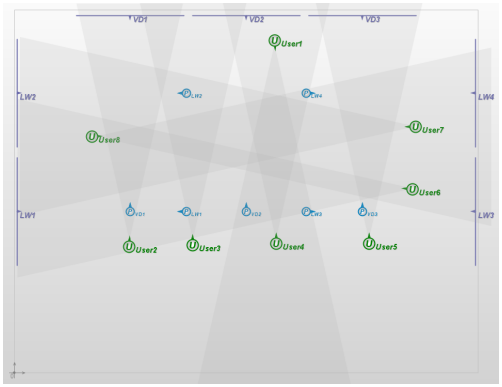
(a) Ausgangssituation im Smart Meeting Room.



(b) Szenario *Präsentation*: Die blauen Pfeile repräsentieren die angenommene Blickrichtung der Nutzer (User2 - User8) zum Presenter (User1).



(c) Szenario *Präsentation*: Die angenommene Blickrichtung zum Presenter ergibt die für die Nutzer dargestellten Blickkegel. Die Überlagerung der Blickkegel verdeutlicht, dass VD2 die von allen Nutzern am besten sichtbare Displayfläche ist.



(d) Szenario *Diskussion*: Durch die realen Blickrichtungen ergeben sich die dargestellten Blickkegel. Diese werden im Szenario der *Diskussion* für alle Nutzer verwendet.

Abbildung 3.6.: Schematische Darstellung des Smart Meeting Rooms als Draufsicht. Nutzer werden durch grüne Kreise (User1 – User8) repräsentiert, wobei der Pfeil die aktuelle Blickrichtung anzeigt. Die mit P markierten blauen Kreise repräsentieren einen Projektor, der jeweils den gleichen Namen trägt wie die Leinwand, auf die der Projektor strahlt. Die Leinwände werden durch blaue Striche mit Bezeichnung (LW1 – LW4, VD1 – VD3) symbolisiert.

Gesucht ist daher eine Funktion ($fov$), mit deren Hilfe die Qualität $q_s(m)$ um die Parameter des Abstandes des Nutzers von der Displayfläche und der Erkennbarkeit von Information auf Basis des primären Gesichtsfelds erweitert werden kann.

Der naive Ansatz wäre, die euklidische Distanz von Nutzer zu Displayfläche als Maß der Distanz zu nutzen. Dies ist aber für heterogene Displays ungeeignet, da z.B. eine Distanz von 2 m zu einem Smart Phone Display eine andere Wirkung hat, als dieselbe Distanz zu einem großen 60" Fernseher.

Um diese Heterogenität der Displayflächen mit einzubeziehen, soll im Rahmen dieser Arbeit zusätzlich zur Entfernung auch das überdeckte Gesichtsfeld des Nutzers berücksichtigt werden. Da hier perspektivisch korrekte Berechnungen zu aufwändig sind, wird der Winkel der von der Displayfläche überdeckten Sichtfläche $\gamma_{\{h,v\}}$ sowohl horizontal als auch vertikal approximiert. Die für die Approximation verwendeten Parameter werden in Abbildung 3.7 illustriert. Berechnet wird die von der Displayfläche überdeckte Sichtfläche $\gamma_{\{h,v\}}$ wie folgt:

$$\gamma_{\{h,v\}} = \arctan\left(\frac{\cos\alpha_{\{v,h\}} \cdot ha_{\{h,v\}}}{d + \sin\alpha_{\{v,h\}} \cdot a_{\{h,v\}}}\right) + \arctan\left(\frac{\cos\alpha_{\{v,h\}} \cdot a_{\{h,v\}}}{d - \sin\alpha_{\{v,h\}} \cdot a_{\{h,v\}}}\right) \tag{3.6}$$

Dabei sind $\alpha$ der Rotationswinkel der Displayfläche, $d$ die Distanz zwischen Mittelpunkt der Displayfläche und Nutzer und $a$ die halbe Ausdehnung der Displayfläche (siehe Abbildung 3.7). Ist $(d - \sin\alpha \cdot ha)$ negativ, werden zusätzlich 180° addiert.

Der Winkel selbst gibt allerdings nur ungenügend Auskunft über die Sichtbarkeit einer Displayfläche und der dort dargestellten *Views*. Deshalb wird eine zusätzliche Bewertungsfunktion verwendet, die es ermöglicht, diesen Winkel für die Einschätzung der Qualität in einem Wertebereich $[0..1]$ anzugeben. Diese Bewertungsfunktion basiert auf grundlegenden Werten der menschlichen visuellen Wahrnehmung. Beim Menschen wird grundsätzlich zwischen dem zentralen Blickfeld und dem peripheren Gesichtsfeld unterschieden. Das gesamte Gesichtsfeld des Menschen überdeckt dabei horizontal 180° und vertikal 60° bis 70° [AP80]. Das Blickfeld (bzw. Gebrauchsblickfeld), also der Bereich des Auges, in dem eine scharfe Farbwahrnehmung möglich ist, beträgt horizontal ca. 20° und vertikal ca. 60° [Kau03]. Daher ist es sinnvoll, die Werte des Blickfelds zu nutzen, um den überdeckten Winkel der Displayfläche ( $\gamma_{\{h,v\}}$ ) zu bewerten.

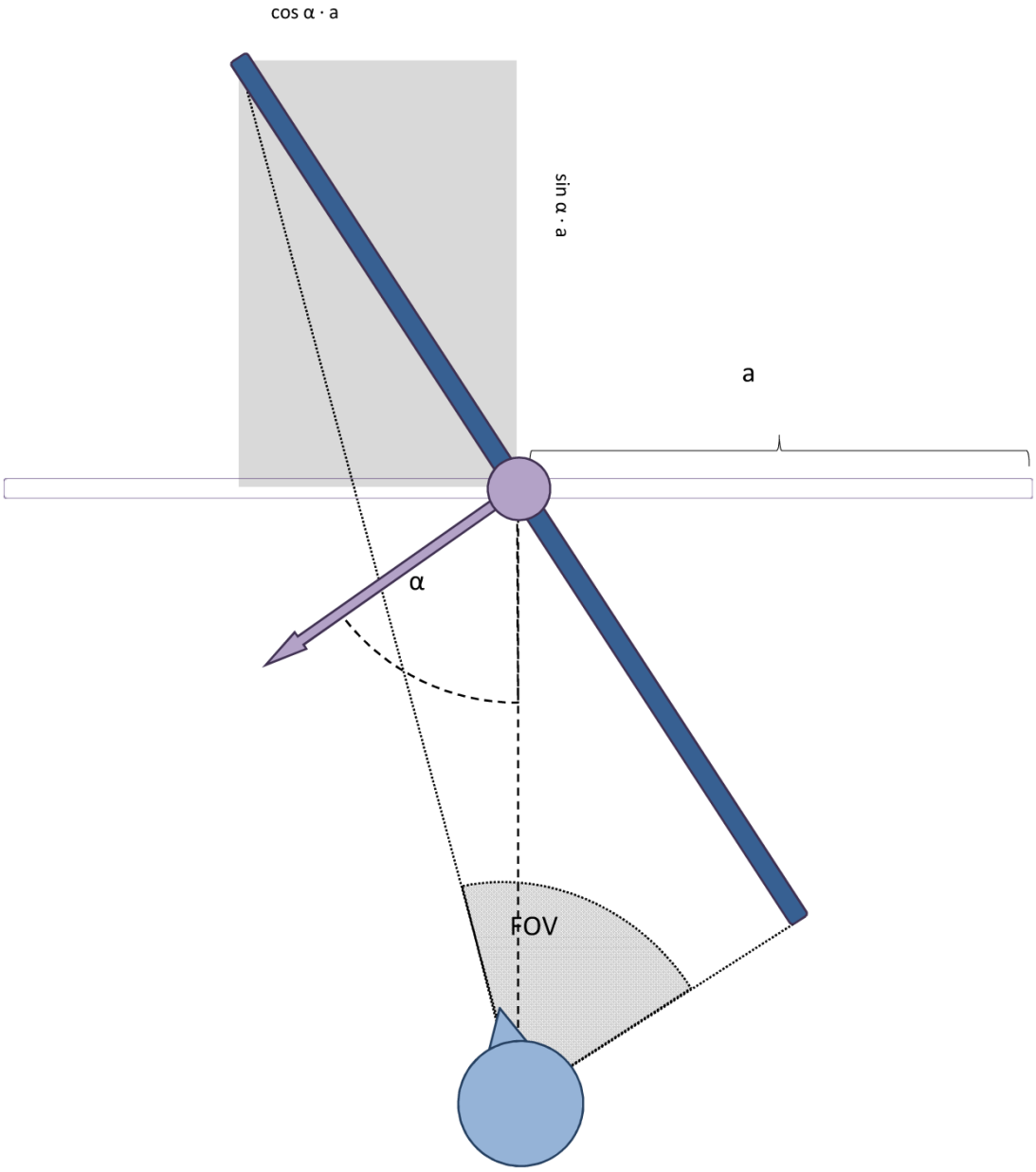
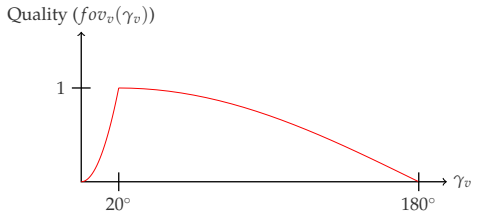
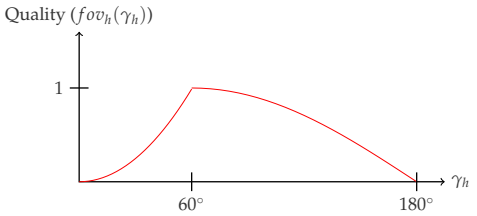



Abbildung 3.7.: Diese Abbildung zeigt die Parameter, die für die Berechnung des Winkels der von der Displayfläche überdeckten Sichtfläche (Siehe Formel 3.6) zugrunde gelegt werden.




(a) Qualität der Darstellung in Abhängigkeit des überdeckten Blickfelds in horizontaler Richtung.




(b) Qualität der Darstellung in Abhängigkeit des überdeckten Blickfeldes in vertikaler Richtung.

Abbildung 3.8.: Die Kurven dieser beiden Abbildungen repräsentieren den Verlauf der Funktionen 3.7 (Abbildung (a)) und 3.8 (Abbildung (b)).

Grundsätzlich ist die Wahrnehmbarkeit einer Displayfläche bei einem überdeckten Gesichtsfeld von 20° optimal. Wird ein geringerer Teil des Gesichtsfeldes überdeckt, wird die Wahrnehmbarkeit eingeschränkt, da z.B. Schrift zu klein zum Lesen wird. Hier wird daher, ausgehend vom Optimum, eine Exponentialfunktion für den Abfall der Qualität verwendet. Wird allerdings ein größerer Teil überdeckt, kann angenommen werden, dass Darstellungen außerhalb des Blickfeldes des Nutzers liegen und daher die Sichtbarkeit nicht gewährleistet ist. Da die Lesbarkeit der einzelnen Informationen noch gewährleistet ist und nur der Komfort abnimmt, wird hier eine sehr viel geringere Abnahme der Qualität bei Zunahme des Winkels verwendet. Deshalb kommt eine gestreckte Kosinus Funktion zum Einsatz (siehe Abbildung 3.8a und 3.8b). Die Bewertungsfunktionen $fov_h(\gamma_h)$ und $fov_v(\gamma_v)$ sehen dadurch wie folgt aus:

$$fov_h(\gamma_h) := \begin{cases} \frac{1}{20^n}\gamma_n^h, n > 1 & , \gamma_h < 20° \\ \cos(\frac{9}{16}\gamma_h - \frac{9 \cdot 20}{16}) & , \gamma_h \geq 20° \end{cases} \tag{3.7}$$

$$fov_v(\gamma_v) := \begin{cases} \frac{1}{60^n}\gamma_n^v, n > 1 & , \gamma_v < 60° \\ \cos(\frac{9}{12}\gamma_v - \frac{9 \cdot 60}{12}) & , \gamma_v \geq 60° \end{cases} \tag{3.8}$$

Mit $n$ wird der Grad der Exponentialfunktion angegeben, die für die Zunahme der Qualität im Blickfeld des Nutzers verwendet wird. Je höher hier $n$ gewählt wird, desto steiler steigt die Funktion in diesem Bereich an. Die Nutzung von $n = 2$ hat sich praktisch als ausreichend erwiesen.

Die in diesen Formeln auftretenden Konstanten werden für die Abbildung auf die Funktionsabschnitte verwendet. Im horizontalen Fall wird $\gamma_h$ mit $\frac{9}{16}$ multipliziert und $\frac{9 \cdot 20}{16}$ subtrahiert. Dadurch wird der Wertebereich von $[20°..180°]$ auf den Wertebereich $[0°..180°]$ abgebildet. Damit liefert $\gamma_h$ im Bereich $[20°..180°]$ eine Qualität im Bereich $[0..1]$. Analog verhält es sich mit der vertikalen Qualität.

Um das Gesichtsfeld und den Abstand der Nutzer von der Displayfläche bei der Bewertung des *Display Mappings* mit einzubeziehen, wird die traditionell berechnete räumliche Qualität $q_s(m)$ mit $fov = \min(fov_h, fov_v)$ multipliziert.

Auch die Möglichkeit, multiple *Views* auf einer Displayfläche darstellen zu können, muss bei der räumlichen Qualität berücksichtigt werden. Mit steigender Anzahl von *Views*, die auf einer Displayfläche angezeigt werden sollen, sinkt die verfügbare Displayfläche pro *View*. Sei $m$ die Anzahl der anzuzeigenden *Views* und $n$ die Anzahl der *Views*, die auf der spezifischen Displayfläche angezeigt werden, wird der Skalierungsfaktor $sf$ folgendermaßen berechnet:

$$sf = 1 - (\frac{n}{m}) \tag{3.9}$$

Die räumliche Qualität $q_s(m)$ wird dann mit $sf$ multipliziert. Dadurch wird eine Verteilung der *Views* auf den zur Verfügung stehenden Displayflächen erreicht.

Durch die Erweiterungen bei der Berechnung der räumlichen Qualität lassen sich nun die Eigenschaften der menschlichen Wahrnehmung besser mit einbeziehen. Neben der Berücksichtigung des Gesichtsfeldes können außerdem Eigenschaften der Displayfläche in die Berechnung der Qualität aufgenommen werden.

**Qualität der zeitlichen Kontinuität**

Die Qualität der zeitlichen Kontinuität wird in [Hei09] dazu verwendet, dass keine unnötigen Änderungen im Display Mapping auftreten. So wird z.B. verhindert, dass ein Display Mapping mit einem qualitativ sehr geringen Unterschied zum aktuellen Mapping verwendet wird, wofür die gesamte Zuordnung von *Views* zu Displayflächen verändert werden müsste. Heider beschreibt, dass dafür alle Nutzer, *Views* und die korrespondierende Sichtbarkeit dafür berücksichtigt werden müssen.

Eine Reduzierung von Mapping Veränderungen kann aber auch dadurch erreicht werden, dass lediglich Fälle betrachtet werden, die definitiv ein Update des Mappings

benötigen. Darum wird ein neues Mapping verwendet, (1) wenn sich die Anzahl der Nutzer ändert, (2) wenn sich die Anzahl der anzuzeigenden *Views* ändert (*View* wird entfernt, *View* wird hinzugefügt), (3) wenn eine signifikante Änderung der Position eines oder mehrerer Nutzer auftritt, (4) wenn im neuen Mapping eine *View* einer neuen Displayfläche zugeordnet werden soll und der Abstand der Displayfläche einen Grenzwert überschreitet und die *View* für eine minimale Zeitspanne auf der Displayfläche verweilte (wodurch ein Flackern verhindert wird).

Durch die Berücksichtigung dieser Fälle kann die Qualität $q_t$ aus der Qualitätsfunktion entfernt werden. Stattdessen wird ein aktuelles Mapping analysiert und bei Bedarf umgesetzt. Dadurch wird die zeitliche Kontinuität erhalten ohne die Qualität für jedes Mapping separat berechnen zu müssen und der Berechnungsaufwand für die Qualität des *Display Mappings* reduziert sich.

Mit Hilfe des *Smart Display Mappings* wurden die generierten *Views* automatisch den zur Verfügung stehenden Displayflächen zugeordnet (siehe Abbildung 3.9). Dafür wurden sowohl die *View Packages* und ihre *View Package Descriptions* (Szenario, Aufgabe) verwendet als auch die Daten über die Nutzer (Position, Blickrichtung) und den Raum (u.a. Position der Leinwände). Der nächste Schritt ist die automatische Anordnung der *Views* auf den verschiedenen Displayflächen.

## 3.6. Smart View Layout

Der letzte Schritt im *Smart View Management* ist die Darstellung der *Views*. Hier ist die Herausforderung multiple *Views* verschiedener Größe automatisch auf einer Displayfläche anzuordnen. Dieses entspricht dem allgemeinen Layout Problem, der Anordnung einer beliebigen Menge von nicht überlappenden Rechtecken in einem minimalen rechteckigen Raum. Beach [Bea85] zeigt, dass dieses Problem NP-vollständig ist.

In den folgenden Abschnitten wird zuerst ein Überblick über den Stand der Forschung bzgl. des Problems der automatischen Anordnung von *Views* gegeben (vgl. Abschnitt 3.6.1). Danach werden zwei Layout Mechanismen des *Smart View Managements* beschrieben: ein *Federkraftbasiertes Layout* (vgl. Abschnitt 3.6.2) und ein *Gitterbasiertes Layout* (vgl. Abschnitt 3.6.3).

Abbildung 3.9.: Diese Abbildung zeigt die *Views*, die vorher erzeugt, mit *View Descriptions* beschrieben, mit *JPEG2000* kodiert und zu *View Packages* gruppiert wurden. Für das *Smart Display Mapping* wird ein Publikum aus Sicht der Kamera angenommen. Durch das *Smart Display Mapping* wurden die *Views* verschiedenen Displayflächen zugeordnet. Die sechs *Views* des *View Packages* mit der Aufgabe *Vergleich* wurden auf einer Displayfläche rechts zugeordnet. Die *Views* zur Präsentation wurden auf den restlichen beiden Displayflächen verteilt.

### 3.6.1. Automatische Layouts - Stand der Forschung

Ein Graphisches Layout beschäftigt sich damit, Position und Größe einer Menge von graphischen Objekten in einer Darstellung festzulegen [Ali09]. Durch die Komplexität dieses Problems und durch die zusätzlichen ästhetischen Anforderungen, die oft an Layouts gestellt werden, ist das Designen von Layouts auch heutzutage eine Aufgabe, für den es eines menschlichen Experten bedarf.

Ziel bei digitalen Medien ist es allerdings, automatische Layouts zu generieren. Hier besteht das Problem, dass neben den allgemeinen Anforderungen an statische Medien zusätzlich wechselnder Inhalt und wechselnde Anforderungen bei der Betrachtung hinzukommen [Ali09].

Um Anforderungen an Layouts zu spezifizieren, werden Constraints (Bedingungen) definiert. Dabei werden zwei Typen von Bedingungen unterschieden: *abstrakte* und *räumliche*. Abstrakte Bedingungen beschreiben semantische Zusammenhänge auf einem höheren Level (z.B. Bild 1 ist wichtig), wobei räumliche Bedingungen die geometrischen Zusammenhänge von Elementen beschreiben (z.B. Bild 1 über Text 1). Weiterhin wird zwischen *harten* und *weichen* Bedingungen unterschieden. Während harte Bedingungen notwendigerweise erfüllt werden müssen, beschreiben weiche Bedingungen Präferenzen mit unterschiedlichen Wichtigkeiten [Ali09].

Das allgemeine Layout Problem kann z.B. mit harten räumlichen Constraints beschrieben werden. Hier wird verlangt, dass graphische Elemente sich nicht verdecken dürfen und alle Elemente vollständig im aktuellen Dokument enthalten sind – also nicht den Rand schneiden oder darüber hinausgehen.

Zur Realisierung von Layouts werden deshalb in der Regel entweder federkraftbasierte Layouts verwendet, die verschiedene Constraints berücksichtigen oder templatebasierte Layouts, um den Suchraum für die Anordnung der graphischen Elemente zu verringern. Dadurch lässt sich schnell eine zufriedenstellende Lösung des Problems erreichen. Ein Nachteil ist, dass Templates auf den Inhalt und die Verwendung abgestimmt werden müssen, um ein gutes Resultat zu erzielen [Ali09].

**Federkraftbasierte Layouts**

Beschränkungen lassen sich auch mit Hilfe von kraftbasierten Layouttechniken lösen. Hier werden die Beschränkungen durch Federkräfte beschrieben. Solche kraftbasierten Techniken werden häufig für das Layout von Graphen verwendet [BETT98, FR91]. Ein erweiterter Ansatz wurde von Lüders et al. [LES95] entwickelt. Sie nutzen einen kraft- und druckbasierten Ansatz, um Fenster einer GUI automatisch anzuordnen. Ali et al. entwickelte weiterhin einen Ansatz, mit dem neben konkaven Elementen auch zusätzliche Bedingungen, wie z.B. die relative Ausrichtung von graphischen Elementen untereinander (Bild 1 bis 3 horizontal nebeneinander), definiert werden können, um eine gewünschte Anordnung der graphischen Elemente zu erhalten [AHFS08].

**Gitterbasierte Layouts**

*Gitterbasierte Layouts* werden häufig im Graphikdesign eingesetzt, um effizient Grafiken und Texte zu organisieren [MB81]. Bei *Gitterbasierten Layouts* werden die graphischen Elemente bzgl. Position, Größe und Seitenverhältnis der Gitterzellen angeordnet. Durch das Gitter werden zusätzliche Beschränkungen für das Layout eingeführt, welches den Suchraum stark reduziert. Beispiele für *Gitterbasierte Layouts* finden sich in [Fei98, JLS+03, WW94].

Im Rahmen dieser Arbeit werden beide Layout Varianten untersucht. Es wird ein Federkraftbasiertes Layout verwendet, um eine Anordnung der *Views* basierend auf den Einschränkungen des gegebenen Problems zu generieren und ein *Gitterbasiertes Layout*, ein einfaches Template (ein Gitter), um den Suchraum zu verkleinern, wobei *Views* einzelnen Gitterzellen zugeordnet werden.

Darstellung und Anordnung der *Views* auf einer Displayfläche sind im *Smart View Management* der letzte Schritt der Anzeige. Wie auch bei der Zuweisung der *Views* zu den Displayflächen ist eine automatische Anordnung hier von großer Wichtigkeit. Im Folgenden werden daher die zwei im Rahmen dieser Arbeit entwickelten automatischen Layout Strategien genauer vorgestellt.

### 3.6.2. Federkraftbasiertes Layout

Die grundsätzliche Anforderung an das Layout von *Views* im *Smart View Management* ist, dass *Views* überlagerungsfrei und vollständig angezeigt werden sollen. Dadurch wird die Sichtbarkeit der Informationen gewährleistet. Weiterhin ist es wünschenswert, die *Views* mit größtmöglicher Auflösung darzustellen, um die Erkennbarkeit der Informationen bestmöglich zu unterstützen. Diese Constraints liefern die Grundlage für die Berechnung des Layouts.

Zur Auflösung solcher Constraints werden federkraftbasierte Layout Mechanismen verwendet. In [AHFS08] stellen Ali et al. einen solchen Ansatz vor. Dies ist ein zweistufiger Ansatz: Im ersten Schritt werden von Designern Templates für verschiedene Anwendungsfälle entworfen. Diese Templates enthalten verschiedene Anordnungen der visuellen Elemente für verschiedene Seitenverhältnisse der Displayflächen oder für verschiedene Geräteklassen der anzeigenden Geräte. Weiterhin wird von den Designern eine Beschreibung der anzuzeigenden Elemente erstellt, wie z.B. Inhalt (Text, Illustrationen, Animationen, ...), Größe und Wichtigkeit. Mit Hilfe dieser Templates und Beschreibungen wird dann ein initiales Layout erstellt. Im zweiten Schritt wird ein federkraftbasierter Layout Mechanismus verwendet, um das Layout an die aktuelle Dokumentengröße anzupassen.

Ali et al. definieren das Dokumenten Layout Problem als Graphen Layout Problem. Ein Graph $G = (V, E)$ besteht dabei aus einer Menge von Knoten $V$ und einer Menge von Kanten $E$, wobei jede Kante ein Paar von Knoten verbindet. Sind zwei Knoten mit einer Kante verbunden, dann werden basierend darauf anziehende und abstoßende Kräfte definiert. Anziehende Kräfte bewirken, dass sich Knoten gegenseitig anziehen und damit im Layout näher zusammenrücken. Abstoßende Kräfte hingegen bewirken ein auseinander Bewegen. Während durch die anziehenden Kräfte erreicht wird, dass sich die graphischen Elemente nicht willkürlich verstreuen, sondern zusammenbleiben, wird durch die abstoßenden Kräfte eine Überlagerung verhindert und ein Mindestabstand zwischen graphischen Elementen garantiert. Basierend auf dem genutzten Template und der Beschreibung des Inhalts wird der Graph aufgebaut und es werden die Kräfte untereinander definiert.

Um die Größe der graphischen Elemente zu bestimmen, wird ein druckbasierter Ansatz verwendet. Für jedes Element wird jeweils ein innerer und für das gesamte Dokument ein äußerer Druck bestimmt. Das Verhältnis von innerem zu äußerem Druck reguliert dann, ob ein graphisches Element vergrößert oder verkleinert wird. Ist der innere Druck größer als der äußere, dehnt sich ein graphisches Element aus, umgekehrt wird es verkleinert.

Im *Smart View Management* ist das vorherige Erstellen spezifischer Templates nicht möglich. Die *Views* werden dynamisch generiert und beliebig in *View Packages* zusammengefasst. Auch kann die Anzeige auf beliebigen Displayflächen mit einer beliebigen Anzahl von *Views* erfolgen, die nicht nur sehr verschiedene Ausprägungen haben können, sondern sich auch dynamisch verändern.

Daher wird im *Federkraftbasierten Layout* des *Smart View Managements* lediglich der zweite Teil des Ansatzes von Ali et al. verwendet. Auf Grund des Fehlens eines Templates wird ein vollständiger Graph angenommen, d.h. jeder Knoten ist mit jedem verbunden. Um die grundsätzlichen Anforderungen an das Layout zu erfüllen (keine Überlagerung der *Views*, vollständig auf der Displayfläche dargestellt) werden nun Kräfte zwischen den *Views* definiert (Abbildungen 3.10a–3.10c):

- Abstoßende Kräfte zwischen *Views*: Verhindern der Überlagerung von *Views*.
- Abstoßende Kräfte zwischen *Views* und Rändern: Halten der *Views* innerhalb der Displayfläche.
- Anziehende Kräfte zwischen *Views*: Zusammenhalt der *Views* zum Verhindern des Verstreuens über die Displayfläche.

Weiterhin wird der druckbasierte Ansatz für die Skalierung von *Views* mit der Definition von innerem und äußerem Druck von Ali et al. [AHFS08] übernommen (siehe Abbildung 3.10d). Bei Überschreiten des inneren Drucks zum äußeren Druck wird die nächstgrößere Zielauflösung aus der *View Description* ausgewählt und die *View* in dieser Größe dargestellt. Übersteigt im Gegensatz dazu der äußere Druck den inneren Druck, wird eine kleinere Auflösung aus den Zielauflösungen ausgewählt. Dieses Vorgehen hat zwei prinzipielle Nachteile:

- Ali et al. zufolge führt das Fehlen eines initialen Layouts, generiert durch ein Template, zu Artefakten und erzeugt demnach nicht zwangsläufig ein gutes Resultat.
- Die Anforderungen des Layouts im *Smart View Management* entsprechen dem allgemeinen Layoutproblem. Nach Beach [Bea85] ist dies ein NP-vollständiges Problem und lässt sich daher nach Ali [Ali09] nicht mit einem effizienten Algorithmus lösen.

Daher wird im *Federkraftbasierten Layout* des *Smart View Managements* ein iteratives Vorgehen umgesetzt. Die Grundidee ist, ein erstes Layout durch das Federkraft Layout zu berechnen, es dem Nutzer schon zu präsentieren, das Layout dann aber im Hintergrund durch eine Qualitätsfunktion zu bewerten und ggf. ein neues Layout zu berechnen und zu bewerten. Hat das neue Layout eine bessere Qualität als das bisher angezeigte, wird das neue Layout angezeigt. Dadurch kann dem Nutzer ein schnelles Feedback gegeben und gleichfalls eine Optimierung in einem Hintergrundprozess vorgenommen werden. Die Qualität $q$ eines Layouts wird wie folgt berechnet:

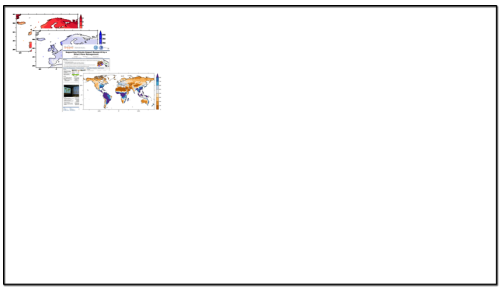
$$q = \frac{c}{a} \cdot \prod_{v \in V} (0.5 \cdot q_h(v) + 0.5 \cdot q_w(v)) \tag{3.10}$$

Dabei ist $c$ die durch alle *Views* $V$ überdeckte Displayfläche mit $c = \sum_{v \in V} w(v) \cdot h(v)$, wobei $w(v)$ die Breite einer und $h(v)$ die Höhe einer *View* $v \in V$ sind. Weiterhin sei $a$ die Fläche der gesamten Displayfläche. Dann wird mit der Qualitätsfunktion die von den *Views* genutzte Fläche der Displayfläche bewertet. Um eine gleichmäßige Skalierung der *Views* zu bevorzugen, wird zusätzlich dieser Wert mit der Skalierungsqualität der *Views* $q_w(v)$: horizontale Skalierungsqualität, $q_h(v)$: vertikale Skalierungsqualität multipliziert. Die Skalierungsqualitäten werden wie folgt berechnet:

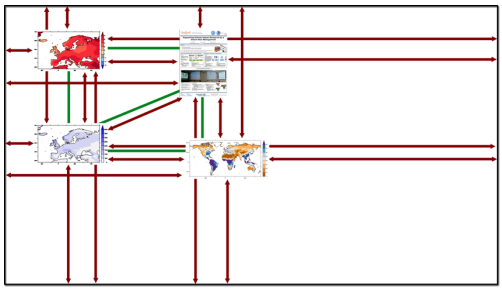
$$q_h(v) := \begin{cases} 1 & \text{, wenn } h_p(v) \leq h_c(v), \\ \frac{h_c(v)}{h_p(v)} & \text{, sonst} \end{cases} \tag{3.11}$$

$$q_w(v) := \begin{cases} 1 & \text{, wenn } w_p(v) \leq w_c(v), \\ \frac{w_c(v)}{w_p(v)} & \text{, sonst} \end{cases} \tag{3.12}$$

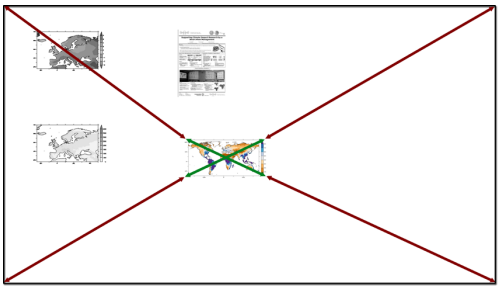
So wird die Anforderung an das Layout im *Smart View Management* umgesetzt, *Views* so groß wie möglich darzustellen, um eine gute Wahrnehmbarkeit zu unterstützen.



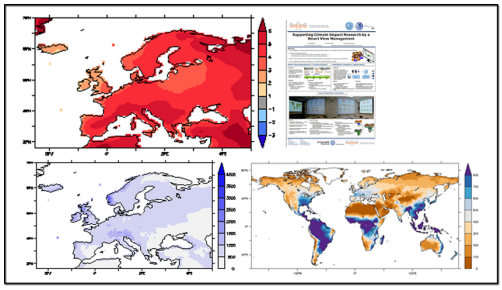
(a) Ausgangssituation für das *Federkraftbasierte Layout*.



(b) Definition der Federkräfte (rot: abstoßende Kräfte, grün: anziehende Kräfte).



(c) Definition der druckbasierten Skalierung (rot: äußerer Druck, grün: innerer Druck).



(d) Mögliches Resultat des *Federkraftbasierten Layouts*.

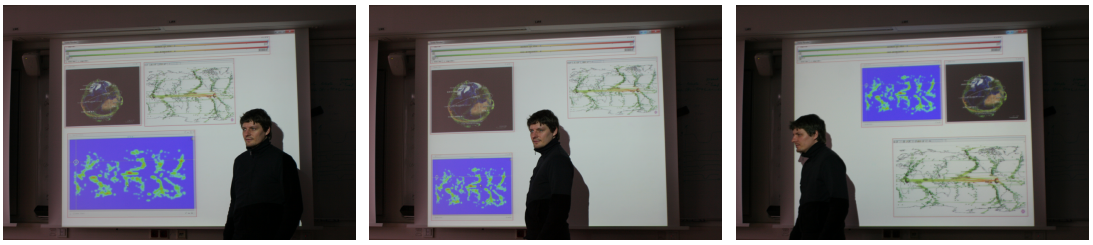
Abbildung 3.10.: Die Abbildungen zeigen den Ablauf des *Federkraftbasierten Layouts*. Abbildung (a) zeigt die initiale Anordnung der *Views*. Abbildung (b) zeigt die Kräfte, wie sie zwischen den *Views* und den Rändern definiert wurden. Rote Pfeile symbolisieren abstoßende und grüne Striche anziehende Kräfte. Abbildung (c) zeigt die druckbasierte Skalierung der *Views*. Ein mögliches Resultat des gesamten *Federkraftbasierten Layouts* zeigt Abbildung (d).

Wobei $w_p(v)$ die bevorzugte Breite (respektive $h_p(v)$ die bevorzugte Höhe) einer *View* darstellt und $w_c(v)$ die aktuelle Breite (respektive $w_p(v)$ die aktuelle Höhe). Die Informationen über die bevorzugte Höhe und Breite der *Views* wird aus der *View Description* entnommen (vgl. Abschnitt 3.3). Unter der Kategorie der Eigenschaften werden zu einer *View* dort die Quellauflösung und die Zielauflösungen angegeben. Die Zielauflösungen beschreiben dabei, in welcher Auflösung die *View* dargestellt werden kann. Die Quellauflösung wiederum stellt die native Auflösung einer *View* dar und wird deshalb als bevorzugte Auflösung verwendet. Die Höhe und die Breite der Quellauflösung stellen damit die bevorzugte Höhe und Breite dar. Somit wird über die Qualitätsfunktion die gleichmäßige Skalierung bei möglichst maximaler Ausnutzung der Displayfläche bevorzugt. Übersteigt die Qualität des neuen Layouts die Qualität des aktuellen, wird das neue Layout dargestellt.

Eine, im Rahmen dieser Arbeit entwickelte, Erweiterung dieses federkraftbasierten Layout Mechanismus ist die dynamische Behandlung von Verdeckungen durch einen Presenter. Vor allem im Szenario der Präsentation kann es häufig dazu kommen, dass der Presenter einen Teil der Displayfläche verdeckt. In Smart Meeting Rooms und im speziellen im Smart Lab in Rostock sind die Wände mit vielen Projektionsflächen bestückt. Das hat zur Folge, dass der Presenter wenige Stellen im Raum hat, in denen er nicht im Projektionskegel eines Projektors steht und damit einen Teil einer Displayfläche verdeckt. Die dort angezeigten *Views* sind entsprechend nicht sichtbar.

Da im Smart Meeting Room die Position des Presenters bekannt ist, wird diese Information dafür verwendet, die *Views* so anzuordnen, dass sie nicht vom Presenter verdeckt werden. Dafür wird im *Federkraftbasierten Layout* ein weiteres Element eingefügt, das als verdeckte Fläche dient. Innerhalb dieser Fläche wird keine *View* angezeigt. Die Position und die Größe dieser Fläche werden durch eine Projektion des Presenters auf die jeweilige Displayfläche berechnet.

Im Federkraftmodell wird diese verdeckte Fläche nicht durch die Federkräfte beeinflusst. Allerdings werden abstoßende Kräfte zwischen der verdeckenden Fläche und den *Views* definiert. Damit erhält diese Fläche denselben Stellenwert im Layout wie der Rand. Als Folge werden durch das automatische *Federkraftbasierte Layout* keine *Views* im verdeckten Bereich angezeigt.



(a) (b) (c)

Abbildung 3.11.: Diese Abbildungen zeigen die Anpassung des Layouts an die Position des Presenters. Verändert der Presenter seine Position, werden die *Views* so angeordnet, dass der er keine *View* verdeckt.

### 3.6.3. Gitterbasiertes Layout

Auch hier gilt wieder die grundsätzliche Anforderung an das Layout von *Views* im *Smart View Management*, dass *Views* vollständig und überlagerungsfrei dargestellt werden müssen. Es ist ebenso wünschenswert, die *Views* mit größtmöglicher Auflösung darzustellen, um die Erkennbarkeit der Informationen zu unterstützen.

Durch die dynamische *View* Generierung und die dynamische Zuweisung von *Views* zu Displayflächen können im Vorfeld keine Annahmen über geeignete Templates für die Anordnung der *Views* auf den Displayflächen getroffen werden. Daher ist es lediglich möglich, ein allgemeines Template zur Anordnung zu verwenden. Um dabei auch die Anforderungen der vollständigen und überlagerungsfreien Darstellung einzuhalten, wird ein Gitter als Template genutzt. Durch die Teilung der Displayfläche in Gitterzellen und die Anzeige der *Views* in diesen Zellen ist die Anforderung der Überlagerungsfreiheit bereits erfüllt. Auch durch die Darstellung innerhalb der Zelle ist die Anforderung der vollständigen Darstellung erfüllt. Es bleibt die Anforderung der Darstellung der *Views* mit größtmöglicher Auflösung. Für ein *Gitterbasiertes Layout* bedeutet dies, dass möglichst wenig freie Gitterzellen generiert werden sollten und dass die Gitterzellen dem Seitenverhältnis der *Views* entsprechen sollten.

Um dies zu realisieren, wurde im Rahmen dieser Arbeit ein *Gitterbasiertes Layout* Verfahren entwickelt. Grundlage ist hier die Erzeugung eines Gitters mit mindestens so vielen Gitterzellen, wie *Views* angezeigt werden sollen. Die Verteilung der Gitterzel-

len, also die Anzahl der dabei entstehenden Zeilen und Spalten, wird so vorgenommen, dass die Seitenverhältnisse einzelner Gitterzellen in etwa den Seitenverhältnissen der darin anzuzeigenden *Views* entsprechen.

Sei dabei $m$ die Anzahl der anzuzeigenden *Views*, dann wird für die Erzeugung des Gitters durch eine Faktorzerlegung von $m$ die Menge aller möglichen Zeilen- und Spalteneinteilungen $G$ berechnet:

$$G = \{(a, b) \mid a \cdot b = m\} \tag{3.13}$$

Dabei sind $a \in \mathbb{N}$ die Anzahl der Zeilen und $b \in \mathbb{N}$ die Anzahl der Spalten des Gitters.

Dadurch wird eine Menge von regulären Gittern mit gleichmäßiger Gittergröße berechnet. Es gilt nun zu entscheiden, welche Verteilung von Zeilen und Spalten am besten den anzuzeigenden *Views* entspricht.

Für diese Entscheidung gibt es zwei Einflussfaktoren. Diese sind das Seitenverhältnis der Displayfläche und die Seitenverhältnisse der anzuzeigenden *Views*. Die Nutzung dieser beiden Einflussfaktoren wird im Folgenden beschrieben.

**Seitenverhältnis der Displayflächen**

Der erste Schritt ist die Einbeziehung des Seitenverhältnisses der Displayflächen. Dazu wird die Zellverteilung so vorgenommen, dass die Verteilung der Zellen in der Horizontalen und Vertikalen in etwa dem Verhältnis der Displayflächen entspricht.

Das Verhältnis von Zeilen- und Spalteneinteilung des Gitters zur Auflösung der Displayfläche wird durch die Funktion $f$ definiert mit $f : G \to \mathbb{R}$:

$$f(g) = \left| \frac{a}{b} - \frac{h}{w} \right| \tag{3.14}$$

Dabei sind $h$ die Höhe und $w$ die Breite der Displayfläche in Pixeln. Weiterhin sind $a$ die Anzahl der Spalten und $b$ die Anzahl der Zeilen. Für das Tupel $(a, b) \in G$, für das f(g) minimal ist, entspricht die Zeilen- und Spaltenaufteilung dem Seitenverhältnis der Displayfläche am besten.

Ein Beispiel liefert dafür Tabelle 3.1. Hier sollen 12 *Views* auf einer Displayfläche mit $w = 1920$ und $h = 1080$ Pixeln angeordnet werden. In diesem Beispiel würde $(4, 3)$ für die Gitteranordnung ausgewählt werden, da $f(g)$ minimal ist.

Tabelle 3.1.: Beispielrechnung für die Anordnung von 12 *Views* unter Berücksichtigung des Seitenverhältnisses der Displayfläche .

| a | 1 | 2 | 3 | 4 | 6 | 12 |
|---|---|---|---|---|---|---|
| b | 12 | 6 | 4 | 3 | 2 | 1 |
| $f(g) = \left\lvert \frac{a}{b} - \frac{1920}{1080} \right\rvert$ | $1,69\overline{4}...$ | $1,44\overline{4}...$ | $1,02\overline{7}...$ | $0,44\overline{4}...$ | $1,22\overline{2}...$ | $10,22\overline{2}...$ |

**Seitenverhältnis der Views**

Im zweiten Schritt wird auch das Seitenverhältnis der *Views* für die Verteilung der Gitterzellen herangezogen. Dafür wird das durchschnittliche Seitenverhältnis der anzuzeigenden *Views* aus den *View Descriptions* ermittelt und als Grundlage für die Anordnung der Gitterzellen verwendet.

Sind $vw$ die durchschnittliche Breite und $vh$ die durchschnittliche Höhe der anzuzeigenden *Views*, dann wird die Funktion $f(g)$ wie folgt modifiziert:

$$f(g) = \left| \frac{a}{b} \cdot \frac{vw}{vh} - \frac{w}{h} \right| \tag{3.15}$$

Tabelle 3.2 liefert eine Beispielrechnung: 12 *Views* sollen auf einer Displayfläche mit $w = 1920$ und $h = 1080$ Pixel angeordnet werden. Zusätzlich haben die *Views* ein durchschnittliches Seitenverhältnis von 2:3. Dabei wird deutlich, dass eine Verteilung von $(6, 2)$ besser an das Seitenverhältnis der *Views* angepasst ist (Abbildung 3.12).

Dieses Verfahren stößt allerdings an seine Grenzen, wenn die Anzahl der *Views* einer Primzahl entspricht die größer ist als 2. Die Anzahl der auswählbaren Fälle wäre dann auf $(1, n)$ und $(n, 1)$ beschränkt. Um dennoch eine Anpassung sowohl an das Seitenverhältnis der Displayfläche als auch der *Views* zu erreichen, wird in diesem Fall

Tabelle 3.2.: Beispielrechnung für die Anordnung von 12 *Views* unter Berücksichtigung des Seitenverhältnisses der Displayfläche und dem durchschnittlichen Seitenverhältnis der *Views*.

| a | 1 | 2 | 3 | 4 | 6 | 12 |
|---|---|---|---|---|---|---|
| b | 12 | 6 | 4 | 3 | 2 | 1 |
| $f(g) = \left\lvert \frac{a}{b} \cdot \frac{2}{3} - \frac{1920}{1080} \right\rvert$ | $1,72\overline{2}...$ | $1,55\overline{5}...$ | $1,27\overline{7}...$ | $0,88\overline{8}...$ | $0,22\overline{2}...$ | $6,22\overline{2}...$ |

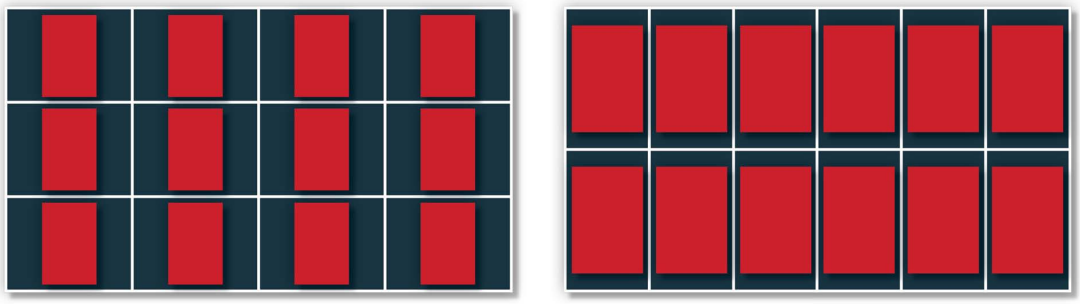
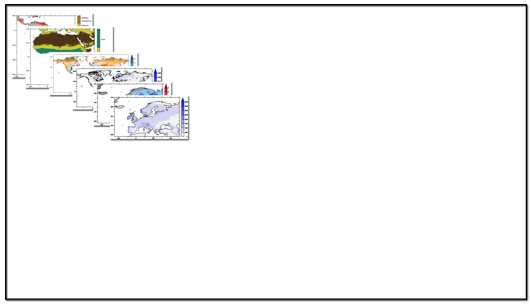


Abbildung 3.12.: Diese Abbildungen zeigen die Berücksichtigung der Seitenverhältnisse der *Views*. Im linken Bild wird ein Gitter verwendet, welches dem Seitenverhältnis der Displayfläche entspricht, während im rechten Bild das Seitenverhältnis der *Views* mit einbezogen wird. Dadurch ergibt sich eine bessere Ausnutzung der Displayfläche, da die *Views* größer dargestellt werden können.

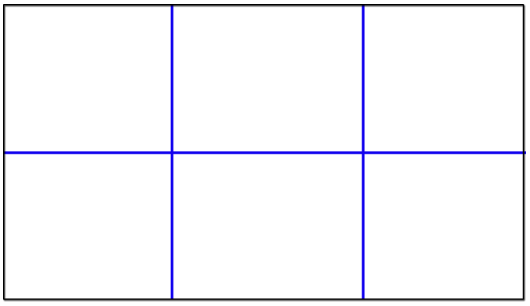
$n + 1$ benutzt. Dadurch entsteht wiederum eine faktorisierbare Anzahl, allerdings auf Kosten einer leeren Zelle. Durch die Anpassung an das Seitenverhältnis ist dennoch mehr Raum für die Darstellung der *Views* gewonnen. Abbildung 3.13 zeigt ein Beispiel für ein *Gitterbasiertes Layout*. Dieses wurde im Rahmen der Bachelorarbeit von Valerius Weigandt entwickelt [Wei12].

Die automatische Anordnung der Views stellt den Abschluss des *Smart View Managements* dar. Die Entscheidung, welches der beiden vorgestellten Layout Verfahren verwendet werden soll, wird im folgenden Abschnitt diskutiert.

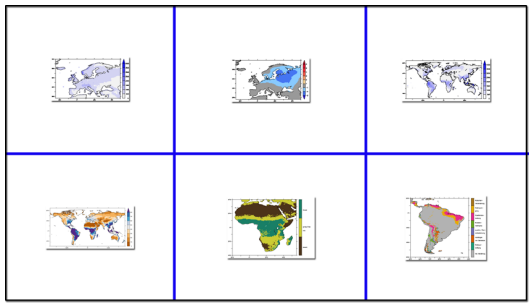
Im ersten Schritt wurden die *Views* generiert, d.h. sie wurden mittels *View Grabber* oder *View API* erzeugt, mit *JPEG2000* enkodiert und mittels *View Descriptions* beschrieben. Im nächsten Schritt wurden die generierten *Views* interaktiv zu *View Packages* gruppiert und mit *View Package Descriptions* beschrieben. Auf dieser Basis wurden die *Views* im folgenden Schritt durch das *Smart Display Mapping* automatisch den Displayflächen zugewiesen. Im letzten Schritt wurden die *Views* auf den jeweiligen Displayflächen durch das *Federkraftbasierte* oder das *Gitterbasierte Layout* angeordnet (siehe Abbildung 3.14).



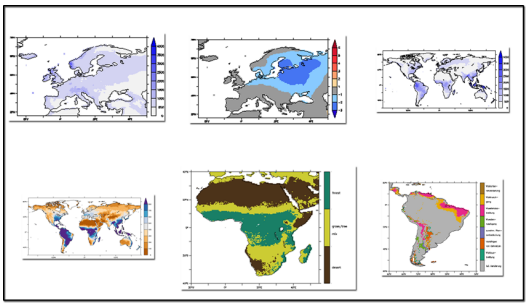
(a) Initial zugeordnete *Views*.



(b) Generiertes Gitter bzgl. der *Views* und der Displayfläche.



(c) In Gitterzellen angeordnete *Views*.



(d) Finales *Gitterbasiertes Layout* auf einer Displayfläche.

Abbildung 3.13.: Die Abbildungen zeigen den Ablauf des *Gitterbasierten Layouts* auf einer Displayfläche. Abbildung (a) zeigt die der Displayfläche zugeordneten *Views*. Abbildung (b) zeigt das Gitter, das für die Anordnung der *Views* und unter Berücksichtigung von Seitenverhältnissen von *Views* und Displayfläche generiert wurde. Abbildung (c) zeigt die Zuordnung der *Views* zu den Gitterzellen. Das Resultat nach Auswahl einer Auflösung aus den *JPEG2000* enkodierten *Views* zeigt Abbildung (d).

Abbildung 3.14.: Nach Bereitstellung, Gruppierung und automatischer Zuordnung zu den Displayflächen wurden die *Views* automatisch angeordnet. Dabei wurden das *Federkraftbasierte Layout* (linke und mittlere Leinwand) und das *Gitterbasierte Layout* (rechte Leinwand) verwendet.

### 3.6.4. Diskussion der Layout Verfahren

Das *Federkraftbasierte Layout* Verfahren liefert eine allgemeine Lösung für die Anforderungen an die Darstellung von *Views* im *Smart View Management*. Außerdem bietet es die Möglichkeit, die Verdeckung zu berücksichtigen, die ein Presenter in einer Präsentation verursacht. Die Anordnung der *Views* bei dieser Darstellung ist allerdings nicht an klaren Grenzen ausgerichtet und wirkt daher sehr ungeordnet.

Das *Gitterbasierte Layout* bietet dagegen durch eine klare Ausrichtung der Views am Gitter eine geordnete Darstellung der *Views*. Durch die Gittererzeugung wird darauf geachtet, die Displayfläche bestmöglich auszunutzen. Da allerdings lediglich das durchschnittliche Seitenverhältnis der *Views* berücksichtigt wird, kann eine gute Ausnutzung der Displayfläche nicht garantiert werden.

Hier stellt sich nun die Frage, unter welchen Umständen welches automatische Layout eingesetzt werden sollte, was im Folgenden diskutiert wird. Dazu wird eine objektive Entscheidungsmöglichkeit basierend auf den Zielen eines Layout Mechanismus im *Smart View Management* vorgestellt. Des Weiteren wird auf Basis einer Nutzerstudie eine subjektive Möglichkeit zur Auswahl eines Layout Mechanismus vorgschlagen.

#### 3.6.4.1. Objektive Auswahl

Ziel der beiden Layout Strategien ist es, eine Anordnung zu finden, bei der die *Views* in größtmöglicher Auflösung dargestellt werden und damit nicht verwendete Displayfläche so gut wie möglich vermieden wird. Daraus lässt sich eine objektive Auswahl der Layout Mechanismen ableiten.

Aus dem Algorithmus zur Erzeugung des Gitters für das *Gitterbasierte Layout* (vgl. Abschnitt 3.6.3) kann bei gegebener Menge an *Views* $V$ mit $n$ *Views* die Größe einer Zelle eindeutig berechnet werden. Weiterhin sind die Zielauflösungen, d.h. die darstellbaren Auflösungen, bei jeder *View* in den *View Descriptions* angegeben. Dadurch kann die maximal überdeckte Fläche einer *View* $A_{v_i} = breite_{v_i} \cdot hoehe_{v_i}$ eindeutig berechnet werden. Zusammen mit der Größe der Displayfläche $A_D = breite_D \cdot hoehe_D$ kann dann der Anteil des freien Raums $f$ von der Displayfläche bestimmt werden durch:

$$f = 1 - \frac{A_D}{\sum_{i=1}^{n}(A_{v_i})} \tag{3.16}$$

Nach Ali [Ali09] ergeben sich konsistente Ergebnisse bei federkraftbasierten Layouts bei einem freien Raum von 20%. Tests mit der Qualitätsfunktion, die für die Bewertung des *Federkraftbasierten Layouts* des *Smart View Managements* vorgenommen wurden, bestätigen dieses Ergebnis.

Aus diesem Grund wird für die Verwendung des *Gitterbasierten Layouts* dieser Wert als Grenze verwendet. Überschreitet der freie Raum $f$ den Wert von $0,2$ (20%), so kann angenommen werden, dass der federkraftbasierte Ansatz eine bessere Platzausnutzung bei der Darstellung der gegebenen *Views* erreicht. Daher kann in diesem Fall automatisch das Federkraftbasierte Layout verwendet werden.

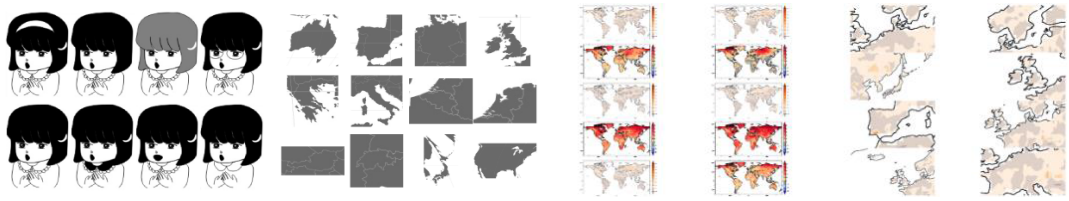
#### 3.6.4.2. Nutzerstudie zur subjektiven Auswahl

Eine weitere Möglichkeit der Entscheidung für einen Layout Mechanismus zur Anordnung von *Views* ist die Durchführung einer Nutzerstudie zur gegenseitigen Evaluation der beiden verwendeten Layout Mechanismen. Im Rahmen einer Studienarbeit [Hol13] wurde eine solche Studie entworfen und durchgeführt.

Ziel der Studie war es, die beiden Layout Verfahren des *Smart View Layouts* an typischen Aufgaben bei der Nutzung von Smart Meeting Rooms vergleichend zu evaluieren. Beim Szenario der Präsentation als auch bei der Diskussion werden die *Views* dafür verwendet, Informationen zu vermitteln. Die Nutzer müssen zwei grundlegende Aufgaben erfüllen, um die dargestellten Informationen extrahieren zu können: (1) *Identifikation* und (2) *Vergleichen*.

- **Identifikation:** Erkennen von Informationen, die in einer *View* dargestellt sind.
- **Vergleich:** Finden von Gemeinsamkeiten oder Unterschieden zwischen zwei oder mehr *Views*.

Für die Durchführung der Studie wurden Layouts von Views im Vorfeld mit den beiden Layout Mechanismen generiert. Diese Bilder wurden den Testpersonen vorgeführt und es wurde ihnen eine spezifische Identifikations- oder Vergleichsaufgabe



(a) (b) (c) (d)

Abbildung 3.15.: Beispiele für Motive der Testbilder (vollständige Liste aller getesteten Bilder siehe Anhang A)

gestellt. Für die Auswertung wurden sowohl die Beantwortungszeit als auch die Korrektheit der Antworten gemessen.

Im Vorfeld wurden drei Hypothesen aufgestellt:

**H1** Bei der Aufgabe *Vergleich* sind die Antwortzeiten beim *Gitterbasierten Layout* kürzer als beim *Federkraftbasierten Layout*.

**H2** Bei der Aufgabe *Identifikation* sind die Antwortzeiten bei beiden Layouts in etwa gleich.

**H3** Die Fehlerrate ist bei beiden Layouts und beiden Aufgaben in etwa gleich.

Im Folgenden wird das genaue Setup der Studie vorgestellt:

**Umgebung und Technik** Die Nutzerstudie wurde im Smart Meeting Room durchgeführt. Damit wurde eine Umgebung genutzt, in der die Layouts praktisch eingesetzt werden. Die Testbilder wurden mit einem Beamer (Epson EMP–82) mit einer Auflösung von 1024x768 Pixel an eine Leinwand projiziert, was eine Displayfläche von 2 m Breite und 1,5 m Höhe ergab. Die Probanden wurden 3,5 m entfernt auf einem Stuhl platziert. Damit ergibt sich aus der Anordnung eine typische Situation im Smart Meeting Room. Die Fenster wurden verdunkelt und das Oberlicht wurde eingeschaltet, um eine einheitliche Beleuchtungssituation zu schaffen.

**Testbilder und Aufgaben** Vier verschiedene Motive kamen als Testbilder zum Einsatz (siehe Abbildung 3.15): Gezeichnete Mädchenköpfe mit unterschiedlichen Merkmalen, stumme Landkarten sowie zwei verschiedene Klimakarten mit unterschiedlicher Koloration.

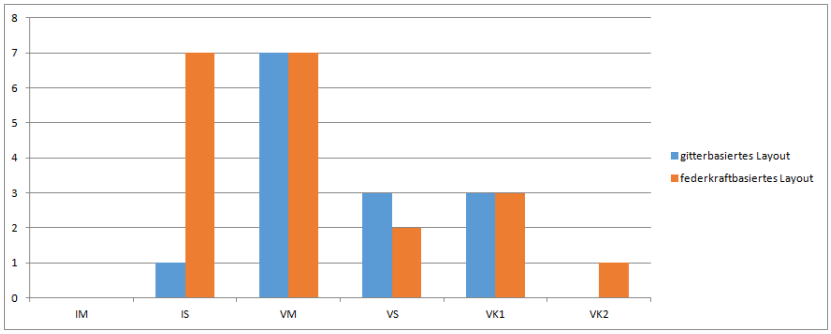
Aus diesen Motiven wurden jeweils verschiedene Layouts für verschiedene Aufgaben erstellt. Bei den Mädchenköpfen musste für die *Identifikation* ein Bild im Layout gefunden werden, das ein vorher spezifiziertes Merkmal hat (z.B. in Abbildung 3.15a): "Finde das Mädchen mit den grauen Haaren"). Für den Vergleich wurden zwei gleiche Köpfe zusammen mit unterschiedlichen Köpfen in einem Layout präsentiert, wobei die Aufgabe darin bestand, die beiden Identischen zu benennen. Ähnlich waren die Aufgaben für die stummen Karten. Für die *Identifikation* wurde die Aufgabe gestellt, ein bestimmtes Land zu finden. Für den Vergleich sollten gleiche Kartenausschnitte gezeigt werden. Bei den Klimakarten wurde lediglich der *Vergleich* getestet. Die Aufgabe bestand darin, die maximal dargestellte Temperatur in einem bestimmten Bereich (z.B. Afrika) zu finden.

Für jede Aufgabe wurden sowohl ein *Gitterbasiertes* und ein *Federkraftbasiertes Layout* erstellt. Die Aufgaben für jedes Layout wurden leicht modifiziert, um einen Lerneffekt zu vermeiden. Dabei wurde darauf geachtet, dass der Schwierigkeitsgrad in etwa gleich ist. Eine vollständige Liste aller getesteten Layouts findet sich im Anhang A.

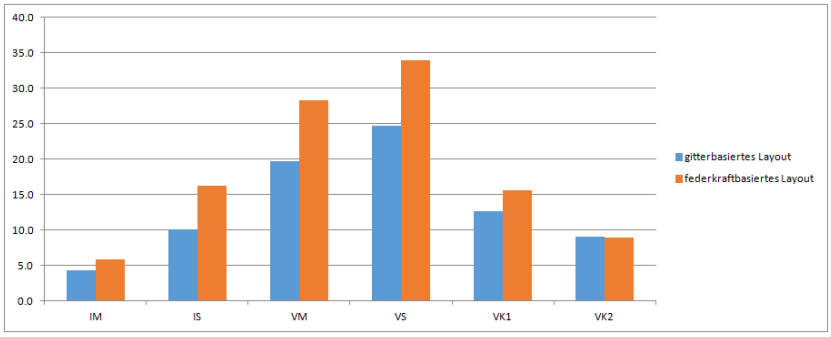
**Probanden** An der Nutzerstudie haben 20 Probanden (6 weiblich, 14 männlich) im Alter von 25 bis 65 Jahren teilgenommen. Unter den Probanden waren 6 Teilnehmer, die Erfahrungen im Bereich des Smart Meeting Rooms aufweisen können, 14 Probanden waren Laien.

**Einweisung und Durchführung** Die Probanden erhielten eine kurze Einführung in die Studie, indem der Ablauf erklärt und Beispiellayouts gezeigt wurden. Die Probanden wurden einzeln eingewiesen und befragt und gaben ihre Antworten durch Zeigen auf die *Views* mittels Laserpointer. Die Testleiterin nahm die Zeit und registrierte die gegebene Antwort.

Am Ende der Studie wurden die Probanden angehalten, ihre Meinung mittels eines Fragebogens abzugeben. Als Grundlage dafür dienten die ästhetischen Kri-




(a)




(b)

Abbildung 3.16.: Ergebnisse der Nutzerstudie zur Untersuchung von *Gitterbasierten* und *Federkraftbasierten Layouts* (IM - Identifikation Mädchen, IS - Identifikation stumme Landkarten, VM - Vergleich Mädchen, VS - Vergleich stumme Landkarten, VK1 - Vergleich Klimakarte Koloration 1 , VK2 - Vergleich Klimakarte Koloration 2). )

terien nach Harrington et al. [HNJ+06]: *Ausrichtung der Objekte an klaren Grenzen, gleichmäßige Größe der Objekte, gleichmäßiger Abstand der Objekte, Leerraumanteil, Einschluss von Leerraumanteil durch Objekte, gleichmäßige Verteilung der Objekte*. Diese Kriterien wurden sowohl für den ästhetischen Eindruck als auch für den Einfluss auf die Bearbeitung der Aufgabe auf einer Skala von 1 (trifft gar nicht zu) bis 6 (trifft voll zu) von den Probanden bewertet.

Abbildung 3.16 zeigt die Ergebnisse der Nutzerstudie. Die Hypothese H1 wurde durch den Test bestätigt. Die benötigten Zeiten waren im Durchschnitt bei allen Tests mit der Aufgabe *Vergleich* bei den *Gitterbasierten Layouts* kürzer.

Hypothese H2 wurde nicht bestätigt. Auch bei der Aufgabe *Identifikation* waren die Antwortzeiten bei der Nutzung des *Gitterbasierten Layouts* im Durchschnitt stets kürzer als beim *Federkraftbasierten Layout*.

Die Hypothese H3 wurde im Grunde bestätigt. Die Fehlerraten waren für alle Layouts in etwa gleich. Die einzige Ausnahme stellt hier die Identifikation auf der stummen Karte mit *Federkraftbasiertem Layout* dar (Abbildung 3.16a, Wert: IS). Hier sollte auf der stummen Karte der *View* identifiziert werden, der Belgien zeigt. Alle Probanden mit falschem Ergebnis gaben an, dass dies an ihren Geografiekenntnissen lag.

Diese Studie gibt einen Hinweis darauf, dass die Bearbeitungszeit einer Aufgabe mit *Gitterbasiertem Layout* im Durchschnitt geringer ist als bei der Nutzung des *Federkraftbasierten Layouts*.

Die Fragebögen, in dem verschiedene Kriterien auf einer Skala von 1 bis 6 abgefragt wurden, zeigen, dass die Probanden es sowohl ästhetisch wichtig als auch wichtig für das Erledigen der Aufgabe empfinden, dass die Objekte an klaren Grenzen ausgerichtet sind (Aufgabe: 4,25, Ästhetik: 4,8), die Objekte gleichmäßig auf der Displayfläche verteilt sind (Aufgabe: 4, Ästhetik: 4) und dass die Objekte gleich groß sind (Aufgabe: 4, Ästhetik: 3,6). Die weiteren Kriterien wurden als eher unwichtig eingestuft.

Die Ergebnisse des Fragebogens belegen, dass die Nutzer ein *Gitterbasiertes Layout* vorziehen, da die Objekte an klaren Grenzen ausgerichtet und gleichmäßig auf der Displayfläche verteilt sind. Für eine einheitliche Größe der Objekte kann in keinem der beiden Layouts garantiert werden.

Aus den Ergebnissen der Bearbeitungszeit und den subjektiven Einschätzungen der Probanden wird subjektiv das *Gitterbasierte Layout* dem *Federkraftbasierten Layout* vorgezogen. Die subjektive Einschätzung ist in diesem Fall dem objektiven Maß vorzuziehen, da sich das objektive Maß auf die Platzausnutzung der *Views* beschränkt, während bei der Nutzerstudie komplexe Faktoren zusammenkommen.

Aus diesem Grund wird das *Gitterbasierte Layout* als Standard im *Smart View Management* verwendet. Ein interaktives Umschalten der beiden Layout Verfahren wird dem Nutzer dennoch stets angeboten. Das objektive Maß für die Platzausnutzung kann dem Nutzer dabei als Entscheidungshilfe dienen.

## 3.7. Implementierung und Anwendung

### 3.7.1. Implementierung

Für eine weitestgehende Unabhängigkeit vom Betriebssystem wurde das *Smart View Management* in Java [Ora13a] implementiert. Das *Smart View Management* besteht aus Komponenten, die in einzelnen Applikationen und auf verschiedenen Geräten gestartet werden können. Im Folgenden werden die einzelnen Komponenten und ihre Funktion genannt:

**SVM Manager** Der SVM Manager bildet die zentrale Kommunikationsstruktur des *Smart View Managements*. Alle folgenden Komponenten melden sich zunächst am *SVM Manager* an. Sollen Daten ausgetauscht werden (z.B. *View* versenden zwischen *View Grabber* und *Metarenderer*), vermittelt der *SVM Manager* die Netzwerkadressen der einzelnen Kommunikationspartner, wickelt die eigentliche Datenübertragung aber nicht selbstständig ab. Der Manager kann auf jedem Gerät des Ensembles gestartet werden.

**View Grabber** Der *View Grabber* wird für die *ad-hoc View Erzeugung* verwendet (vgl. Abschnitt 3.3). Der *View Grabber* kann als eigenständige Applikation unabhängig von anderen Applikationen gestartet werden. Er muss dabei auf dem Gerät laufen, mit dem eine *View* ad-hoc erzeugt werden soll. Für die *JPEG2000* Enkodierung der *Views* wird im *View Grabber* die *kakadu SDK* verwendet [Sof13].

**View API** Die *View API* wird für die *API View Erzeugung* verwendet. Dafür muss die API in eine bestehende Applikation integriert werden. Auch in der API wird für die *JPEG2000* Enkodierung der *Views* im *View Grabber* das *kakadu SDK* verwendet [Sof13].

**Smart Display Mapper** Der *Smart Display Mapper* kann auf einem beliebigen Gerät im Ensemble als eigenständige Applikation gestartet werden. Sind Informationen über *Views* und *View Packages* verfügbar, erstellt der *Smart Display Mapper* automatisch ein Mapping und meldet dies an den SVM Manager. Der Manager verteilt dieses Mapping dann an die für die Anzeige zuständigen Softwarekomponenten.

**Metarenderer** Der *Metarenderer* wird auf einem Gerät gestartet, das einen Beamer für den Smart Meeting Room ansteuert. Der *Metarenderer* wird dann in Vollbild auf der Displayfläche angezeigt. Er kann als eigenständige Applikation gestartet werden. Der *Metarenderer* enthält dabei die Layout Mechanismen für die automatische Anordnung der *Views* auf einer Displayfläche. Für die Dekodierung der *JPEG2000* enkodierten *Views* wird das *kakadu SDK* verwendet [Sof13].

**GUI für die View Package Erstellung** Die GUI wird für die interaktive *View Package* Generierung verwendet (vgl. Abschnitt 3.4). Sie kann sowohl als eigenständige Applikation als auch eingebettet in den *Metarenderer* gestartet werden (vgl. Abschnitt 4).

Durch Java Web Start [Ora13b] wird ein einfacher Zugriff auf alle Applikationen des *Smart View Managements* ermöglicht.

Zudem verwendet das *Smart View Management* die *Helferlein* Middleware [BN12, BRK10] des Smart Labs Rostock (vgl. Abschnitt 2.1.2). Durch die Middleware wird eine Selbstorganisation der einzelnen Komponenten ermöglicht. Dafür meldet sich der SVM Manager in der Middleware an. Jede weitere Komponente durchsucht beim Start die Middleware nach einer *SVM Manager* Instanz. Durch den Eintrag in der Middleware kann dann die Kommunikation zwischen der Softwarekomponente und dem Manager ermöglicht werden, ohne dass zusätzliches Wissen des Nutzers über die Netzwerkadressen notwendig ist.

### 3.7.2. Anwendung

Eine beispielhafte Anwendung des *Smart View Managements* wurde in Zusammenarbeit mit Thomas Nocke vom Potsdamer Institut für Klimafolgenforschung gezeigt [RNS11]. Diese beispielhafte Anwendung dient außerdem dazu, das generelle Vorgehen bei der Anpassung von Desktopapplikationen für die Generierung von *Views* zu verdeutlichen.

Die Klimafolgenforschung befasst sich mit den Auswirkungen des Klimawandels unter anderem auf ökologische Systeme. Dafür müssen Attribute klimatischer Modelle identifiziert werden, die Einfluss auf die Entwicklung ökologischer Systeme haben.

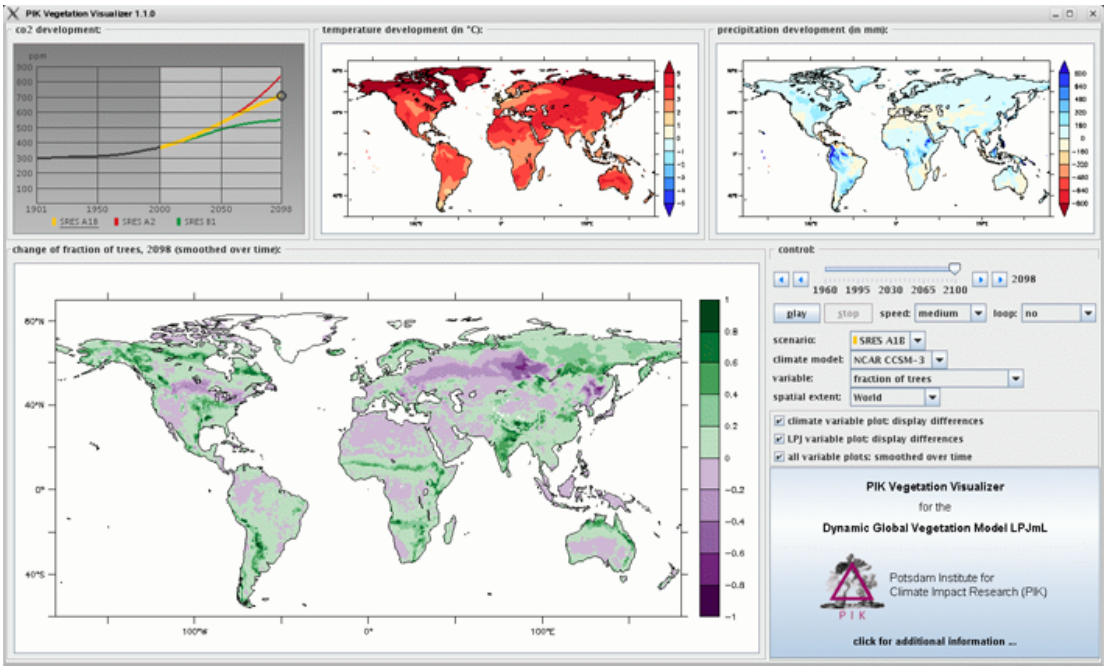



Abbildung 3.17.: Potsdam Vegetation Visualizer (aus [NHP+09]). Ansicht der Desktopapplikation.

Eine weitere Aufgabe der Klimafolgenforschung ist es, sowohl Attribute konkurrierender Modelle der Klimafolgenforschung als auch Attribute eines Modells zu vergleichen.

Für die Anzeige von Informationsrepräsentationen dieser Modelle wurde von Thomas Nocke der *Vegetation Visualizer* [NHP+09] (*PVV – Potsdam Vegetation Visualizer*) entwickelt. Der *PVV* ist eine Desktopapplikation die es erlaubt, interaktiv verschiedene Modelle, Attribute und Ansichten anzuzeigen (siehe Abbildung 3.17). Allerdings ist die Anzahl der gleichzeitig vergleichbaren Ansichten streng begrenzt. Um dies zu erweitern wurde der *PVV* für die Nutzung im Rahmen des *Smart View Managements* angepasst. Im Folgenden werden die einzelnen Schritte dieser Anpassung vorgestellt.

**Bereitstellen der Views**

Für die *Erzeugung der Views* wurde die API zur *View* Erzeugung in den *PVV* integriert. Dadurch kann das *PVV* eine *View* generieren, ohne dass zusätzlich ein *View*

*Grabber* für das Abgreifen der Bildinformationen verwendet werden muss. Weiterhin ist die API damit in der Lage, multiple *Views* simultan zu generieren (ohne diese in der Desktopapplikation anzeigen zu müssen) und damit eine zentrale Anforderung der Klimafolgenforschung zu unterstützen. Um das Erzeugen multipler *Views* zu erleichtern, wurde die GUI des *PVV* um einen Button erweitert. Mit Druck auf den Button wird aus der aktuellen Ansicht (dargestelltes Klimafolgenmodell, Attribut, Bezugsraum) eine separate *View* generiert.

Durch die API wird außerdem eine erweiterte *View Description* ermöglicht (vgl. Abschnitt 3.3). Für die *Views*, erzeugt durch den PVV, wurden folgende Punkte in die *View Description* aufgenommen:

**Klimafolgenmodell** Gibt den Namen des Modells an, auf dessen Basis die visuelle Repräsentation erzeugt wurde.

**Attribut** Gibt den Namen des dargestellten Attributs an.

**Bezugsraum** Gibt den Raum an hinsichtlich dessen die Daten angezeigt werden. Im *PVV* wird dabei zwischen den einzelnen Kontingenten (bzw. der Welt) als Bezugsraum unterschieden.

Auf Grundlage dieser zusätzlichen *View Descriptions* wurde die *View Package Generierung* um einen automatischen Schritt erweitert. Bei der Analyse der Klimafolgenforschungsmodelle treten zwei grundsätzliche Aufgaben auf: (1) Vergleich desselben Attributs eines unterschiedlichen Klimafolgenmodells und (2) Vergleich verschiedener Attribute desselben Klimafolgenmodells.

Daher werden die *View Packages* für diese Aufgaben automatisch zusammengestellt:

- *Views* mit demselben Attribut und unterschiedlichem Klimafolgenmodell.
- *Views* mit unterschiedlichen Attributen desselben Klimafolgenmodells.

Diese automatisch zusammengestellten *View Packages* werden in der GUI zur interaktiven *View Package* Generierung angezeigt. Dort können sie interaktiv durch den Nutzer modifiziert werden. Die *View Description* für diese *Views* wird ebenfalls automatisch erstellt. Da bei beiden Aufgaben ein Vergleich der *Views* vorgenommen werden soll, handelt es sich um die Aufgabe *Vergleichen*.

Abbildung 3.18.: Photo des Smart Meeting Rooms (aus [RNS11]). Verschiedene *Views* generiert durch den angepassten *Vegetation Visualizer* werden auf den Displayflächen dargestellt.

Da *PVV* für die *View* Generierung sowohl für die Präsentation durch einen Presenter als auch für die Diskussion zwischen verschiedenen Nutzern verwendet werden kann, wird das Szenario interaktiv gewählt.

Die Anzeige der *Views* wird durch den Einsatz von *PVV* zur *View* Generierung nicht verändert. Durch den *Smart Display Mapper* werden die *Views* den Displayflächen zugeordnet (vgl. Abschnitt 3.5) und auf den Displayflächen durch einen Layout Algorithmus angeordnet (vgl. Abschnitt 3.6). Das Resultat der Anzeige zeigt Abbildung 3.18.

Durch den Einsatz der API zur *View* Erzeugung wird es ermöglicht, mit nur einer Applikation, die für den Einsatz auf einem einzelnen Gerät konzipiert wurde, eine Vielzahl von *Views* zu generieren und anzuzeigen. Somit bietet dieses Anwendungsbeispiel außerdem eine generelle Vorgehensweise für die Nutzung von Applikationen für die *View* Generierung und die Anzeige im *Smart View Management*.

## 3.8. Zusammenfassung

In diesem Kapitel wurde das *Smart View Management* vorgestellt, ein Konzept zur Bildbasierten Informationspräsentation in Smart Meeting Rooms. Im *Smart View Mana-*

*gement* werden die grundlegenden Probleme der Informationspräsentation in Smart Meeting Rooms adressiert – das Bereitstellen der Informationen und die automatische Anzeige der Informationen.

Für das Bereitstellen der Informationen werden zuerst *Views* – die visuellen Ausgaben der Applikationen – erzeugt. Es werden zwei Optionen der *View* Erzeugung bereitgestellt: *ad-hoc View Erzeugung* und die *API View Erzeugung*. Dies ermöglicht sowohl die sofortige Nutzung unveränderter Software als auch die Anpassung von Software zur *View* Generierung.

Die *ad-hoc View Erzeugung* entspricht im Allgemeinen der technischen Umsetzung der Bildbasierten Informationsdarstellung aus der aktuellen Literatur (vgl. Abschnitt 2.2.2.3). Ein gravierender Unterschied ist dabei allerdings, dass bei der Bildbasierten Informationsdarstellung im Allgemeinen die gesamte Displayfläche des persönlichen Geräts bereitgestellt wird. Die *ad-hoc Erzeugung* liefert nicht nur die Möglichkeit Ausschnitte zu definieren, sondern zusätzlich die gesamte Fläche in separaten Teilen bereitzustellen. Diese Art der Bereitstellung für die Informationsanzeige ist in der gängigen Literatur nur aus Betriebssystemspezifischer Informationsdarstellung bekannt (z.B. [TMC04]). Dies gilt auch für das Bereitstellen von *Views* ohne die zusätzliche Anzeige auf dem persönlichen Gerät. Im *Smart View Management* wird dies durch die *API View Erzeugung* ermöglicht, wobei aber keine Beschränkung auf ein Betriebssystem stattfindet.

Die *Views* werden im nächsten Schritt durch *View Descriptions* beschrieben. Für die Definition der Zusammengehörigkeit von *Views* werden außerdem *View Packages* interaktiv durch Nutzer erzeugt und mit *View Package Descriptions* charakterisiert.

In der gängigen Literatur findet sich bisher keine konzeptionelle Betrachtung der zu übertragenden Bilddaten. Dies wird durch das Konzept der *Views*, *View Descriptions*, *View Packages* und *View Package Descriptions* adressiert. Dadurch wird es ermöglicht, *Views* automatisch anzuzeigen (Zuweisen zu einer Displayfläche und Anordnen auf der Displayfläche).

In weiteren Arbeiten würde eine weiterführende Beschreibung des Inhalts einer *View* dazu beitragen, die Automatismen der Anzeige noch mehr zu verbessern und auszubauen. Ein erster Ansatz für die inhaltliche Beschreibung wurde bei der An-

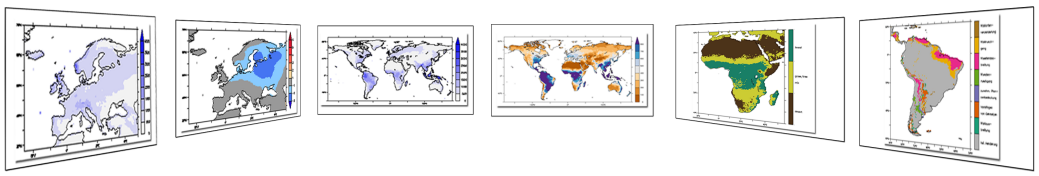
wendung des *Smart View Managements* für die Klimafolgenforschung gezeigt (siehe Abschnitt 3.7). Hier wurden allerdings lediglich spezifische Anforderungen der Anwendung umgesetzt. Eine allgemeinere Betrachtung dieses Problems der Inhaltsbeschreibung wäre für weitere Arbeiten lohnenswert.

Für die Übertragung werden die *Views* mittels *JPEG2000* enkodiert, wodurch die *Views* in eine hierarchische Multilevel Repräsentation überführt werden. Dies ermöglicht eine flexible Dekodierung der anzuzeigenden *Views*. So kann z.B. die Auflösung einer *View* an die zur Verfügung stehende Displayfläche angepasst werden. Ansichten werden in der Regel von Applikationen so erzeugt, dass sie alleine auf einer Displayfläche angezeigt werden können. Durch die gemeinsame Anzeige multipler *Views* auf einer Displayfläche steht so viel Platz aber nicht zur Verfügung. Daher ist das typische Vorgehen im *Smart View Management*, *Views* zu verkleinern. Diese skalierten *Views* können durch die flexible Dekodierung von *JPEG2000 Views* erzeugt werden.

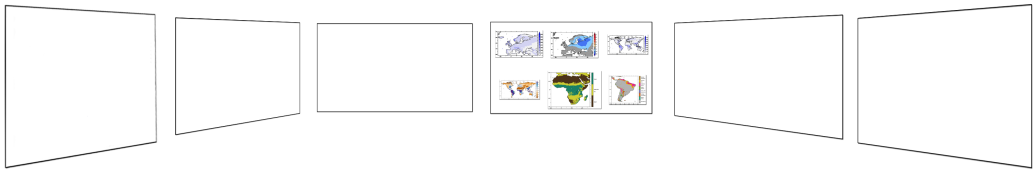
Im nächsten Schritt werden die *Views* durch das *Smart Display Mapping* automatisch den Displayflächen zugewiesen. Auf Grundlage der *View Packages* und der *View Package Descriptions* wird eine automatische Zuweisung von *Views* zu Displayflächen berechnet. Dieses *Smart Display Mapping* basiert auf dem *Display Mapping* von Heider et al. [Hei09, HK08]. Das *Display Mapping* wurde allerdings entscheidend erweitert: (1) die Sichtbarkeitsberechnung wurde dahingehend erweitert, dass auch das Sichtfeld der Nutzer berücksichtigt wird, (2) die Nutzung multipler *Views* auf einer Displayfläche und (3) die Nutzung der *View Packages* und *View Package Descriptions* für die Steuerung des *Display Mappings* wurden ermöglicht.

Dadurch kann nicht nur ein Dokument pro Displayfläche zugewiesen werden, sondern eine Vielzahl *Views* von verschiedenen Applikationen und Geräten (vgl. Abbildung 3.19). Auch das *Smart Display Mapping* profitiert von einer inhaltlichen Beschreibung der *Views*. So könnten weitere Zusammenhänge der *Views* untereinander bei der Anordnung automatisch berücksichtigt werden.

Weiterhin werden die *Views* auf den Displayflächen durch automatische Layout Verfahren dargestellt. Dazu wurden zwei Layout Mechanismen entwickelt: ein *Federkraftbasierter* und ein *Gitterbasierter Layout* Ansatz. Zusätzlich zum automatischen Layout bietet dabei das *Federkraftbasierte Layout* die Möglichkeit, einen präsentierenden Nut-



(a) Eine *View* pro Displayfläche.



(b) Multiple *Views* auf einer Displayfläche.

Abbildung 3.19.: Im *Display Mapping* von Heider et al. kann eine Darstellung einer Displayfläche zugewiesen werden (a). Durch das *Smart View Management* können multiple *Views* von multiplen Applikationen auf einer Displayfläche angezeigt werden (b). Die verbleibenden Displayflächen können für die Anzeige weiterer *Views* verwendet werden.

zer beim Layout der *Views* zu berücksichtigen. Dazu wird die von dem Presenter verdeckte Displayfläche berechnet und die Darstellung von *Views* dort verhindert.

Diese beiden Layout Verfahren ermöglichen es, *Views* auf einer Displayfläche automatisch anzuordnen. Für diese Anordnung ist im Vorfeld kein Template notwendig, das von einem Designer erstellt wurde. Durch die flexible Skalierung der *Views* durch die *JPEG2000* Dekodierung können die Auflösungen der *Views* an die Anforderungen der Layouts angepasst werden. Darüber hinaus wird die flexible *JPEG2000* Dekodierung auch für eine interaktive Anpassung der *Views* genutzt. So können z.B. Regionen von Interesse ausgewählt und in höherer Auflösung dargestellt werden. Die Nutzung dieser Dekodierung für die interaktive Anpassung wird in Kapitel 4 adressiert.

Das *Smart View Management* ermöglicht es durch die flexible Bereitstellung und die automatische Anzeige der *Views*, von multiplen Geräten und multiplen Applikationen von verschiedenen Nutzern *Views* zu generieren und diese automatisch anzuzeigen. Dabei werden Veränderungen dynamisch berücksichtigt und die Anzeige dementsprechend dynamisch angepasst.

# 4. Interaktion

Die Generierung und Anzeige von *Views* in Smart Meeting Rooms übernimmt das *Smart View Management* (vgl. Kapitel 3). Für das Verständnis der damit dargestellten Informationen ist die Interaktion mit diesen Informationen von entscheidender Bedeutung [KHG03, TC05, YKSJ07], denn jede statische Repräsentation wird mächtiger durch das Hinzufügen interaktiver Elemente [DE98].

Deshalb wird in diesem Kapitel die Interaktion im *Smart View Management* vorgestellt. Nachdem der aktuelle Stand der Forschung präsentiert wird (siehe Abschnitt 4.1), werden Anforderungen und Problemstellungen für die Interaktion mit *Views* in Smart Meeting Rooms identifiziert (siehe Abschnitt 4.2.1). Anschließend wird das *Smart Interaction Management* vorgestellt, ein Konzept zur nahtlosen Interaktion mit allen dargestellten *Views* unabhängig vom genutzten Interaktionsgerät (siehe Abschnitt 4.2).

Das *Smart Interaction Management* wird zunächst als technische Basis vorgestellt. Danach wird in Abschnitt 4.3 die Interaktion bezogen auf die Anzeige der *Views* präsentiert. Dazu wird zuerst eine Problembeschreibung erarbeitet (siehe Abschnitt 4.3.1). Folgend werden dann Techniken für die interaktive Adaption der Anzeige vorgestellt.

Der Abschnitt 4.4 beschäftigt sich mit der Interaktion zur Generierung der *Views*. Dazu werden interaktive Methoden zur Anpassung der Informationsdarstellungen in den *Views* vorgestellt. Auch hier werden zunächst die Anforderungen analysiert und Probleme beschrieben. Anschließend werden Methoden zur Unterstützung der Wahrnehmbarkeit dargestellter Informationen im *Smart View Management* präsentiert.

Im Folgenden wird der aktuelle Stand der Forschung für die Interaktion in Smart Meeting Rooms dargelegt.

## 4.1. Interaktion in Smart Meeting Rooms

Systeme für die Informationsanzeige in Smart Meeting Rooms bzw. Multi-Display-Umgebungen sind mit einem grundsätzlichen Problem konfrontiert: Wie können Nutzer mit den dargestellten Informationen über multiple Displayflächen hinweg interagieren, die an verschiedene Computer angeschlossen sind? Dafür gibt es zwei Alternativen.

Eine Lösung ist, Maus- und Tastatureingaben nicht nur für ein Gerät zur Verfügung zu stellen, sondern diese Interaktionen auf die verschiedenen Geräte umzulenken. Dadurch wird es ermöglicht, mit Maus und Tastatur eines Nutzers auf allen Displayflächen und mit allen dargestellten Informationen zu interagieren.

Eine weitere Lösung ist das Entwerfen und Einbinden neuer Interaktionsgeräte. Dafür werden Interaktionsgeräte benutzt, die eine den Gegebenheiten und Möglichkeiten des Raumes angepasste Interaktion ermöglichen.

Im Folgenden werden zunächst verschiedene Lösungsstrategien für die Nutzung von Maus und Tastatur in Multi-Display-Umgebungen bzw. Smart Meeting Rooms und danach Ansätze für neue Interaktionsgeräte vorgestellt.

### 4.1.1. Maus- und Tastaturlösungen

Bei Maus- und Tastaturlösungen besteht die Hauptaufgabe darin, den Zeiger einer Maus, die an ein persönliches Gerät (z.B. Laptop) angeschlossen ist, auf die Displayflächen der Umgebung umzuleiten, so dass dort interagiert werden kann. Zu lösende Teilprobleme sind dabei jeweils: (1) wie wird die Interaktion auf ein anderes Gerät geschickt und dort ausgewertet und (2) wie ist die Anordnung der Displayflächen im Raum (auf welcher Displayfläche wird der Zeiger angezeigt, wenn eine Displayfläche an einer Seite verlassen wird)?

Im Folgenden werden prominente Vertreter solcher Systeme vorgestellt.

**PointRight**

*PointRight* [JHW00] wurde für die Interaktion mit persönlicher Maus und Tastatur in einem Smart Meeting Room, dem iRoom [JFW02], entwickelt. Dafür wurde ein Server-Client System geschaffen, das es erlaubt, persönliche Geräte als Quelle und jede vorhandene Displayfläche als Ziel einer Interaktion zu nutzen.

Wird ein *PointRight* Server auf einem persönlichen Gerät gestartet, können Maus und Tastatur im Raum verwendet werden. Auf jedem der Computer, der eine Displayfläche ansteuert, läuft dabei ein *PointRight* Client. Der Client empfängt die Interaktionen und wertet sie aus. Tastatureingaben werden an dem Gerät ausgeführt, an dem der Zeiger sich momentan befindet. Die Events, die sich aus Maus- und Tastatureingaben ergeben, werden mit Hilfe der Middleware (*iROS* [BRTF02]) verteilt und ausgewertet. Dabei ist *PointRight* in der Lage, multiple Zeiger auf einer Displayfläche darzustellen, indem sogenannte *ghost Pointer* für jeden Zeiger gemalt werden. Ausgeführte Interaktionen verschiedener Nutzer werden dann nacheinander ausgewertet. Dabei entstehende Konflikte werden nicht behandelt. Diese Server-Client Architektur wurde auf Windows PCs implementiert.

Die Anordnung der Displayflächen erfolgt für *PointRight* über eine Konfiguration, die in einer Datenbank gespeichert ist. Diese Konfiguration wird zu den *PointRight* Servern beim Start übertragen. Aus diesem Grund ist es mit *PointRight* auch möglich, für jede einzelne Maus und Tastatur eine eigenständige Anordnung zu verwenden.

**Clicky**

Auch *Clicky* [ASWC04] stellt die Interaktion mit Maus und Tastatur auf allen Displayflächen im Raum zur Verfügung. Neben der Unterstützung dieser klassischen Interaktionsgeräte sieht die Anforderung an *Clicky* ebenfalls vor, andere Interaktionsgeräte zu unterstützen, wie z.B. PDAs oder Ubisense Tags. Dabei setzen Andrews et al. auch eine Server - Client Architektur ein. Hier kommen *endpoint* für die Displayflächen und *redirector* für die persönlichen Geräte zum Einsatz. Über genauere Details zur Umsetzung werden allerdings keine Angaben gemacht.

Die relative Anordnung der Displayflächen zueinander wird durch einen graphischen Editor interaktiv manipuliert.

### Deskotheque

Bei der Maus- und Tastaturverteilung gehen Waldner et al. im Rahmen der *Deskotheque* noch einen Schritt weiter [WPKS10, WKS10, Wal11, WS10]. Neben dem Bereitstellen der persönlichen Maus und Tastatur für alle Displayflächen und der Anordnung der Displayflächen anhand eines geometrischen Modells wird hier auch das Verhalten des Zeigers bei der Navigation thematisiert.

Das *Deskotheque* Framework zur Informationsanzeige basiert auf einem Linux mit spezifischer XWindow Implementierung. Dadurch ist es möglich, für das Umleiten der Maus- und Tastatureingaben eine modifizierte Version der Open-Source-Software *Synergy* [Syn13] zu nutzen. *Synergy* erlaubt es, über verschiedene Plattformen hinweg Maus- und Tastaturinputs zu verteilen. Im Rahmen der *Deskotheque* wurden zwei Erweiterungen implementiert. Zum einen wurde die Handhabung multipler Mäuse und Tastaturen und multipler Zeiger implementiert und durch Compiz [Com13], ein Fenster Manager Plugin, gerendert. Des Weiteren wurde das Modell für die relative Anordnung der Fenster zueinander durch das geometrische 3D Modell der *Deskotheque* [WPS08] erweitert.

Im Unterschied zu anderen Ansätzen wird im Rahmen der *Deskotheque* ein dreidimensionales Modell der Displayanordnung genutzt. Diese Anordnung wird durch ein Kamerasystem automatisch erfasst und erstellt, wodurch auch Änderungen sehr schnell im Modell berücksichtigt werden können.

Diese Anordnung lässt es auch zu, die Zeigerbewegung der Maus bei der Navigation für einen einzelnen Nutzer anzupassen. So wird auf Basis der Position des Nutzers eine perspektivisch korrekte Navigation ermöglicht. Durch die dreidimensionale Anordnung geht dieser Ansatz auch einen Schritt weiter als der Ansatz des *Perspective Cursor* [NSC+06], bei dem ausgehend vom Nutzer lediglich eine zweidimensionale Anordnung erstellt wird. Weiterhin stellen Waldner et al. zwei Arten der Navigation vor: *free navigation* und *path navigation*. Bei der *free navigation* wird ein Zeiger mit perspektivisch korrekter Bewegung angezeigt. Die Freiräume zwischen den Displayflächen werden mit einem perspektivischen *warping* überbrückt. Bei der *path navigation* werden zunächst die kürzesten Kanten zwischen den Displayflächen berechnet. Verlässt ein Zeiger eine Displayfläche, wird er zur nächstgelegenen Displayfläche ge-

warpt. Die Bewegungen auf den Displayflächen verlaufen wiederum perspektivisch. Dadurch werden nicht nur Konzepte wie das *pointer warping* [BCHG04, BF07] mit der perspektivisch korrekten Navigation [NSC+06] vereint, sondern darüber hinaus durch das dreidimensionale Modell eine freie Anordnung der Displayflächen für eine perspektivisch korrekte Navigation ermöglicht.

### 4.1.2. Neue Interaktionsgeräte

Die Entwicklung und der zielgerichtete Einsatz neuer Interaktionsgeräte wird im Bereich der Multi-Display-Umgebungen bzw. Smart Meeting Rooms in der Literatur ebenfalls thematisiert.

Neue Interaktionsgeräte werden dann verwendet, wenn es ermöglicht werden soll, statt eines relativen Zeigers das direkte Zeigen zu unterstützen. Bei relativen Zeigern muss die gesamte Fläche der Displayflächen überbrückt werden, die zwischen den verschiedenen Informationen liegt. Gerade bei vielen Displayflächen, die weit auseinander stehen, bedarf es dabei eines sehr weiten Weges. Beim direkten Zeigen entfällt dies, da mit einer kurzen Handbewegung der gesamte Raum überbrückt ist. Vor allem im Bereich der Large High Resolution Displays[1] (LHRD) stehen diese Arten der Interaktion im Fokus der Forschung, da im Umfeld der LHRD die Nutzung von Maus und Tastatur ineffizient ist [FTS+10, OS02].

Eine Technik für die direkte Interaktion ist die Nutzung von Laserpointern. Diese Interaktion wird in der Literatur umfangreich behandelt. [Pec01, ATK+05, FTS+10]. Der *XWand* [WS03, WP03] nutzen das gleiche Prinzip des direkten Zeigens, erweitern aber die Mächtigkeit des Interaktionsgeräts durch zusätzliche Funktionen. Durch Sensoren (Gyroskopische-, Magetometrische- und Beschleunigungssensoren) und zusätzliche Knöpfe am Interaktionsgerät selbst werden Freiheitsgrade und Leistungsfähigkeit wesentlich erhöht.

Durch die Einführung der *Nintendo Wii* und dem dazugehörigen *Wiimote*[2] im Jahr 2006 [Nin13] wurde solch ein Interaktionsgerät für das direkte Zeigen mit zusätzlichen

[1] große hochauflösende Displays
[2] *Wiimote*: kurz für Wii Remote, bzw. Wii Fernbedienung

Freiheitsgraden für einen geringen Preis als Standardhardware verfügbar [SWMB09]. Von Johnny Chung Lee wurde außerdem gezeigt, dass sehr viel mehr Potential für die Interaktion in der *Wiimote* steckt [Lee13]. So entwickelte er auf Basis der Standardhardware ein Fingertracking, ein interaktives Whiteboard mit Multipointer und Headtracking für ein Desktop Virtual Reality Display.

Der Nachteil von direktem Zeigen ist die Ungenauigkeit, welche hauptsächlich vom Zittern der Hand ausgelöst wird [Pec01, MBN+02, ST06b, WP03]. Um diesen Nachteil bei der Interaktion zu reduzieren, wird z.B. der reale durch einen geglätteten Zeiger ersetzt. Darüber hinaus präsentieren Lehmann et al. einen Ansatz für ein distanzabhängiges Mapping der Interaktion. Dabei wird abhängig von der Distanz des Nutzers zur Displaywand die automatische Genauigkeit des Zeigers eingestellt [LS12].

Auch die Interaktion ohne zusätzliche Interaktionshardware durch Gestensteuerung wird ausführlich in der Literatur thematisiert. Eine Übersicht liefern dazu Karam et al. [KS05].

### 4.1.3. Zusammenfassung

Für die Interaktion in Smart Meeting Rooms über multiple Displayflächen hinweg werden in der Literatur diverse Lösungsansätze angegeben. Diese lassen sich in die Interaktion mit *Maus und Tastatur* und die Nutzung *neuer Interaktionsgeräte* einteilen.

Bei der Interaktion mit Maus und Tastatur werden in der Regel die persönlichen Geräte der Nutzer auf die anderen Displayflächen umgelenkt. Durch die positionsabhängige Navigation wird der Zeiger der Anordnung der Displayflächen angepasst, so dass ein komfortables Arbeiten ermöglicht wird.

Bei der Interaktion mit neuen Interaktionsgeräten werden in der Regel Lösungen für spezifische Anwendungen entwickelt. Auch die eigens entwickelte Interaktionshardware wird dabei an die spezifischen Aufgaben angepasst. Diese Verbindung von physischer Interaktion mit der semantischen Interpretation verstärkt die Mächtigkeit der Interaktionsgeräte. Beispielsweise werden beim *UIC* (Universal Interaction Controller) [ST06b, ST06a] spezifische Interaktionen für das Kopieren, Ausschneiden und Einfügen bereitgestellt.

Ein Konzept für die Interaktion mit allen Displayflächen, welches in der Lage ist sowohl Maus und Tastatur als auch neue Interaktionsgeräte zu nutzen, wurde bisher in der gängigen Literatur nicht vorgestellt. Dies wäre in Smart Meeting Rooms eine sinnvolle Lösung. So könnte z.B. der Presenter durch neue Interaktionsgeräte unterstützt werden, während bei einer Diskussion die Teilnehmer mit Maus und Tastatur mit den verschiedenen *Views* auf verschiedenen Displayflächen interagieren.

Durch den Einbezug jeglicher Interaktionsgeräte wird es dem Nutzer ermöglicht, die bevorzugten und ihm vertrauten Interaktionsgeräte einzusetzen.

## 4.2. Smart Interaction Management

### 4.2.1. Problembeschreibung

In Smart Environments und speziell in Smart Meeting Rooms ist die ständige Interaktion des Menschen mit dem Computer ein integraler Bestandteil [CD04]. Aber im Hinblick auf die von Mark Weiser formulierte Vision der verwobenen Sensoren, Aktuatoren, Displays und rechnenden Elemente in einem Smart Environment [Wei91] wird die Interaktion zunehmend komplexer. Heider und Kirste geben in [HK05b] weiterhin zu bedenken, dass bezüglich der Vision von Don Norman [Nor99] – dem ´´The Invisible Computer" – das Problem der Interaktion noch komplexer wird. Sie werfen die Fragen auf: (1) wie man mit smarten Dingen interagiert, die man nicht kennt, (2) wie man Geräte steuert, die man nicht sieht und (3) wie man diese Interaktionen darüber hinaus in einer dynamischen Umgebung realisiert.

Mit dem *Smart View Management* als Grundlage für die Informationsdarstellung in Smart Meeting Rooms treten für die Interaktion diese von Heider und Kirste in [HK05b] beschriebenen Probleme auf. Durch die dynamische Erzeugung von *Views* an verschiedenen Geräten und Applikationen und durch die dynamische Anzeige der *Views* auf multiplen Displayflächen ist es für den Nutzer nicht ersichtlich, wo welche *View* generiert wurde. Weiterhin ist es für den Nutzer nicht offensichtlich, welche Geräte die Displayflächen ansteuern, die diese *Views* anzeigen. Daraus resultierend werden an die Interaktion in Smart Meeting Rooms folgende Anforderungen gestellt:

**Manipulierbarkeit aller *Views* auf jeder Displayfläche mit jedem Interaktionsgerät** Die grundlegende Anforderung für die Interaktion mit den *Views* des *Smart View Managements* ist daher, dass jede dargestellte Information auf jeder Displayfläche mit jedem Interaktionsgerät manipulierbar sein sollte [JHW00, NSC+06]. Dadurch wird außerdem ermöglicht, dass Nutzer nicht zwischen verschiedenen Interaktionsgeräten wechseln müssen, um mit Darstellungen auf anderen Displayflächen zu interagieren. Hierbei sollte der Übergang von einer Displayfläche zur anderen nahtlos sein [NSC+06]. Zudem kann der Nutzer Interaktionsgeräte je nach zu erledigender Aufgabe auswählen und einsetzen [FFH00].

**Unterstützen persönlicher Interaktionsgeräte für die Interaktion mit *Views*** Diese Anforderung ergibt sich aus der Anwendung im Smart Meeting Room, welche als ad-hoc Umgebung zur Unterstützung persönlicher Geräte konzipiert wurde. Das bedeutet im Kontext der Interaktion, dass persönliche Interaktionsgeräte der Nutzer für die Interaktion mit *Views* unterstützt werden sollten. Die Nutzung persönlicher Interaktionsgeräte ermöglicht es ebenfalls, dass Nutzer mit Interaktionsgeräten arbeiten, mit deren Eigenschaften sie vertraut sind, was wiederum eine erfolgreiche Interaktion ermöglicht [FFH00].

**Unterstützen neuer Interaktionsgeräte** Neben den klassischen Interaktionsgeräten wie z.B. Maus und Tastatur sollten auch neue Interaktionsgeräte wie z.B. die *Wiimote* für die Interaktion unterstützt werden.

**Die "Unsichtbarkeit" der Interaktion sollte unterstützt werden** Die Nutzer sollen den Ablauf der Interaktion nicht verstehen müssen, sondern ein ihnen vertrautes Interaktionsgerät ad-hoc für die Interaktion mit allen *Views* auf allen Displayflächen im Raum nutzen können [RC00].

### 4.2.2. Prinzipieller Lösungsansatz

Die in dieser Arbeit verfolgte Lösungsidee für die Interaktion mit *Views* im Smart Meeting Room ist die Entkopplung der Interaktionsgeräte (z.B. Maus, Tastatur, Wiimote) von der Interpretation der Interaktion. Durch diese Entkopplung wird erreicht, dass

eine physisch ausgeführte Interaktion an einem beliebigen Gerät des Smart Meeting Room Ensembles unabhängig vom Interaktionsgerät semantisch interpretiert werden kann.

Um die genannten Anforderungen an die Interaktion im Smart Meeting Room zu lösen, wurde das Konzept des *Smart Interaction Managements* entwickelt. Durch das *Smart Interaction Management* werden die drei grundsätzlichen Anforderungen adressiert: (1) Interaktion mit allen dargestellten *Views* auf allen Displayflächen mit allen Interaktionsgeräten, (2) ad-hoc Nutzung persönlicher Interaktionsgeräte, (3) Einbindung neuer Interaktionsgeräte und (4) keine Notwendigkeit der Kenntnis des Ablaufs der Interaktion beim Nutzer.

Die Entkopplung der Interaktionsinterpretation von der Interaktionshardware basiert auf der Trennung der Interaktion in physische und logische Interaktion nach de Melo [Mel10]:

**Physische Interaktion** Interaktion des Nutzers mit Ein-/Ausgabegeräten (Interaktionsgeräten), wie z.B. das Drücken einer Maustaste.

**Logische Interaktion** Interaktion, die ein Nutzer mit dem System zur Änderung des Systemzustands vornehmen will (die Intention des Nutzers).

Dix et al. [DFAB03] unterteilen die logische Interaktion weiter in drei Ebenen:

**Syntaktische Ebene** Beschreibung von Ordnung (Reihenfolge) und Struktur der Interaktion.

**Lexikalische Ebene** Beschreibung der Repräsentation von Elementen der Interaktion.

**Semantische Ebene** Wirkung von Interaktionen auf die internen Datenstrukturen des Systems.

Basierend auf dieser Trennung (siehe Abbildung 4.1) ergeben sich für eine nahtlose Interaktion auf allen Displayflächen mit beliebigen Interaktionsgeräten vier grundlegende Problemstellungen:

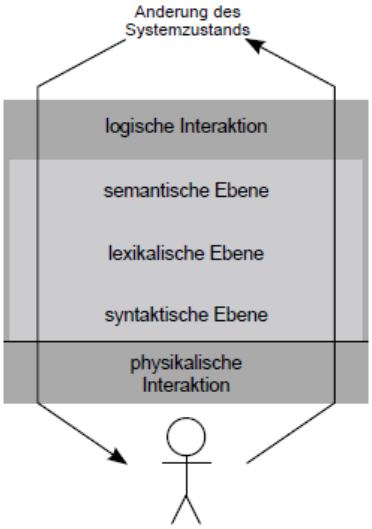



Abbildung 4.1.: Trennung von physikalischer und logischer Interaktion aus [Mel10]

1. Überführen der physikalischen Interaktion in die syntaktische Ebene der logischen Interaktion.
2. Auswertung der syntaktischen Interaktion und Überführen in die lexikalische Ebene.
3. Zuweisen der Interaktion zur Zieldisplayfläche.
4. Interpretation der Interaktion durch Überführung der Interaktion von der lexikalischen Ebene in die semantische Ebene.

Der erste Punkt, die Überführung der physikalischen Interaktion in die syntaktische Ebene der logischen Interaktion, wird vom Betriebssystem und dem verwendeten Treiber übernommen. Der Treiber eines Gerätes stellt die Hardware dem System zur Verfügung, überlässt den Aspekt der Interpretation aber den Applikationen. Damit stellt der Treiber nur die Ressourcen bereit ohne Constraints hinzuzufügen [CRKH09].

Der zweite Punkt ist das Auswerten der syntaktischen Interaktion und Überführen in die lexikalische Ebene. Dazu wird im *Smart Interaction Management* der **Interaction Grabber** verwendet. Bei einer physikalischen Interaktion werden durch den Treiber

Events generiert. Diese Events werden vom *Interaction Grabber* registriert und in eine Zwischenbeschreibung übersetzt. Die Zwischenbeschreibung repräsentiert dabei die lexikalische Beschreibung und damit liegt die Interaktion in lexikalischer Ebene vor. Der *Interaction Grabber* wird spezifisch für verschiedene Interaktionsgeräte angepasst.

Der dritte Punkt ist die Zuweisung der Interaktion zu einer Displayfläche. Dazu wird im *Smart Interaction Management* der **Interaction Mapper** verwendet. Hier wird die Interaktion in lexikalischer Ebene gesammelt. Diese Interaktion wird dann dem *Metarenderer*[3] übertragen, der auf der korrespondierenden Displayfläche läuft.

Der vierte Punkt ist die Interpretation der Interaktion, d.h. Übertragung der lexikalischen Interaktion in die semantische Interaktion. Dafür wird im *Smart Interaction Management* der **Interaction Handler** verwendet. Der *Interaction Handler* übersetzt die Interaktion in lexikalischer Ebene in vom Zielsystem interpretierbare Interaktionen und führt sie aus. Dabei wird unterschieden zwischen der Interaktion mit der Anzeige der *Views*– eine Interaktion zur Modifikation der Zuweisung der *Views* zu den Displayflächen und dem Layout der *Views* auf den jeweiligen Displayflächen – und der Interaktion mit dem Inhalt der *Views* – einer Interaktion zur Manipulation der dargestellten Information.

Somit ergibt sich zusammenfassend der folgende generelle Ablauf des *Smart Interaction Managements*: Die physische Interaktion wird vom *Interaction Grabber* ausgewertet. Der *Interaction Grabber* stellt dabei die Verbindung der Interaktionshardware zum *Smart Interaction Management* her. Der *Interaction Grabber* übersetzt die Interaktion von der syntaktischen in die lexikalische Ebene, d.h. der Input des *Interaction Grabbers* ist eine Interaktion in syntaktischer Ebene und der Output ist die Interaktion in lexikalischer Ebene. Der nächste Schritt ist der *Interaction Mapper*. Hier wird die Interaktion einer Displayfläche mit korrespondierendem *Metarenderer* zugewiesen und per Netzwerk übertragen. In- und Output sind jeweils in lexikalischer Ebene. Im nächsten Schritt wird im *Metarenderer* der *Interaction Handler* verwendet, um die Interaktion von der lexikalischen in die semantische Ebene zu übersetzen. An dieser Stelle ist die Interpretation der Interaktion notwendig, um entscheiden zu können, ob eine Interaktion mit der Anzeige oder dem Inhalt der *Views* ausgeführt werden soll. Bei einer

[3] Der *Metarenderer* ist die Softwarekomponente des *Smart View Managements*, die das Anordnen und die Anzeige der *Views* auf den jeweiligen Displayflächen ermöglicht.

Interaktion mit der Anzeige der *Views* wird die Interaktion am *Metarenderer* interpretiert und die Änderungen werden ausgeführt. Bei einer Interaktion mit dem Inhalt der *Views* wird ermittelt, mit welcher *View* interagiert werden soll und von welchem Gerät die *View* bereitgestellt wird. Dann wird die Interaktion in lexikalischer Ebene an das entsprechende Gerät versendet, dort durch einen *Interaction Handler* interpretiert und dementsprechend Aktionen auf Ebene der *View* generierenden Applikation ausgelöst. Damit kann sowohl der Presenter als auch jeder Nutzer von seinem persönlichen Gerät in die Generierung einer beliebigen *View* und damit in die Erzeugung des dargestellten Inhalts eingreifen.

Im Folgenden werden die einzelnen Komponenten des *Smart Interaction Managements* im Detail vorgestellt.

### 4.2.3. Interaction Grabber

Der *Interaction Grabber* wertet die syntaktischen Interaktionen aus und überführt sie in die lexikalische Ebene. Somit stellt er die Verbindung zwischen dem verwendeten Interaktionsgerät und dem *Smart Interaction Management* dar. Dafür müssen drei grundlegende Teilaufgaben durchgeführt werden: (1) das Auswerten der syntaktischen Interaktionen, (2) das Übersetzen der Interaktionen in die lexikalische Ebene und (3) das Weiterleiten der Interaktion an den *Interaction Mapper*.

Das **Auswerten der Interaktion** ist die Grundlage für die Nutzung eines Interaktionsgerätes für das *Smart Interaction Management*. Wird eine physische Interaktion von einem Nutzer mit einem Interaktionsgerät ausgeführt, so wird diese vom Treiber des Gerätes erkannt. Durch den Treiber werden die analogen Signale des Interaktionsgeräts (z.B. Drücken einer Maustaste) ausgewertet und in eine logische Interaktion auf syntaktischer Ebene übersetzt. Diese Interaktionen auf syntaktischer Ebene werden dann an das System mittels sogenannter Events gemeldet. Ein Event beschreibt dabei syntaktisch die vom Nutzer ausgeführte Interaktion. Das Auswerten besteht nun darin, diese Events zu erkennen und bei Auftreten eines Events darauf zu reagieren. Um dies zu erreichen, ist für jedes zu verwendende Interaktionsgerät ein *Interaction Grabber* erforderlich.

Nach dem Erkennen eines Events erfolgt das **Übersetzen der Interaktionen in die lexikalische Ebene**. Hierfür können zwei Möglichkeiten einer lexikalischen Beschreibung genutzt werden. Zum einen kann die syntaktische Beschreibung der durchgeführten Interaktion direkt ohne Nutzung einer Zwischenbeschreibung übertragen werden. Zum anderen kann die Interaktion in eine Zwischenbeschreibung übersetzt und anschließend übertragen werden.

Die Option ohne eine Zwischenbeschreibung hat den Vorteil, dass die Realisierung des *Interaction Grabbers* sehr einfach ist. Die Interaktion in syntaktischer Ebene wird erkannt und ohne zusätzlichen Aufwand weitergeleitet. Damit lassen sich *Interaction Grabber* für neue Interaktionsgeräte sehr schnell und einfach implementieren. Eine Übersetzung in die semantische Ebene ist allerdings weiterhin vonnöten. Der Verzicht auf eine Zwischenbeschreibung hat aber den Nachteil, dass der *Interaction Handler* später die Übersetzungsleistung übernehmen muss. Er müsste also eine Übersetzung der spezifischen syntaktischen Beschreibung aller genutzten Interaktionsgeräte in die semantische Ebene implementieren.

Die zweite Option der Nutzung einer Zwischenbeschreibung hat den Vorteil, dass die Übersetzung von Zwischenbeschreibung zu semantischer Ebene für jedes Gerät gleich ist. Daher muss der *Interaction Handler* in diesem Fall lediglich die Übersetzung aus einer festgelegten Beschreibung implementieren, was den Aufwand erheblich reduziert. Ein Nachteil ist aber, dass dann der Aufwand am *Interaction Grabber* steigt, da jetzt dort die Übersetzung implementiert werden muss. Da die Übersetzung der spezifischen syntaktischen Ebene in die Zwischenbeschreibung allerdings nur genau einmal für jedes Interaktionsgerät implementiert werden muss, ist hier der Aufwand geringer.

Deshalb wurde im *Smart Interaction Management* eine Zwischenbeschreibung verwendet. Für die Zwischenbeschreibung gibt es wiederum zwei Möglichkeiten: (1) die Nutzung einer abstrakten Zwischenbeschreibung oder (2) die Nutzung einer konkreten Zwischenbeschreibung.

Der Vorteil einer *abstrakten Zwischenbeschreibung* liegt darin, dass diese Beschreibung generisch aufgebaut werden kann. Dadurch kann sie beliebig erweitert werden und ist damit in der Lage, jedes beliebige Interaktionsgerät mit seinen Freiheitsgraden abzu-

bilden. Der Nachteil ist, dass bei der Übersetzung der Zwischenbeschreibung in die semantische Ebene, wie es an den Applikationen durch den *Interaction Handler* umgesetzt wird, die Struktur der Zwischenbeschreibung bekannt sein muss.

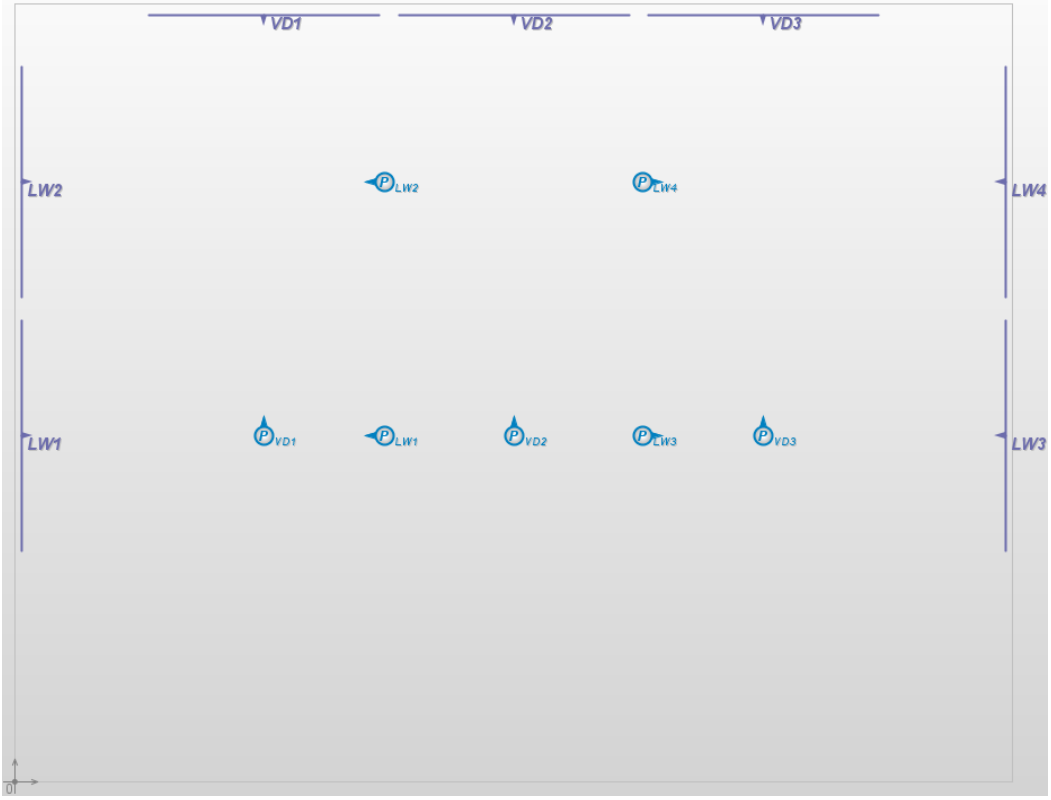
Die *konkrete Zwischenbeschreibung* hat den Vorteil, dass bei richtiger Wahl der Zwischenbeschreibung die Übersetzung in die semantische Ebene sehr stark vereinfacht wird. Wenn eine Applikation bereits die Semantik mittels dieser Beschreibung definiert, ist eine weitere Übersetzung nicht notwendig. Dies reduziert den Aufwand am *Interaction Handler* erheblich. Ein Nachteil der konkreten Zwischenbeschreibung ist, dass sie in der Regel nicht erweiterbar ist. Dadurch ist nicht gesichert, dass die vollständige Funktionalität aller Interaktionsgeräte abgebildet werden kann.

Im *Smart Interaction Management* wird dennoch eine *konkrete Zwischenbeschreibung* verwendet. Hier wird auf die syntaktische Beschreibung von *Maus und Tastatur* Interaktionen zurückgegriffen. Diese Zwischenbeschreibung bietet gleich mehrere Vorteile. Zum einen wird in dieser Beschreibung oft die Semantik der Interaktionen in den Applikationen implementiert. So ist ein weiteres Mapping nicht erforderlich. Zum anderen bieten Maus und Tastatur eine sehr große Mächtigkeit. Daher ist die Zwischenbeschreibung in der Lage, eine Vielzahl verschiedener Interaktionsgeräte abzubilden.

Das Resultat des *Interaction Grabbers* ist die Interaktion, die in lexikalischer Ebene durch eine Zwischenbeschreibung dargestellt ist. Dabei wird Maus- und Tastatursyntax als Zwischenbeschreibung verwendet. Das hat zur Folge, dass die weiteren Schritte, Zuweisung zur Displayfläche und Interpretation der Interaktion, nun für alle Interaktionsgeräte gleich ablaufen und der *Interaction Grabber* als Interface für die Anbindung beliebiger Interaktionsgeräte an das *Smart Interaction Management* fungiert.

### 4.2.4. Interaction Mapper

Ziel des *Interaction Mappers* ist es, eine Interaktion auf lexikalischer Ebene von einem *Interaction Grabber* zu empfangen und an das Zieldisplay und den korrespondierenden *Metarenderer* weiterzuleiten. Damit wird es ermöglicht, über verschiedene Displayflächen hinweg zu interagieren, ohne dass die Displayflächen untereinander jeweils ein Modell des Raums bzw. der Anordnung der Displayflächen besitzen.




(a) Aufriss des Smart Meeting Rooms mit der Positionierung der Displayflächen (LW1 – LW4 und VD1 – VD3).


(b) Projizierte Anordnung der Displayflächen mit relativer Ausrichtung zueinander.

Abbildung 4.2.: Grundlage für den *Interaction Mapper*: die Projektion der Displayflächen in einer zweidimensionalen Ebene zur Ermittlung der relativen Anordnung zueinander.

Um dies zu erreichen benutzt der *Interaction Mapper* zwei Beschreibungen: (1) Verwaltung multipler Zeiger und (2) Anordnung der Displayflächen.

Durch die Verwaltung multipler Zeiger wird es ermöglicht, Eingaben von unterschiedlichen Interaktionsgeräten auf multiplen Displayflächen zu verwalten. Eine Interaktion auf lexikalischer Ebene kann dadurch eindeutig einem Zeiger zugeordnet werden. Dafür registriert sich ein *Interaction Grabber* beim Start beim *Interaction Mapper*. Dem *Interaction Grabber* wird eine eindeutige ID zugeordnet. Der Zeiger, der zu dieser ID gehört, ist damit genau einer Displayfläche (respektive einem *Metarenderer*) zugeordnet. Erfolgt nun eine Interaktion, wird mittels ID festgestellt, auf welcher Displayfläche sich der entsprechende Zeiger befindet. Zu dieser Displayfläche wird dann die Interaktion weitergeleitet.

Die Beschreibung der *Anordnung der Displayflächen* ist notwendig, um den Wechsel eines Zeigers von einer Displayfläche zur Anderen zu ermöglichen. Wie bereits erwähnt wird eine Interaktion einem *Metarenderer* zugeordnet. Dort wird die Interaktion mittels des *Interaction Handlers* ausgewertet. Ein mögliches Resultat dieser Interaktion ist das Verlassen einer Displayfläche an einer Kante. Tritt dieser Fall auf, wird es an den *Interaction Mapper* zurückgemeldet. Mit Hilfe der geometrischen Anordnung wird dann ermittelt, auf welche Displayfläche der Zeiger verschoben werden soll. Der *Interaction Mapper* verändert dann diese Zuordnung. Die folgende Interaktion dieses Interaktionsgerätes wird dann an die neue Displayfläche (bzw. dessen *Metarenderer*) versendet.

Zur Ermittlung der Anordnung der Displayflächen werden in der Literatur verschiedene Möglichkeiten vorgeschlagen (vgl. Abschnitt 4.1). Im Rahmen des *Smart Interaction Managements* wird eine relative Positionierung der Displayflächen verwendet. Eine relative Positionierung bedeutet, dass nicht ein detailliertes geometrisches Modell der Displayflächen angefertigt wird (vgl. [WPS08]), sondern die Nachbarschaften ermittelt werden (z.B. Displayfläche 1 ist links von Displayfläche 2) (vgl. [Syn13]). Zur Ermittlung dieser Nachbarschaften wird eine 2D Projektion der Displayflächen aus dem räumlichen Modell erstellt (siehe Abbildung 4.2). Basierend auf dieser Projektion lässt sich dann die relative Anordnung bestimmen (siehe Abbildung 4.2b).

### 4.2.5. Interaction Handler

Der *Interaction Handler* übernimmt den letzten Schritt im Interaktionsprozess. Seine Aufgabe ist es, die Interaktion auf lexikalischer Ebene (in Zwischenbeschreibung) in die semantische Ebene zu übersetzen. Dabei wird im *Smart Interaction Management* zwischen der Interaktion mit der Anzeige der *Views* und der Interaktion mit dem Inhalt der *Views* unterschieden.

Durch die *Interaktion mit der Anzeige der Views* wird der Nutzer (im Allgemeinen der Presenter) in die Lage versetzt, die Zuordnung zur Displayfläche, das Layout und das Erscheinungsbild der *Views* zu manipulieren. Die Interaktion mit der Anzeige der *Views* wird in Abschnitt 4.3 im Detail besprochen.

Für die *Interaktion mit dem Inhalt der Views* ist es notwendig, dass die *View* generierende Applikation den *Interaction Handler* verwendet. Die Interaktion mit dem Inhalt der *Views* wird in Abschnitt 4.4 im Detail besprochen.

Um zu unterscheiden, welche Art der Interaktion ausgeführt werden soll, erhält jeder Zeiger einen separaten Status. Dieser Status kann durch Druck auf einen Button umgeschaltet werden.

Bei einer Interaktion mit der Anzeige der *Views*, wird die Interaktion semantisch mit Hilfe des *Interaction Handlers* im *Metarenderer* interpretiert. Dies ermöglicht es, das Layout und die Zuordnung der *Views* zu den Displayflächen zu modifizieren.

Wird eine Interaktion mit dem Inhalt der *Views* ausgeführt, wird zunächst detektiert, welche *View* Ziel der Interaktion ist. Dann wird die Interaktion in lexikalischer Ebene (Zwischenbeschreibung) an die *View* generierende Applikation weitergeleitet. Der entsprechende *Interaction Handler* interpretiert dort die Interaktion.

Das Resultat des *Interaction Handlers* ist die Ausführung der Interaktionen an dem vom Nutzer definierten Ziel. So wird es möglich, sowohl die Anzeige als auch das Angezeigte mit einem beliebigen Interaktionsgerät zu manipulieren. Der vollständige Ablauf des *Smart Interaction Managements* ist in Abbildung 4.3 dargestellt.

In den folgenden Abschnitten wird gezeigt, wie auf der Grundlage des SVM eine weitreichende Interaktionsfunktionalität mit den *Views* realisiert werden kann.

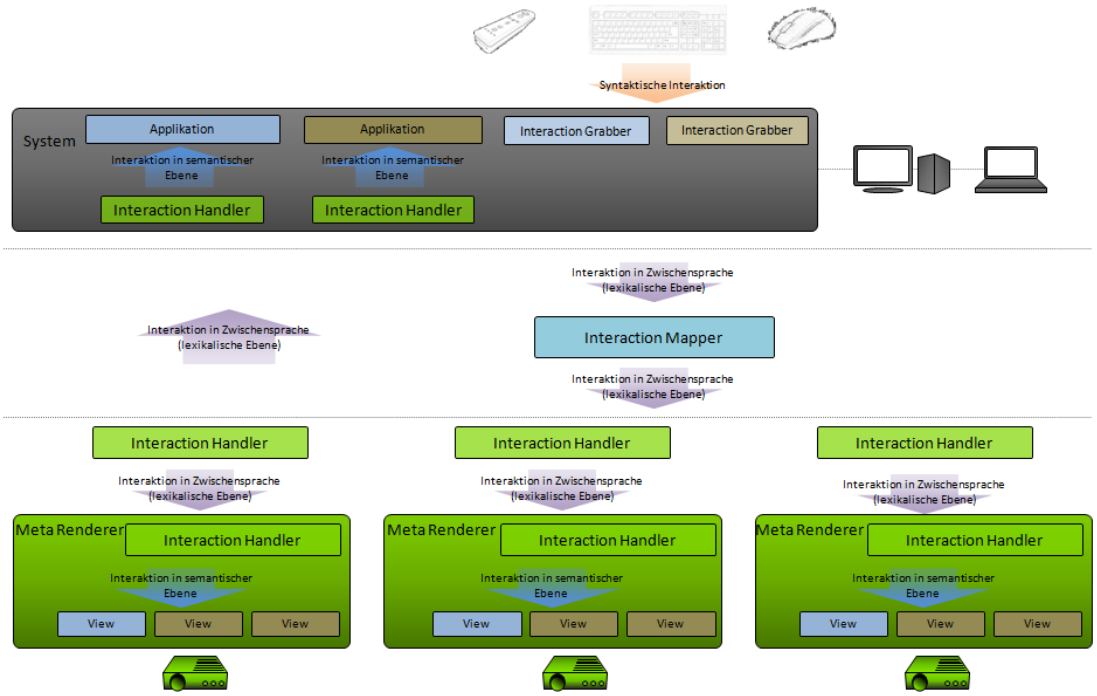



Abbildung 4.3.: Grundsätzlicher Ablauf des *Smart Interaction Managements*. Interaktionen werden vom *Interaction Grabber* ausgewertet, durch den *Interaction Mapper* an den zuständigen *Metarenderer* übertragen und durch den *Interaction Handler* am *Metarenderer* oder an der *View* generierenden Applikation semantisch interpretiert.

## 4.3. Interaktion mit der Anzeige der Views

In diesem Abschnitt wird die Interaktion mit der Anzeige der *Views* thematisiert. Zunächst wird eine Problembeschreibung erarbeitet und danach werden die verschiedenen Teilprobleme bei der Interaktion mit der Anzeige diskutiert.

### 4.3.1. Problembeschreibung

Die Anzeige der *Views* wird im *Smart View Management* automatisch konfiguriert. Die verschiedenen *Views* werden durch das *Smart Display Mapping* automatisch den Displayflächen zugeordnet (vgl. Abschnitt 3.5), durch die Federkraft- und Gitterbasierten Layouts werden die *Views* auf den verschiedenen Displayflächen angeordnet und durch die flexible *JPEG2000* Dekodierung kann die Auflösung der *Views* an das Layout angepasst werden (vgl. Abschnitt 3.6). Nun stellt sich die Frage, warum hier eine interaktive Anpassung der Anzeige sinnvoll sein kann.

Grundsätzlich werden zwei Szenarien im Smart Meeting Room unterschieden: die Präsentation und die Diskussion (vgl. Abschnitt 3.4). Für das *Display Mapping*, und damit die Zuordnung der *Views* zu den Displayflächen, wurden diese beiden Szenarien bzgl. der Positionen der einzelnen Nutzer betrachtet. So werden bei der Präsentation die *Views* neben dem Presenter angezeigt, während bei der Diskussion die *Views* frei auf den Displayflächen verteilt werden (vgl. Abschnitt 3.5).

Diese Szenarien haben allerdings auch Einfluss auf die Auswahl und Anordnung der *Views*. Um dies zu verdeutlichen, wird im Folgenden ein Beispiel für die Präsentation und Diskussion zwischen Experten verschiedener Fachgebiete gegeben. Ziel ist hier die Präsentation und Diskussion von Erkenntnissen, die aus visuellen Repräsentationen gewonnen wurden. Ein dynamischer Übergang zwischen der Präsentation und anschließender Diskussion von Informationen ist ein gängiges Beispiel in Smart Meeting Rooms [BK09, HBGK09].

Am Anfang gibt der Presenter eine Einführung in die zu diskutierenden Daten. Dabei werden auch vorläufige Analyseergebnisse dem Publikum vorgestellt. Das Publikum ist dadurch über die Charakteristik und das Ziel der Diskussion informiert.

Für diesen Schritt ist zunächst keine interaktive Anpassung notwendig, da man davon ausgehen kann, dass der Presenter die Präsentation entsprechend seiner geplanten Moderation erzeugt. Das heißt, der Presenter generiert im Vorfeld der Präsentation *Views* (das können Folien oder einzelne Bilder sein), stellt diese zu *View Packages* zusammen und kann so die automatische Anordnung der *Views* während der Präsentation nutzen.

Nach der einführenden Präsentation startet dann die Diskussion. Im Gegensatz zur Präsentation können sich während der Diskussion die Schwerpunkte und Fragestellungen der Teilnehmer dynamisch ändern. Diese Änderungen kann ein Presenter nicht alle im Vorfeld berücksichtigen. So können beispielsweise Aspekte der Daten relevant werden, die mit den bisherigen *Views* nicht ausreichend kommuniziert sind, oder Daten sollen in Beziehung gesetzt werden, wobei die entsprechenden *Views* nicht gruppiert wurden.

Um also solche Änderungen zu unterstützen, bedarf es einer interaktiven Anpassung der Anzeige. Dabei müssen verschiedene Anforderungen berücksichtigt werden:

1. Auf dieser Stufe der Interaktion sollte die Anpassung auf Ebene der *View* Anzeige stattfinden, also ohne eine Neugenerierung von *Views* auskommen. Dies ermöglicht ein einheitliches Interface für die Anpassung unabhängig von der Implementierung der *View* generierenden Applikationen.

2. Der Presenter sollte in der Lage sein, Anpassungen auf beliebigen Displayflächen selbstständig durchzuführen. So sollte er z.B. *Views* beliebig verschieben können. Aber auch die Diskussionsteilnehmer sollten in der Lage sein, beliebig auf *Views* zuzugreifen, um diese z.B. während ihres Diskussionsbeitrages zu vergrößern und in den Fokus zu rücken.

3. Es sollten nur lokale Änderungen vorgenommen werden. Wird z.B. in einer Diskussion eine Verbindung zwischen bisher nicht zusammenhängenden *Views* entdeckt, soll nicht die gesamte Anzeige neu erstellt werden müssen, um diese beiden *Views* in der Anzeige kombinieren zu können. Um die Orientierung der Nutzer zu erhalten, sollte eine interaktive Anpassung angeboten werden, die weitgehend das Gesamtbild erhält.

Die Anzeige der *Views* besteht grundsätzlich aus der Zuordnung der *Views* zu den Displayflächen und der Anordnung der *Views* auf den jeweiligen Displayflächen (Position, Auflösung) (vgl. Abschnitt 3). Für die Interaktion mit der Anzeige von *Views* wird vom aktuellen Zustand der Anzeige ausgegangen, der vom *Smart Display Mapping* (vgl. Abschnitt 3.5) und einem Layout Mechanismus (vgl. Abschnitt 3.6) automatisch berechnet und dargestellt wurde. Dieser Zustand soll interaktiv so angepasst werden, dass die aktuellen Anforderungen erfüllt werden, die während einer Präsentation oder Diskussion auftreten.

Um eine solche interaktive Manipulation der Anzeige zu ermöglichen, sind prinzipiell drei Aufgaben zu unterstützen: (1) interaktives Zuweisen einer *View* zu einer (anderen) Displayfläche (siehe Abschnitt 4.3.2), (2) interaktive Manipulation der Anordnung der *Views* (siehe Abschnitt 4.3.3) und (3) interaktive Anpassung der Darstellung einer *View* (siehe Abschnitt 4.3.4). Diese Teilprobleme werden in den folgenden Abschnitten behandelt.

### 4.3.2. Interaktive Zuweisung von Views zu Displayflächen

Die interaktive Zuordnung von bereits angezeigten oder neu zu ergänzenden *Views* zu bestimmten Displayflächen ist eine erste sinnvolle Option für die Anpassung der Anzeige. Dadurch kann z.B. eine weitere *View* die dargestellten Informationen anderer *Views* ergänzen.

Dieses Anliegen ließe sich grundsätzlich auch durch das *Smart View Management* erreichen. In diesem Fall müsste der Presenter aber die *View Packages* modifizieren, indem er zusammenhängende *Views* in das gleiche *View Package* verschiebt oder *Views*, die aktuell nicht angezeigt werden, in neue oder bestehende *View Packages* eingliedern. Die Generierung neuer *View Packages* bzw. die Modifikation bestehender *View Packages* hat allerdings einen globalen Einfluss auf die Verteilung der *Views* auf den Displayflächen. Außerdem würde eine neue Zuweisung der *Views* die Berechnung des Layouts auf den Displayflächen neu initialisieren. Die Folge wäre eine Anordnung der *Views*, die zwar die neuen Schwerpunkte der Diskussion aufgreift, allerdings nicht garantiert, dass die Orientierung der Teilnehmer erhalten bleibt.

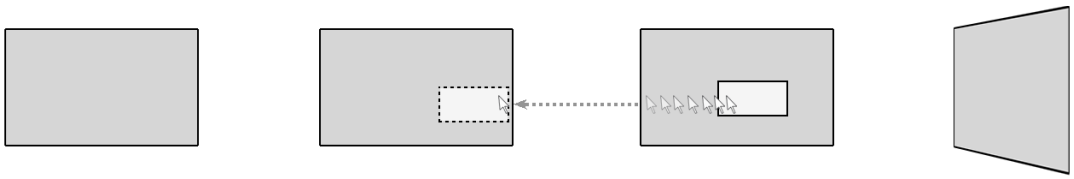


Abbildung 4.4.: Interaktive Zuordnung einer *View* zu einer anderen Displayfläche mittels drag-and-drop Interaktion.

Deshalb sollte hierfür nicht das *Smart View Management* eingesetzt werden, sondern der Nutzer, insbesondere der Presenter, soll die *Views* interaktiv von einer Displayfläche zur anderen verschieben und neue *Views* auf einer Displayfläche anzeigen können.

Für die interaktive Zuweisung von *Views* zu Displayflächen wurden zwei Lösungen entwickelt. Zum einen die Möglichkeit, *Views* per drag-and-drop von einer Displayfläche auf eine andere zu verschieben und zum anderen die Möglichkeit, eine *View* explizit einer Displayfläche zuzuweisen. Diese beiden Lösungen werden im Folgenden beschrieben.

- Das **Verschieben einer View auf eine andere Displayfläche** wird im *Smart Interaction Management* durch den *Interaction Handler* realisiert (vgl. Abschnitt 4.2.5). Dieser informiert darüber, dass der Zeiger des Presenters oder eines Nutzers die Displayfläche mit einer *View* (durch eine drag Interaktion) verlässt (siehe Abbildung 4.4). Anhand des geometrischen Modells der Anordnung der Displayflächen wird die neue Displayfläche durch den *Interaction Mapper* ermittelt (vgl. Abschnitt 4.2.5).

  Die beiden involvierten Displayflächen werden nun über die Änderung der Zuordnung der *Views* informiert und nur für diese wird dann automatisch ein neues Layout berechnet. Die Displayflächen, die nicht involviert sind, registrieren keine Veränderung und behalten ihr aktuelles Layout bei. Damit werden nur lokale Änderungen vorgenommen, die globale Anzeige bleibt erhalten und damit die Orientierung der Nutzer. Je nachdem welcher Layout Mechanismus eingesetzt wurde, ergeben sich folgende Adaptionen:

  - Beim *Gitterbasierten Layout* wird auf den beiden betroffenen Displayflächen die Anzahl der Zellen an die Anzahl der *Views* angepasst und das Gitter wird neu berechnet (vgl. Abschnitt 3.6.3). Die Reihenfolge der angezeigten *Views* bleibt weitestgehend erhalten.
  - Beim *Federkraftbasierten Layout* wird die neue *View* in das aktuelle Layout mit aufgenommen und ausgehend von der aktuellen Anordnung ein neues Layout berechnet (vgl. Abschnitt 3.6.2).

- Die zweite mögliche Interaktion ist die **explizite Zuweisung einer View zu einer Displayfläche**. Hier wird im Gegensatz zur Verschiebung einer *View*, die an die Constraints des geometrischen Modells der Displayflächen gebunden ist, eine weitere *View* konkret einer beliebigen Displayfläche hinzugefügt. Diese zusätzliche *View* kann sowohl eine *View* sein, die bereits auf einer anderen Displayfläche angezeigt wird als auch eine beliebige *View*, die auf einem persönlichen Gerät verfügbar ist und bisher noch nicht angezeigt wird.

  In der gängigen Literatur wird eine interaktive Zuordnung meist über ein zusätzliches visuelles Interface vorgenommen. Dabei wird textuell oder per drag-and-drop in einem globalen Modell des Raums eine visuelle Repräsentation einer Displayfläche zugeordnet (z.B. [BB04]). Für diese Art der Zuordnung müssen im globalen Modell die gewünschte Displayfläche und die gewünschte *View* in einem separaten Interface identifiziert werden.

  Um solch einen Kontextwechsel zu vermeiden, ist es das Ziel der expliziten Zuweisung, auf einer Displayfläche direkt eine *View* mit dem Zeiger auszuwählen, um sie der jeweiligen Displayfläche hinzuzufügen. Dadurch wird es sowohl dem Presenter als auch den Nutzern ermöglicht, *Views* auf einer spezifischen Displayfläche anzuzeigen ohne das Gerät bzw. die Displayfläche bei der Interaktion wechseln zu müssen.

  Für die direkte Zuordnung wird dasselbe visuelle Interface genutzt, welches bereits für die Zusammenstellung der *View Packages* verwendet wurde. Das hat den Vorteil, dass es bereits bekannt ist. Für die *interaktive View Package Generierung* wird eine GUI verwendet, die eine Vorschau der *Views* und eine Ansicht der *View Packages* in einer Liste bietet (vgl. Abschnitt 3.4). Diese GUI wurde in

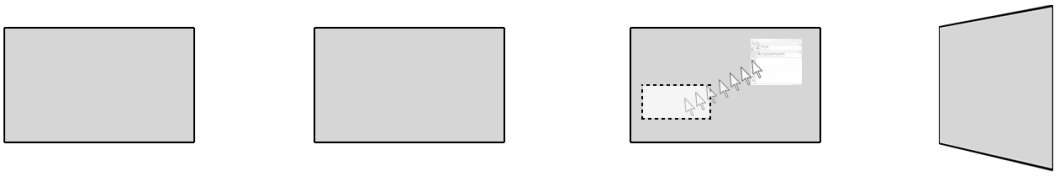


Abbildung 4.5.: Interaktives Hinzufügen einer neuen *View* zu einer Displayfläche. Aus dem im *Metarenderer* gestarteten *View Package* Generator wird mittels drag-and-drop eine *View* hinzugefügt.

den *Metarenderer* integriert und lässt sich damit auf jeder Displayfläche starten.

Durch das Starten der GUI in einem *Metarenderer* wird deshalb bereits die Displayfläche spezifiziert, auf der die neue *View* angezeigt werden soll. Nun kann aus der Liste eine *View* ausgewählt und per drag-and-drop auf die Displayfläche gezogen werden. (siehe Abbildung 4.5). Als Folge dessen wird auch hier wieder das Mapping nur lokal, spezifisch für diese Displayfläche, verändert und das Layout dementsprechend angepasst.

### 4.3.3. Interaktives Anordnen von Views

Die interaktive Anordnung der *Views* auf den Displayflächen ist eine weitere Option, um die Anzeige an die aktuelle Fragestellung anzupassen. Aufgaben während einer Diskussion können z.B. sein, *Views* miteinander zu vergleichen, die im Vorfeld nicht in Beziehung gesetzt wurden oder Informationen, die in *Views* dargestellt sind, genauer zu untersuchen. Für den Vergleich sollten *Views* dabei in möglichst großer räumlicher Nähe zueinander positioniert werden. Für die genauere Untersuchung hingegen sollte eine *View* in möglichst großer Auflösung dargestellt werden.

Obwohl bereits die interaktive Zuweisung von *Views* und *View Packages* zu Displayflächen (vgl. Abschnitt 4.3.2) in gewissem Rahmen einen Vergleich unterstützt, sollen hierfür weitere Konzepte entwickelt werden, um *Views* interaktiv einem geometrischen Raum zuzuweisen und anzuzeigen.

Für die Positionierung von *Views* im gleichen geometrischen Raum, d.h. für uns auf einer Displayfläche, stellen Javed und Elmqvist in [JE12] einen Designspace bereit, der auf einer ausführlichen Literaturanalyse beruht. Dabei werden vier grundlegende Arten beschrieben, *Views* relativ zueinander darzustellen:

**Juxtaposition (nebeneinander)** Anordnung von *Views* nebeneinander.

**Superimposition (überdecken)** Überlagern von zwei *Views* in einer einzigen *View*.

**Overloading (überladen)** Ausnutzen des Raums einer *View* für die Positionierung einer anderen.

**Nesting (einbetten)** Einbetten einer Visualisierung in eine andere.

All diese Anordnungen eignen sich für das Vergleichen von *Views*, da sie eine größere räumliche Nähe zueinander schaffen. Allerdings erfordern das *Overloading (überladen)* und das *Nesting (einbetten)* einen Eingriff in den Prozess der *View* Erzeugung, da es einer Modifikation des Inhalts von *Views* bedarf. Auf die Modifikation des Inhalts wird erst später in Abschnitt 4.4 eingegangen. *Juxtaposition (nebeneinander)* und *Superimposition (überdecken)* verändern hingegen den Inhalt der *Views* nicht und können an dieser Stelle eingesetzt werden.

Zunächst soll davon abstrahiert werden, ob *Juxtaposition (nebeneinander)* oder *Superimposition (überdecken)* für die Anordnung verwendet wird. Wir gehen davon aus, dass die automatischen Layout Verfahren des *Smart View Managements* (vgl. Abschnitt 3.6) eine automatische Anordnung der *Views* auf einer Displayfläche bereitgestellt haben und nun das Layout verändert wird, um Vergleiche durch *Juxtaposition* und *Superimposition* zu unterstützen. Dazu sind zwei grundlegende Schritte notwendig:

**Positionierung** Positionierung bedeutet, dass mit Hilfe des Zeigers die Position einer *View* im Layout interaktiv modifiziert werden kann. Dabei ist die Positionierung abhängig vom verwendeten Layout. Im *Federkraftbasierten Layout* bedeutet dies, dass die *Views* frei auf der Displayfläche angeordnet werden können. Im *Gitterbasierten Layout* wird mit Hilfe des Zeigers die Zuordnung der *Views* zu den Zellen des Gitters modifiziert.

**Skalierung** Eine Skalierung passt die Größe einer *View* an. Auch diese Modifikation ist abhängig vom verwendeten Layout. Während im *Federkraftbasierten Layout* eine freie Skalierung angeboten werden kann (abhängig von dem zur Verfügung stehenden Platz und den Zielauflösungen), müssen im *Gitterbasierten Layout* Methoden angeboten werden, die die Zellen des Gitters verändern können.

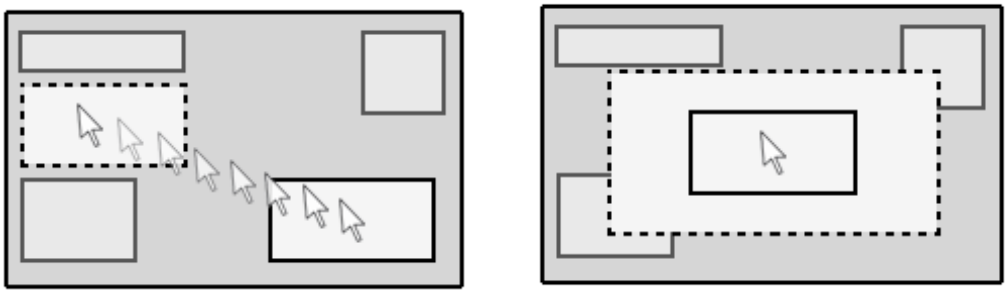
Der Unterschied zwischen *Juxtaposition (nebeneinander)* und *Superimposition (überdecken)* ist, dass bei Letzterem Überlagerungen zulassen werden. Diese führen allerdings zu Verdeckungen. Im Rahmen dieser Dissertation wurde ein neues Verfahren entwickelt, das sich überlagernde Objekte behandelt. Dieses Verfahren wird unabhängig vom verwendeten Layout Mechanismus für die überlagernde Darstellung von *Views* verwendet (siehe Abschnitt 4.3.3.3). Zunächst sollen für *Federkraft-* und *Gitterbasiertes Layout* die beiden Schritte *Positionierung* und *Skalierung* genauer diskutiert werden.

#### 4.3.3.1. Interaktives Anordnen im Federkraftbasierten Layout

Im *Federkraftbasierten Layout* werden die *Views* federkraftbasiert angeordnet und druckbasiert skaliert (vgl. Abschnitt 3.6.2). Diese Federkräfte werden ständig berechnet und mittels Qualitätsfunktion bewertet. Wird ein Layout berechnet, das eine höhere Qualität als das aktuelle Layout besitzt, wird das neue Layout verwendet.

Um eine interaktive Anpassung dieses Layouts zu ermöglichen, muss zunächst die automatische Berechnung unterbrochen werden. Andernfalls würde das automatische Layout die interaktiven Anpassungen wieder durch eine automatische Anordnung ersetzen.

Für das *Modifizieren der Positionen* von *Views* wird eine drag-and-drop Interaktion angeboten. Der Nutzer kann eine *View* mittels Zeiger auswählen. Diese *View* kann nun mittels drag-and-drop frei auf der Displayfläche positioniert werden (siehe Abbildung 4.6a). Hier ergibt sich nun das Problem, dass die Positionierung der *Views* nicht mehr dem allgemeinen Layout Problem entspricht, d.h. *Views* werden nicht mehr zwangsweise überlagerungsfrei dargestellt. Daher wurde die Möglichkeit geschaffen, die Reihenfolge zu modifizieren, in der die *Views* gerendert werden. Durch den Druck auf + bzw. - wird eine *View* in den Vordergrund bzw. in den Hintergrund versetzt.



(a) Verschieben einer *View* mittels drag-and-drop Interaktion.

(b) Skalieren einer *View* mittels Drehung des Mausrads.

Abbildung 4.6.: Interaktive Anordnung von *Views* im *Federkraftbasierten Layout*.

Die *Skalierung* wird durch die Drehung des Mausrades ausgeführt. Durch die Drehung wird automatisch die nächstgrößere bzw. nächstkleinere *Zielauflösung* aus den *JPEG2000* enkodierten *Views* ausgewählt und dargestellt (siehe Abbildung 4.6b).

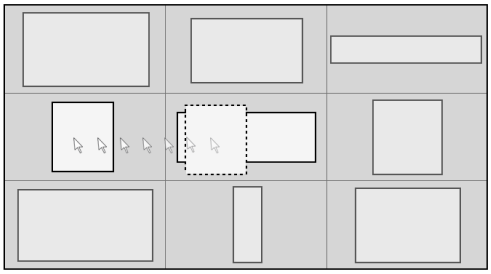
Das *Federkraftbasierte Layout* kann jederzeit wieder aktiviert werden, um die *Views* automatisch anzuordnen. Die interaktiv ausgeführten Modifikationen werden dann allerdings verworfen.

#### 4.3.3.2. Interaktives Anordnen im Gitterbasierten Layout

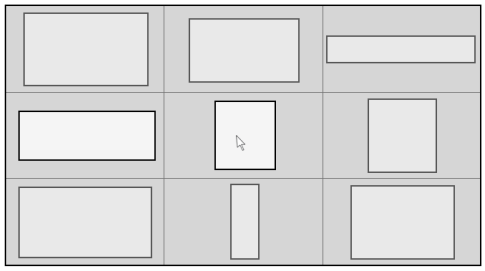
Im *Gitterbasierten Layout* werden die *Views* in Zellen eines Gitters angeordnet. Das Gitter wird automatisch auf Basis der darzustellenden *Views* erstellt und die *Views* werden dann darin positioniert (vgl. Abschnitt 3.6.3).

Auch hier sollen die *Views* bzgl. der Freiheitsgrade des Layouts interaktiv angeordnet werden können. Dabei wird wiederum zwischen dem Modifizieren der Position und der Skalierung einer *View* unterschieden.

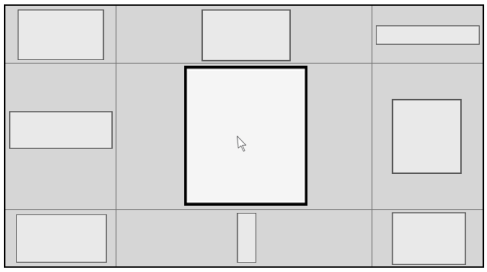
Für das *Modifizieren der Position* im *Gitterbasierten Layout* werden *Views* nicht frei auf der Displayfläche bewegt. Hier werden *Views* vielmehr einer anderen Gitterzelle zugeordnet. Auch dies wird mit Hilfe einer drag-and-drop Interaktion realisiert. Mit Hilfe dieser Interaktion wird eine *View* von einer Zelle in eine andere bewegt (siehe Abbil-



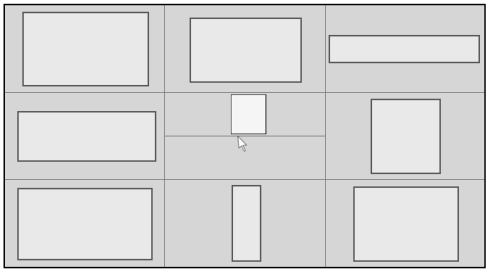
(a) Bewegen einer *View* von einer Zelle in eine andere Zelle mittels drag-and-drop Interaktion.



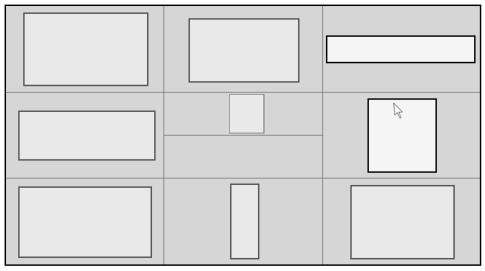
(b) Nach erfolgter drag-and-drop Interaktion für das Verschieben einer *View* tauschen beide *Views* die Zellen.



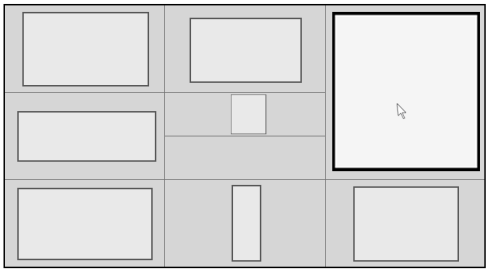
(c) Vergrößern einer Zelle (und Skalierung der darin befindlichen *View*): Die ausgewählte Zelle wird vergrößert während die umliegenden automatisch verkleinert werden.



(d) Teilen einer Zelle (hier horizontal, vertikal analog): zwei gleich große Zellen werden erzeugt, in einer der beiden Zellen wird die *View* der Originalzelle angezeigt.



(e) Zusammenfassen von zwei Zellen: zwei Zellen werden markiert, um zusammengefasst zu werden.



(f) Resultat des Zusammenfassens: die neue Zelle umfasst den Bereich beider ursprünglicher Zellen, wobei die *View* in der neuen Zelle angezeigt wird, die sich in der zuerst markierten Zelle befand.

Abbildung 4.7.: Interaktive Anordnung von *Views* im *Gitterbasierten Layout*.

dung 4.7a). Da durch den automatischen Layout Mechanismus ein Gitter mit möglichst wenigen Gitterzellen erzeugt wird, ist in der Regel die Gitterzelle bereits belegt, in die eine *View* verschoben werden soll. Diese beiden *Views* tauschen daraufhin die Positionen (siehe Abbildung 4.7b).

Da *Views* im *Gitterbasierten Layout* sowohl mit der Position als auch mit der Größe an die Gitterzellen gebunden sind, bedarf es für die *Skalierung* von *Views* interaktiver Methoden für die Anpassung des Gitters. Diese ermöglichen es, sowohl die Größe von Gitterzellen zu modifizieren, als auch neue Zellen für weitere *Views* zu schaffen.

Dabei basieren die Interaktionen auf drei grundlegende Modifikationen: (1) Modifikation von Zellgrößen, (2) Teilen von Zellen und (3) Zusammenfassen von Zellen.

Bei der Modifikation von Zellgrößen werden eine oder eine Menge von rechteckig zusammenhängenden Zellen ausgewählt. Durch das Drücken der + bzw. - Taste werden diese Zellen gleichmäßig in alle Richtungen vergrößert bzw. verkleinert. Die umliegenden Zellen passen sich automatisch an (siehe Abbildung 4.7c). Auch hier wird wieder automatisch eine Skalierung der *Views* aus den Zielauflösungen für die Darstellung verwendet.

Beim Teilen von Zellen wird eine Zelle in zwei Zellen geteilt, die den gleichen Raum einnehmen. Dadurch können mit einer einfachen Interaktion zwei exakt gleich große, nebeneinander liegende Zellen geschaffen werden, um einen Vergleich von *Views* zu unterstützen. Dabei wird zwischen der horizontalen und der vertikalen Teilung unterschieden. Die *View*, die in dieser Zelle angezeigt wurde, verbleibt in einer der beiden Zellen. Die andere wird zunächst leer angezeigt (siehe Abbildung 4.7d). Hier kann die neue *View* für Vergleichszwecke eingepasst werden. Für die Darstellung wird automatisch aus den Zielauflösungen der *View* die größtmöglich passende ausgewählt und die *View* wird in dieser Größe angezeigt.

Beim Zusammenfassen von Zellen müssen vorher multiple Zellen markiert werden. Diese Zellen müssen zusammenhängend sein, dürfen keine Löcher enthalten und müssen ein Rechteck bilden. Werden multiple Zellen zusammengefasst, wird in der resultierenden Zelle die *View* der zuerst markierten Zelle angezeigt (siehe Abbildungen 4.7e und 4.7f). Die anderen *Views* werden nicht mehr dargestellt. Für die Darstellung wird wiederum automatisch die größtmöglich passende Zielauflösung für die Skalie-

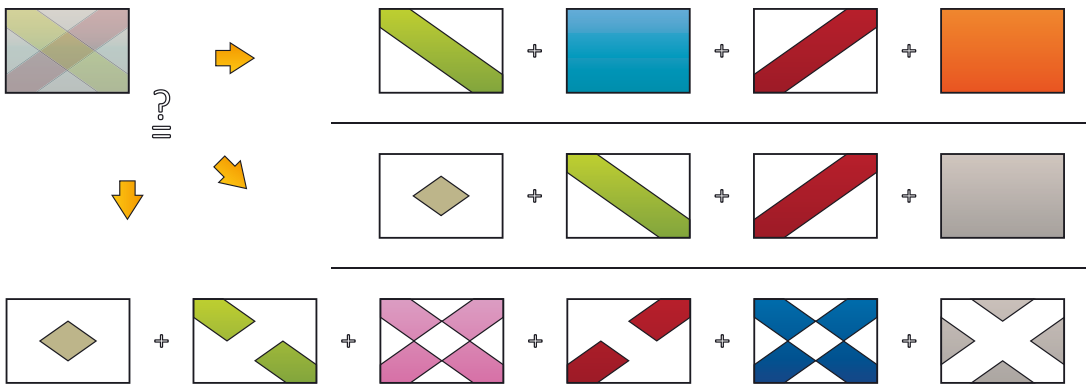


Abbildung 4.8.: Bildliche Darstellung der Probleme einer transparenten Darstellung. Die transparente Darstellung könnte das Resultat der drei dargestellten verschiedenen Objektkombinationen sein.

rung der *View* ausgewählt. Damit bildet das Zusammenfassen von Zellen die inverse Operation zur Teilung von Zellen. Zum einen lässt sich durch das Zusammenfassen eine große Zelle schaffen, zum anderen kann die Teilung von Zellen damit schnell rückgängig gemacht werden.

#### 4.3.3.3. Behandlung von Überlagerungen – ein neuer Weaving Ansatz

Bei der *Superimposition* werden *Views* überdeckend dargestellt. Zur Behandlung sich überdeckender Objekte wurde im Rahmen dieser Arbeit ein neuer Ansatz für das *Weaving* entwickelt. Im Folgenden wird dieses Verfahren genauer erläutert. Danach wird die Anwendung für die überlagernde Darstellung von *Views* diskutiert.

Zur Behandlung von sich überlagernden Objekten wird in der gängigen Literatur Transparenz verwendet. Bei der Verwendung der Transparenz müssen nach [BG04] allerdings verschiedene Probleme beachtet werden:

**Interpretation der Farben** Die transparente Darstellung von verschiedenen Farben erzeugt neue Farben durch die Kombination der einzelnen Farbanteile. Diese neu entstandenen Farben sind allerdings in der Regel nicht Teil der verwendeten Farbskala, wodurch die neue Farbe nicht interpretiert werden kann. Wenn die

neue Farbe allerdings bereits Teil der Farbskala ist, kann es zu Fehlinterpretationen kommen.

**Unterscheidbarkeit der Farben** Nach der transparenten Überlagerung verschiedener Farben ist es sehr schwierig oder gar unmöglich, die ursprünglichen Farben zu rekonstruieren. Erschwerend tritt dabei das Problem auf, dass verschiedene Farbkombinationen zu der gleichen transparenten Farbe führen.

**Erkennbarkeit der Objektformen** Sich transparent überlagernde grafische Objekte bilden im Überlagerungsbereich eine geometrische Überschneidung. Dies kann bei der Betrachtung fälschlicherweise als separates Objekt wahrgenommen werden.

In Abbildung 4.8 werden diese Probleme der Transparenz veranschaulicht. Sie wurden bereits in verschiedenen Arbeiten behandelt. Der Aspekt der Farbwahrnehmung wird etwa in [Hea96, KRC02] adressiert, indem die Farbskala der zu überlagernden Farben angepasst wird. Weiterhin wurden Ansätze entwickelt, die es erlauben, Blendingfunktionen für die Erzeugung der Transparenz zu modifizieren (z.B. [CWM09, WGM+08]). Andere Ansätze versuchen hingegen, die Transparenz gänzlich zu vermeiden. In [Dol07] wurde z.B. ein farbiges Gitter verwendet und die Gitterzellen anteilig bzgl. der auftretenden Farben koloriert.

Ein weiteres Verfahren zur Vermeidung von Transparenz ist das *Weaving*. Das Prinzip des *Weavings* kann zusammengefasst werden als das Nebeneinanderzeichnen von ansonsten sich überlappenden Farben. Das *Weaving* von Farben wurde erstmals von Urness et al. in [UIM+03] für die Visualisierung von LIC-flow Visualisierungen verwendet. Für die Informationsvisualisierung wurde das *Weaving* von Hagh-Shenas, Interrante und Healey [HSIHK06] adaptiert und für Kartendarstellungen eingesetzt.

Im Rahmen dieser Arbeit wurde das *Weaving* systematisiert (vgl. [LRS10a]). Zudem wurden neue *Weaving* Muster zur Behandlung überlappender Objekten entwickelt.

Das Prinzip des *Weavings* beruht darauf, für jede Position im Bild einen Farbwert aus verschiedenen sich überlagernden Ebenen auszuwählen. Sei dabei $E$ die Menge von sich überlagernden Ebenen, dann ist es das Ziel des *Weavings*, für jede Position $(x, y)$ im Zielbild einen Farbwert aus den Ebenen $E$ auszuwählen. Die Farbe an einer bestimmten Position im Zielbild $zf(x, y)$ wird dabei wie folgt bestimmt:

$$zf(x,y) = qf(x,y,e_i) \tag{4.1}$$

Dabei bestimmt $qf(x,y,e_i)$ die Quellfarbe, also die Farbe der Ebene $e_i$ an der Stelle $(x,y)$. $e_i$ wird dann wie folgt bestimmt:

$$e_i = f_{sel}(x,y) \tag{4.2}$$

Die Funktion $f_{sel}(x,y)$ wählt die zu verwendende Ebene in Abhängigkeit von der Position aus. Die konkrete Realisierung von $f_{sel}(x,y)$ bestimmt dabei das *Weaving* Muster.

Das *Weaving*, wie es von Hagh-Shenas et al. vorgeschlagen wurde [HSIHK06], wird für die Kolorierung von Flächen einer Karte verwendet. Für jedes Pixel einer Kartenregion wird zufällig eines von sechs Farben ausgewählt. Da diese Selektion zufällig ist, kann es als *Zufälliges Weaving* bezeichnet werden. Die zugehörige Auswahlfunktion $f_{sel}(x,y)$ sieht dann wie folgt aus:

$$f_{sel}(x,y) = random\,(E)\,. \tag{4.3}$$

Das Ergebnis des *Zufälligen Weaving* wird in Abbildung 4.9c dargestellt. Hier sind im Gegensatz zur Transparenz die *Farben* voneinander trennbar. Die Verteilung der Farben ähnelt allerdings gleichförmig verteiltem Rauschen. Das macht die Farbwahrnehmung schwieriger. Speziell in kleinen Teilen mixen sich wiederum die Farben optisch. Der Einfluss der Größe gleichfarbiger zusammenhängender Farbblöcke wurde in [HSIHK06] gezeigt. Hier wurde auch vorgeschlagen, größere zusammenhängende Blöcke zu verwenden. Das würde es auch vereinfachen, auf den ersten Blick festzustellen, welche Farben überhaupt verwendet wurden.

Die Wahrnehmung der *Form* wird durch das *Zufällige Weaving* eher erschwert. Durch die zufällige Verteilung sind die Konturen der Objekte ausgefranst und schwer zu erkennen. Im Falle größerer zusammenhängender Blöcke für eine bessere Farbwahrnehmung würde das Problem der Formwahrnehmung noch erschwert werden.

Neben dieser zufälligen Auswahl lassen sich weitere *Weaving* Muster einsetzen. Im Folgenden werden zwei weitere *Weaving* Muster vorgestellt, die im Rahmen dieser Dissertation entwickelt wurden: (1) das *stack-aligned Weaving*, und (2) das *modulo-aligned Weaving*.

Das *stack-aligned Weaving* liefert eine spaltenweise oder zeilenweise geordnete Auswahl der Pixel, die es ermöglicht, Inhalte direkt einer *View* zuzuordnen (siehe Abbildung 4.14e). Zur Vereinfachung der Beschreibung wird im Folgenden das spaltenweise angeordnete Muster verwendet. Zur Erzeugung des Musters wird jede Spalte mit einem Stack $\langle \ldots \rangle\rangle_{Spalte}$ aller Ebenen $E$ assoziiert. Die Reihenfolge der Ebenen zirkuliert in aufeinander folgenden Spalten. Bei beispielsweise $n$ Ebenen ist der Stack der Spalte 1 $\langle e_1, e_2, e_3, \ldots, e_n \rangle\rangle_1$, bei Spalte 2 $\langle e_2, e_3, e_4, \ldots, e_n, e_1 \rangle\rangle_2$, bei Spalte 3 $\langle e_3, e_4, e_5, \ldots, e_n, e_1, e_2 \rangle\rangle_3$ usw. Damit sind $n$ verschiedene Stacks verfügbar und bei der $(n+1)$ten Spalte wird wiederum der erste Stack verwendet.

Für die Auswahl des Farbwertes $zf$ wird dann die oberste Ebene im Stack ermittelt, in der ein Farbwert vorhanden ist. Die Selektionsfunktion $f_{sel}$ sieht dann wie folgt aus:

$$f_{sel}(x,y) = e_{top} | e_{top} \text{ ist oberstes Element in } \langle \ldots \rangle\rangle_{(x \mod n)}, \text{bei dem } qf_{e_{top}} \neq \text{leer} \quad (4.4)$$

Für jedes Pixel wird aus dem jeweiligen Stack das oberste Element selektiert, das einen Farbwert enthält. Das Resultat ist in Abbildung 4.9d dargestellt.

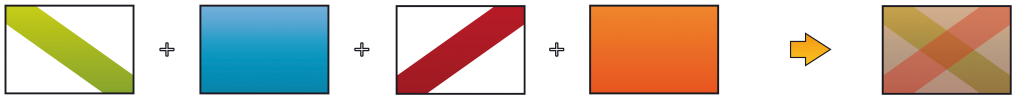
Um die Erkennung der Konturen weiter zu vereinfachen, wurde das *modulo-aligned Weaving* Muster entwickelt. Auch hier wird wieder zeilen- oder spaltenweise ein Muster zur Anordnung der Farbwerte verwendet. Im Gegensatz zum *stack-aligned Weaving* wird hier allerdings nicht ein Stack aus allen im Bild vorhandenen Ebenen erstellt, sondern zunächst eine Liste aus allen beteiligten Ebenen. Eine beteiligte Ebene ist dabei eine Ebene, bei der an der Position $(x,y)$ ein Farbwert vorhanden ist. Die Menge der beteiligten Ebenen $BE$ an der Position $(x,y)$ lässt sich wie folgt bilden:

$$BE(x,y) = \{e_i \in E | qf(x,y,e_i) \neq leer\} \quad (4.5)$$

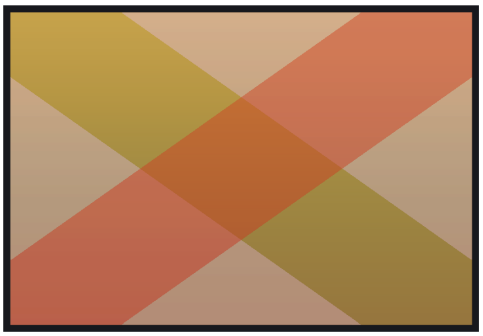
Für die Auswahl des Farbwertes an der Stelle $(x,y)$ wird beim *modulo-aligned Weaving* zunächst eine sortierte Liste $L$ der beteiligten Ebenen erstellt $(e_0, e_1, \ldots, e_k)$. Dabei sei $k$ die Menge aller beteiligten Ebenen mit $k = |BE(x,y)|$. Dann wird die Ebene an der Stelle $e_m$ aus der Liste der beteiligten Ebenen verwendet, wobei $m = (x \; mod \; k)$. Die Selektionsfunktion des *modulo-aligned Weaving* wird folgendermaßen gebildet:

$$f_{sel}(x,y) = e_m | e_m \in BE \text{ und Element } m \text{ aus } L \text{ wobei, } m = (x \; mod \; k). \quad (4.6)$$

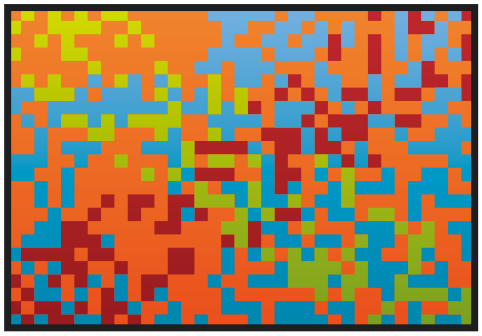
Die Anwendung des *modulo-aligned Weaving* ist in Abbildung 4.9e dargestellt.



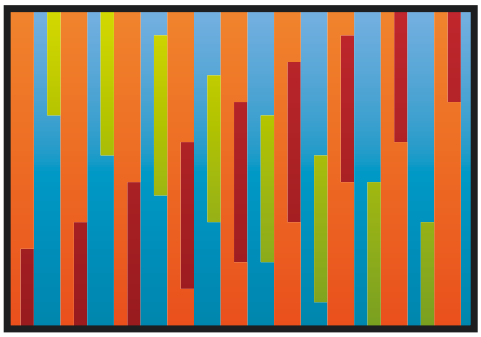
(a) Einfaches Beispiel zur Demonstration des *Weavings*. Vier Ebenen überdecken sich in einem Bild. Zwei quer laufende Balken (grün und rot) schneiden sich auf orangem und blauem Hintergrund. Das Ergebnis bei der Anwendung der Transparenz ist zum Vergleich rechts im Bild dargestellt.



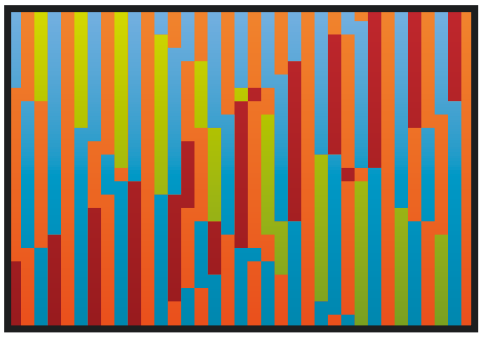
(b) Ergebnis der Anwendung von Transparenz.



(c) Ergebnis der Anwendung des *Zufälligen Weavings*.



(d) Ergebnis der Anwendung des *stack-aligned Weavings*.



(e) Ergebnis der Anwendung des *modulo-aligned Weaving*.

Abbildung 4.9.: Ergebnis der Anwendung verschiedener *Weaving* Muster.

Zur Demonstration der verschiedenen *Weaving* Muster wird im Folgenden ein einfaches Beispiel verwendet. Dieses Beispiel ist in Abbildung 4.9a dargestellt. Dort schneiden sich zwei unterschiedlich farbige Balken auf zwei gleichfarbigen Hintergründen.

**Stack-aligned Weaving** Die *Farben* sind im *Stack-aligned Weaving* gut voneinander unterscheidbar (siehe Abbildung 4.9d). Durch die Streifen werden große zusammenhängende Fläche geschaffen, wie es in [HSIHK06] vorgeschlagen wurde.

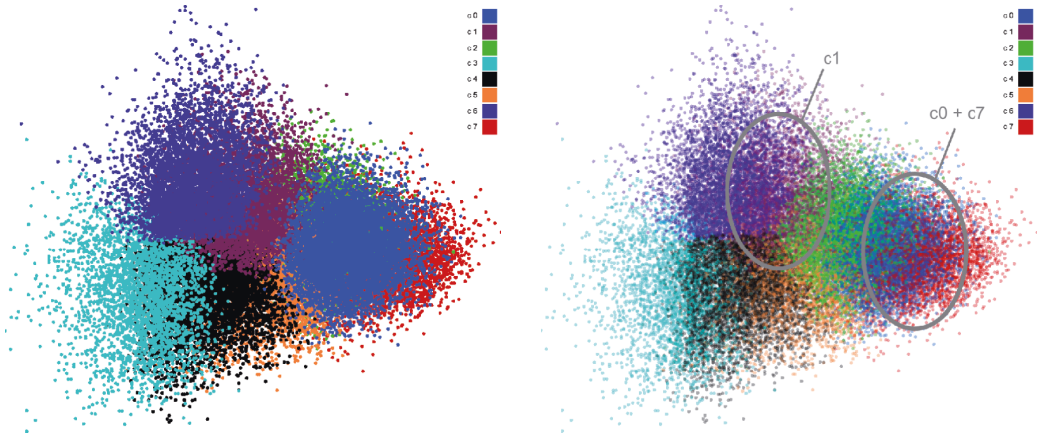
Die *Form* der einzelnen Objekte kann besser als im *Zufälligen Weaving* erkannt werden. Durch die fixe Anordnung der Ebenen kann der Fall auftreten, dass nebeneinander liegende Spalten in der gleichen Farbe dargestellt werden. Dies wird dadurch ausgelöst, dass in diesen Spalten die höchstpriorisierten Ebenen keine Farbe enthalten. Deshalb kann es vorkommen, dass Farben prominenter erscheinen als der Anteil der Objekte real ausmacht. Dies kann sich dann auch auf die Wahrnehmung der Form auswirken.

**Modulo-aligned Weaving** Auch beim *Modulo-aligned Weaving* sind die *Farben* durch die Bildung größerer Blöcke sehr gut voneinander unterscheidbar (siehe Abbildung 4.9e). Der Effekt der zu prominent auftretenden Farben, wie es beim *stack-aligned Weaving* auftritt, wird aber verhindert.

Die *Form* der Objekte ist hier zudem besser erkennbar. Auch der Überlagerungsbereich der beiden Balken tritt beim *modulo-aligned Weaving* stärker hervor.

Im Folgenden wird die Anwendung der neuen *Weaving* Muster an einem konkreten Beispiel, der Darstellung sich überlagernder Cluster in *Scatterplots*, demonstriert, um daraus abgeleitet die Anwendung für die Überlagerung von *Views* zu motivieren.

*Scatterplots* sind ein Beispiel für Darstellungen, in denen ein hohes Maß an Überlagerung auftreten kann. Sie werden oft dazu verwendet, die Verteilung von Datenwerten und Clustern zu analysieren. Dafür werden den unterschiedlichen Clustern verschiedene Farben zugeordnet. Durch die Vielzahl der Datenwerte und damit die Vielzahl sich überlagernder Punkte sind die Formen der Cluster oft schwer zu erkennen. Abbildung 4.10a zeigt einen typischen Fall. In dieser Abbildung wurde kein Verfahren zur Auflösung der Verdeckungen angewendet. Dieser *Scatterplot* wurde aus einem 10 dimensionalen Klimadatensatz erstellt und enthält ca. 40.000 Datenwerte in 8 Clustern.




(a) Ohne Behandlung der Verdeckung.

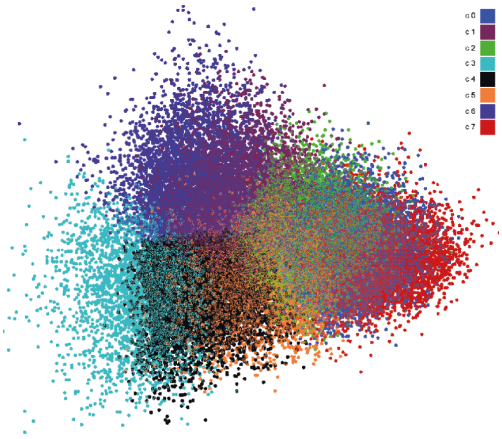
(b) Nutzung von Transparenz: Das Blending von Farbe 0 und Farbe 7 ergibt Farbe 1.

Abbildung 4.10.: *Scatterplot* mit Überlagerung und Behandlung mit Transparenz.

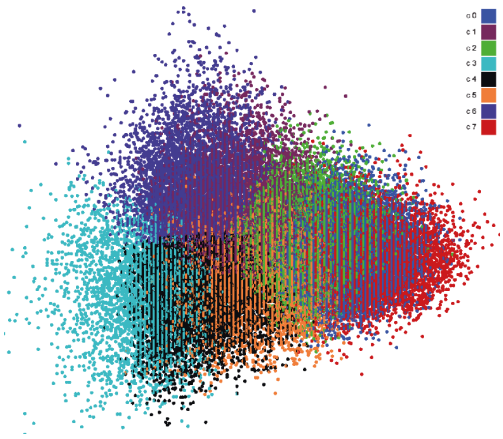
Als Vergleich wurde Transparenz und *Zufälliges Weaving* zur Auflösung der Verdeckung verwendet. Bei der Anwendung der Transparenz führt das Blending der Farben zu einer fehlerhaften Interpretation der Darstellung. Abbildung 4.10b zeigt deutlich, dass die transparente Überlagerung der Farben 0 und 7 exakt die Farbe 1 erzeugt. Damit ist eine eindeutige Interpretation nicht mehr möglich. Auch die Anwendung des *Zufälligen Weavings* verbessert hier nicht die Wahrnehmbarkeit. Durch die zufällige Verteilung wird ein Effekt hervorgerufen, wie er aus dem Dithering bekannt ist. Es mischen sich die Farben wiederum optisch und klare Grenzen zwischen den Clustern sind schwer auszumachen (siehe Abbildung 4.11a).

Die Anwendung des *modulo-aligned Weavings* unterstützt die Wahrnehmbarkeit hingegen besser. Die Farben sind klar voneinander unterscheidbar und die Grenzen der Cluster lassen sich deutlich ausmachen. Durch das sich ständig verändernde Muster auf Grund der ständigen Änderung der beteiligten Ebenen wird der Darstellung aber zusätzliches optisches Rauschen hinzugefügt (siehe Abbildung 4.11b).

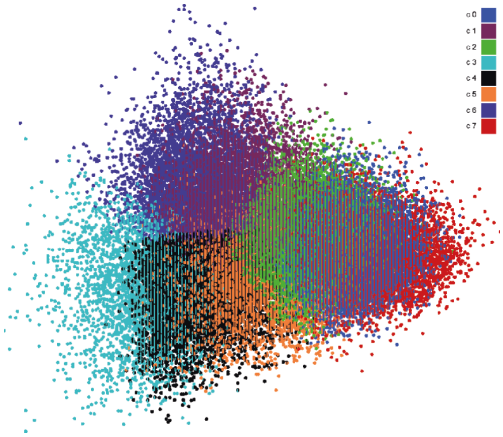
Auf Grund der hochfrequenten Änderung der beteiligten Ebenen in solch einem *Scatterplot* ist hier die Verwendung des *stack-aligned Weavings* von Vorteil. Wie in Abbildung 4.11c sichtbar ist, sind ebenso die Farben und Grenzen der Cluster klar erkennbar, das optische Rauschen wurde durch das verwendete *Weaving* Muster reduziert.



(a) Nutzung von *Zufälligem Weaving*: Klare Grenzen zwischen den Clustern sind schwer zu erkennen.



(b) Nutzung von *modulo-aligned Weaving*: Farben und Grenzen sind gut erkennbar, allerdings erzeugt das *modulo-aligned Weaving* zusätzliches Rauschen in der Darstellung.



(c) Nutzung von *modulo-aligned Weaving*: Farben und Grenzen sind gut erkennbar, das Rauschen ist durch das strikte Muster reduziert.

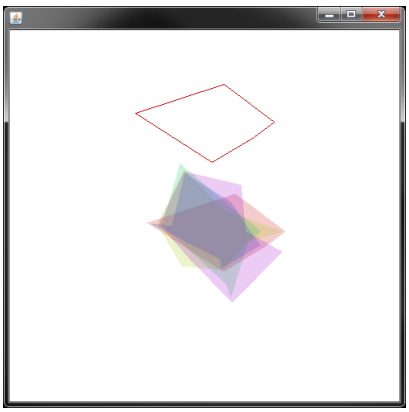
Abbildung 4.11.: *Scatterplot* mit unterschiedlichen *Weaving* Mustern.

Dieses Beispiel zeigt, dass das *Weaving* eine bessere Trennbarkeit der Farben und Formen von sich überlagernden Clustern bietet. Bei *Scatterplots* handelt es sich um eine feingranulare Darstellung. *Views* hingegen stellen eine geschlossene Fläche dar. Daher wurde im Rahmen dieser Dissertation eine Nutzerstudie mit dem Ziel durchgeführt, die Unterscheidbarkeit von flächigen Objekten bei Anwendung der neuen *Weaving* Muster (*stack-aligned Weaving* und *modulo-aligned Weaving*) zu untersuchen und dabei mit etablierten Methoden zu vergleichen.

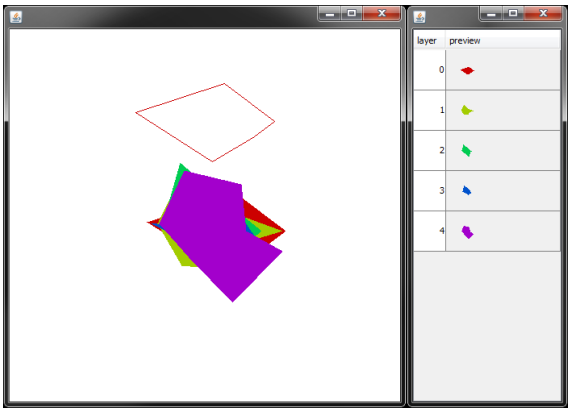
Die Unterscheidbarkeit von Farben bei der Anwendung des *Zufälligen Weavings* wurde bereits von Hagh-Shenas et al. in [HSIHK06] nachgewiesen. Dabei wurde die multivariate Kodierung auf Kartendarstellungen untersucht und mit der transparenten Darstellung verglichen. Die Studie hat gezeigt, dass das *Zufällige Weaving* durchgängig in allen Tests besser war als die Transparenz. Im Gegensatz zum *Zufälligen Weaving* erzeugen das *stack-aligned Weaving* und das *modulo-aligned Weaving* größere zusammenhängende Bereiche mit gleicher Farbe. Hagh-Shenas et al. weisen bei der Auswertung ihrer Studie schon darauf hin, dass die Farbwahrnehmung durch größere zusammenhängende Flächen verbessert wird. Daher ist eine weitere Studie zur Farbwahrnehmung der neuen *Weaving* Muster nicht notwendig.

Für die Evaluation wurde ein konstantes Setup verwendet (Displayauflösung von 1920x1200 Pixel, Abstand der Probanden zur Displayfläche 80cm, einheitliche Beleuchtung). Der Test wurde in einem 500x500 Pixel Fenster durchgeführt.

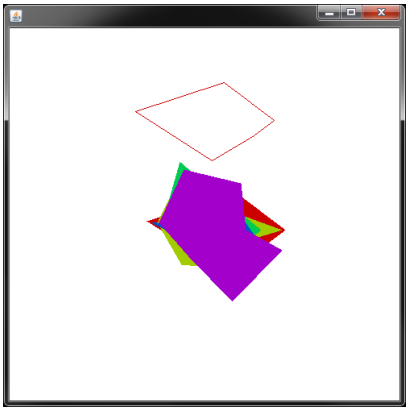
In diesem Fenster wurden zufällig 5 oder 8 sich überlagernde zufällig erstellte Formen gezeigt (siehe Abbildung 4.12). Die Formen wurden mit einer Auswahl an Farben aus einer fixen Farbtabelle koloriert. Dieses Setup wurde verwendet, um einen Lerneffekt auszuschließen. In einer Zielregion wurde die Kontur einer der Formen angezeigt. Diese Form sollte identifiziert und in die Zielregion gezogen werden. Nachdem alle 5 oder 8 Formen in die Zielregion bewegt wurden, war der Durchlauf beendet und nach einer Pause von 2 Sekunden wurden die nächsten Formen angezeigt. Dabei wurden fünf verschiedene Techniken verglichen: (1) *Transparenz*, wobei die oberste Form gegriffen werden konnte (siehe Abbildung 4.12a), (2) *Liste mit Ebenen* (bekannt etwa aus Adobe Photoshop), bei der die Form der aus der Liste ausgewählten Ebene verschoben werden kann (siehe Abbildung 4.12b), (3) *Splatter* Technik [RRC+06], bei der die Formen nach Aktivierung der Technik radial um den Mauszeiger angezeigt werden



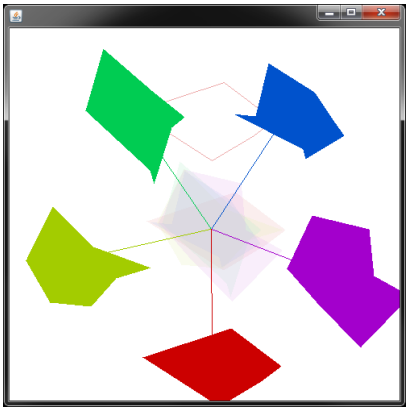
(a) *Transparenz*



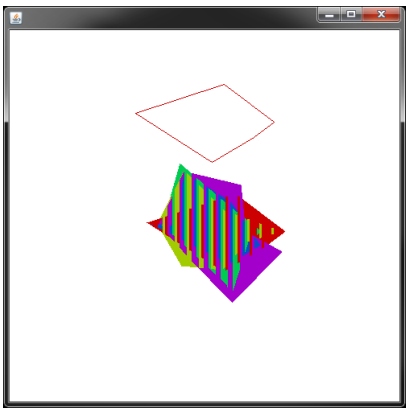

(b) *Liste mit Ebenen*



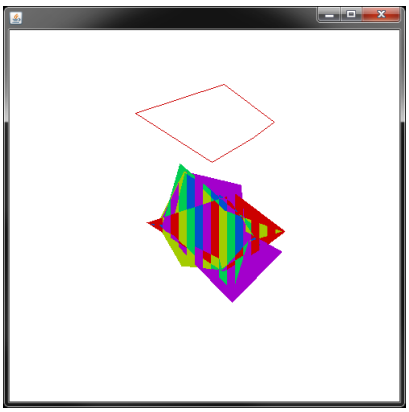
(c) *Splatter* ohne Aktivierung



(d) *Splatter* nach Aktivierung



(e) *Stack-aligned Weaving*



(f) *Modulo-aligned Weaving*

Abbildung 4.12.: Studie zur Untersuchung der neuen *Weaving* Muster. Sich überlagernde Formen werden mit verschiedenen Techniken dargestellt.

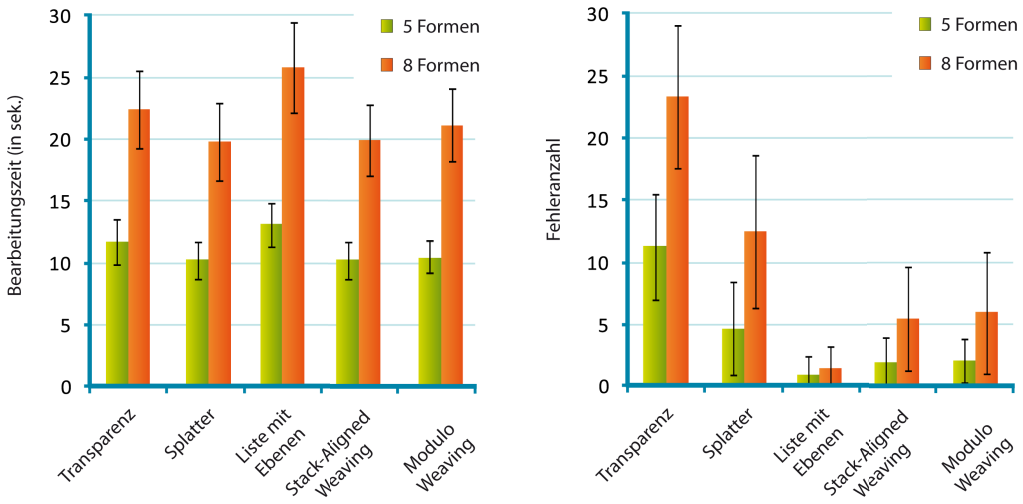
und dann ausgewählt werden können (siehe Abbildung 4.12d), (4) *stack-aligned Weaving* (siehe Abbildung 4.12e) und (5) *modulo-aligned Weaving* (siehe Abbildung 4.12f). Beim *Weaving* konnte eine Form ausgewählt werden, indem die korrespondierende Farbe mit dem Mauszeiger ausgewählt und mittels drag-and-drop bewegt wird.

An der Studie nahmen 10 Probanden (4 weiblich, 6 männlich) im Alter von 25 bis 35 teil. Jeder Proband testete dabei für jede Interaktionstechnik 20 Beispiele, was insgesamt 1.000 Testbeispielen entspricht. Gemessen wurden die Zeit, die für die Bearbeitung eines Durchlaufs (5 oder 8 Formen) gebraucht wurde und die Fehlerrate bei der Bearbeitung. Die Fehlerrate entspricht dabei den fälschlich bewegten Formen, also jenen, die bewegt wurden, obwohl sie nicht an der Reihe waren, in den Zielbereich bewegt zu werden.

Im Vorfeld der Studie wurden folgende Hypothesen formuliert:

1. *Transparenz* benötigt die kürzeste Zeit, da es die für die Probanden bekannteste Technik ist. Weiterhin hat die *Transparenz* die höchste Fehlerrate.
2. Die Fehlerrate des *Splatter* ist gering aber die benötigte Zeit ist nahezu so hoch wie bei der *Liste mit Ebenen*.
3. Die *Liste mit Ebenen* hat die geringste Fehlerrate aber benötigt die meiste Zeit.
4. Das *stack-Aligned Weaving* und das *modulo-aligned Weaving* benötigen eine ähnliche Zeit wie *Transparenz* mit einer Fehlerrate ähnlich der *Liste mit Ebenen*.
5. Je geringer die benötigte Zeit ist, desto höher ist die Fehlerrate und umgekehrt.

Die Ergebnisse der Studie sind in Abbildung 4.13 dargestellt. Die Ergebnisse der Zeit und der Fehlerrate unterschieden sich mit Ausnahme der *Liste mit Ebenen* von den Erwartungen. Die *Transparenz* wurde von den Probanden als mühsam wahrgenommen, da nicht klar war, welches die oberste Form ist. Daher sind sowohl die Fehlerrate als auch die benötigte Zeit entgegen der Erwartung aus Hypothese 1 hoch. Auch *Splatter* verhielt sich unerwartet. Hier sind entgegen den Erwartungen geringe Zeit und relativ hohe Fehlerraten aufgetreten. Dies resultierte daraus, dass die Probanden trotz einer intensiven Erläuterung und Einführung die *Splatter* Technik nicht nutzten, sondern die oberste solide dargestellte Form zunächst zur Seite schoben, um an die zu



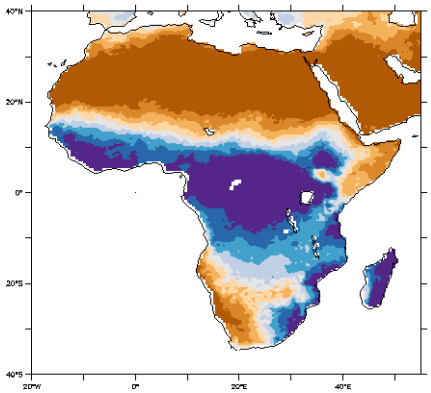

(a) (b)

Abbildung 4.13.: Die Resultate der Nutzerstudie. Abbildung (a) zeigt die durchschnittlich benötigte Zeit und Abbildung (b) die dabei gemachten Fehler.

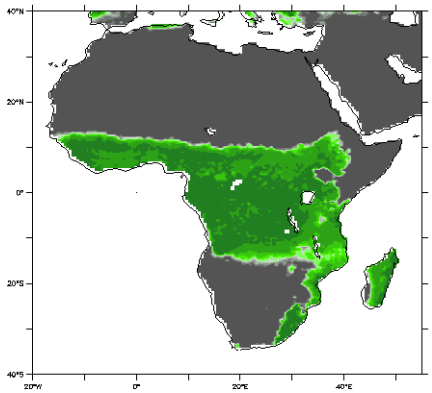
verschiebende Form zu gelangen. Dies entspricht der Erwartung für die *Transparenz* im Vorfeld, da dies die bekannteste Interaktion darstellt.

Das *stack-Aligned Weaving* und das *modulo-aligned Weaving* verhielten sich in etwa wie erwartet. Bei geringer benötigter Zeit war die Fehlerrate relativ gering. Damit ist das *Weaving* ein guter Kompromiss zwischen den verschiedenen Techniken. Dies gibt einen Hinweis darauf, dass die *Weaving* Technik im Allgemeinen gut dafür geeignet ist, überlagernde Objekte zu behandeln. Die Erkennbarkeit der Objekte und der Farben wird nachweislich verbessert.

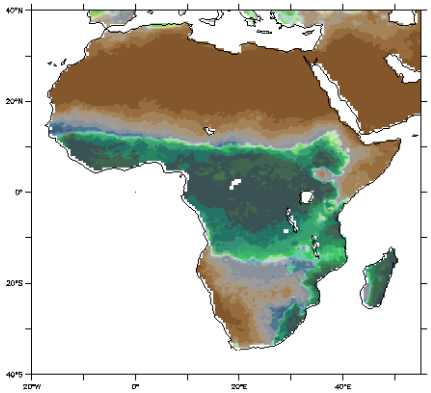
Abbildung 4.14 zeigt die *Superimposition* von *Views*. Hier wird der Vergleich mit den verschiedenen Ansätzen der Literatur sehr deutlich. Bei der *Transparenz* (Abbildung 4.14c) ist eine Zuordnung der Farben zu den *Views* sehr schwierig. Zusätzlich werden neue Farben erzeugt, die in den originalen *Views* nicht vorhanden sind. Beim *Zufälligen Weaving* sind die Farben eindeutig zu erkennen, die Zuordnung zu den *Views* ist auf Grund des Musters allerdings sehr schwierig. Die Nutzung des *stack-aligned Weaving* ermöglicht sowohl die Erkennung der Farben als auch die Zuordnung zu den originalen *Views*.



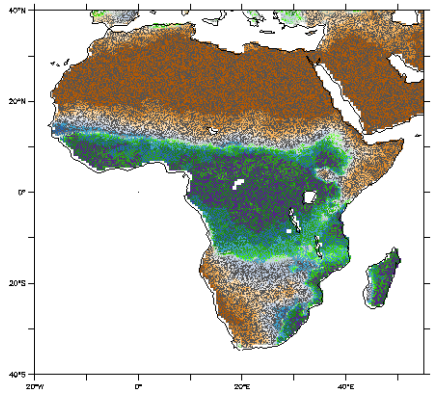
(a) Original *View* 1



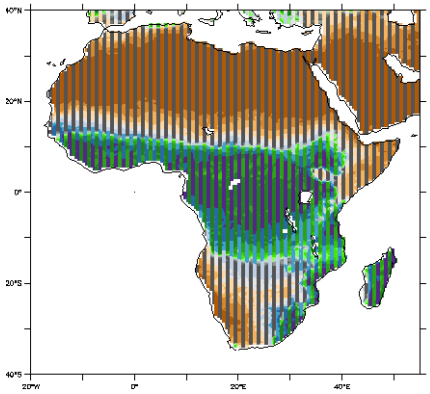
(b) Original *View* 2



(c) Überlagerung zweier *Views* mittels Transparenz.

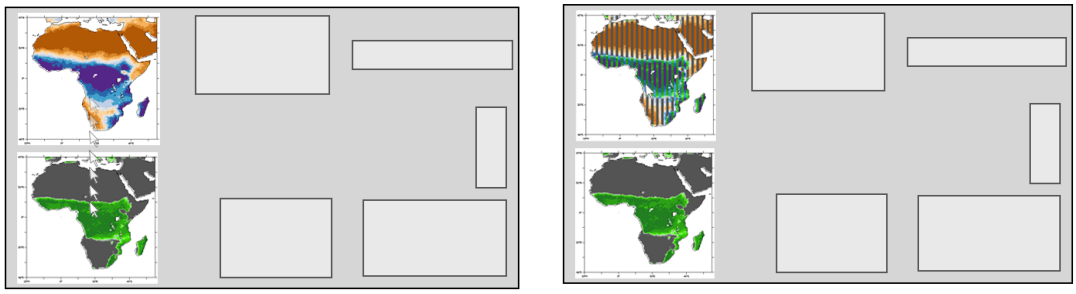


(d) Überlagerung zweier *Views* mittels *Zufälligem Weaving*.



(e) Überlagerung zweier *Views* mittels *stack-aligned Weaving*.

Abbildung 4.14.: Überlagerte Darstellung zweier *Views* im Detail.



(a) (b)

Abbildung 4.15.: Interaktion für die Überlagerung von *Views* durch *Weaving*. Mittels drag-and-drop wird das *Weaving* von *Views* interaktiv ausgewählt.

Für die *Superimposition* von *Views* gibt es keinen Unterschied zwischen der Nutzung des *modulo-aligned Weavings* und des *stack-aligned Weavings*. Während das *stack-aligned Weaving* feste Reihenfolgen der überdeckten Ebenen verwendet, variieren die Reihenfolgen beim *modulo-aligned Weaving* entsprechend der beteiligten Ebenen. Da bei einer *View* aber stets alle Ebenen beteiligt sind, liefern beide *Weaving* Muster hier dasselbe Ergebnis.

Daher können sowohl das *stack-aligned Weaving* als auch das *modulo-aligned Weaving* genutzt werden, um *Views* in *Superimposition (überdeckend)* darzustellen. Durch das *Weaving* wird damit keine Farbmischung erzeugt und die Farben können eindeutig den ursprünglichen *Views* zugeordnet werden.

Die überlagerte Darstellung zweier *Views* kann interaktiv vom Presenter oder Nutzer in beiden Layouts hergestellt werden. Um diese Darstellung zu erhalten, selektiert der Presenter oder Nutzer eine *View* mit seinem Zeiger. Mittels drag-and-drop Interaktion wird dann ein *View* auf den zu überlagernden *View* gezogen. Wird bei der drag-and-drop Interaktion eine zusätzliche Taste ( [Alt] ) gedrückt, wird statt der Neupositionierung eine *Superimposition* der *Views* mittels *Weaving* hergestellt (siehe Abbildung 4.15).

### 4.3.4. Manipulation der Anzeige

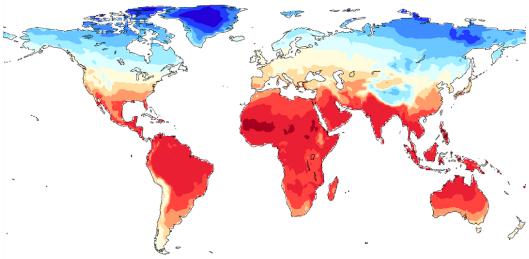
Zur Unterstützung der Diskussion ist es oft von Vorteil, wenn sowohl der Presenter als auch die Nutzer die *View* Anzeige so manipulieren können, dass ein spezieller Fokus auf Bereiche von Interesse gelenkt werden kann. Daher sollte es sowohl dem Presenter als auch dem Nutzer ermöglicht werden, *Regionen von Interesse* (*RoI*) interaktiv zu spezifizieren.

Auf Basis dieser *RoI* kann dann die Anzeige einer *View* interaktiv manipuliert werden. So kann z.B. die Auflösung einer Region einer *View* erhöht werden, um während einer Diskussion diesen Bereich genauer untersuchen zu können, ohne eine neue *View* generieren zu müssen. Die Manipulation der Anzeige von *Views* auf Basis von *RoI* stellt dabei einige Anforderungen.

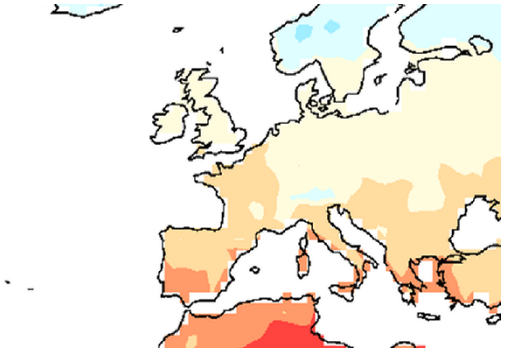
Die *RoI* sollten von Presenter und Nutzer gleichermaßen definierbar sein. Außerdem sollten die *RoI* interaktiv manipulierbar sein, um den dynamisch wechselnden Schwerpunkten einer Diskussion gerecht zu werden.

Weiterhin sollte die Anzeige der *RoI* ermöglicht werden, ohne dass in den Erzeugungsprozess der *Views* eingegriffen werden muss. Damit wird zum einen die Funktionalität für alle dargestellten *Views* zur Verfügung gestellt und nicht nur für die *Views*, bei denen die *View* generierenden Applikationen die *RoI* implementieren. Zum anderen wird damit ein einheitliches Interface für die Definition und Anzeige der *RoI* geschaffen.

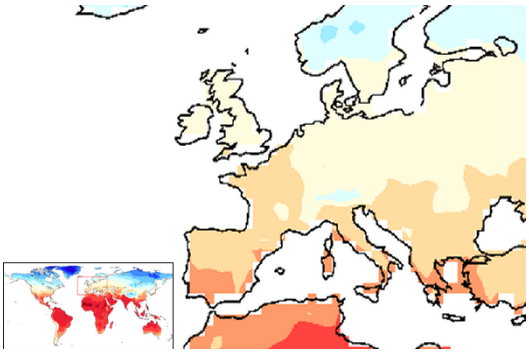
Diese Anforderung lässt sich einfach umsetzen, da die *Views JPEG2000* kodiert sind. Damit können die von Rosenbaum vorgestellten Anzeigetechniken [Ros06] integriert werden. Zur Anzeige von *RoI* stellt Rosenbaum ein Klassifikationsschema vor. Darin wird grundlegend unterschieden zwischen: der Anzeige von (1) *RoI ohne Hintergrund* und der Anzeige von (2) *RoI mit Hintergrund*. Wird zu der *RoI* ein Hintergrund mit angezeigt, wird für die Darstellung zwischen (A) *Detail&Overview* und (B) *Fokus&Context* unterschieden.



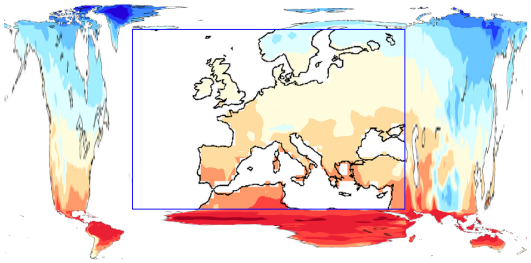
(a) Originale *View* auf der ein Bereich von Interesse ausgewählt werden soll.



(b) Darstellung eines *RoI* ohne Hintergrund.



(c) Darstellung eines *RoI* mit *Detail&Overview*.



(d) Darstellung eines *RoI* mit *Focus&Context*.

Abbildung 4.16.: Definition eines Bereichs von Interesse auf einer *View*.

Zur Unterstützung der Diskussion werden demnach drei Anzeigestrategien zur Verfügung gestellt:

**RoI ohne Hintergrund** Nur die *RoI* wird angezeigt. Die *RoI* wird mit der gleichen Skalierung wie die ursprüngliche *View* angezeigt, kann aber mit höherer Auflösung dargestellt werden, da nur ein Ausschnitt aus der ursprünglichen *View* angezeigt werden soll (siehe Abbildung 4.16b).

**RoI mit Detail&Overview** Die *RoI* wird in höherer Auflösung als die ursprüngliche *View* im Detail in der Skalierung der originalen *View* dargestellt. Den Hintergrund bildet die originale *View* als kleine skalierte Miniaturansicht. Diese Miniatur bildet den Überblick (siehe Abbildung 4.16c).

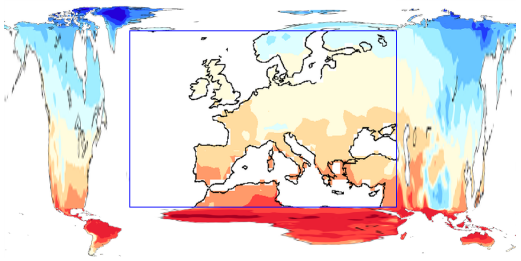
**RoI mit Fokus&Context** Die *RoI* wird in höherer Auflösung innerhalb der originalen *View* dargestellt. Den Kontext bildet die verzerrte Darstellung der außerhalb der *RoI* liegenden Bereiche der originalen *View* (siehe Abbildung 4.16d).

Diese drei Anzeigestrategien wurden für die Manipulation der Anzeige einer *View* umgesetzt. Für die Definition und Manipulation der *RoI* durch Presenter und Nutzer sind diverse Interaktionen notwendig: (1) die Definition einer *RoI* und (2) die Manipulation einer *RoI*.

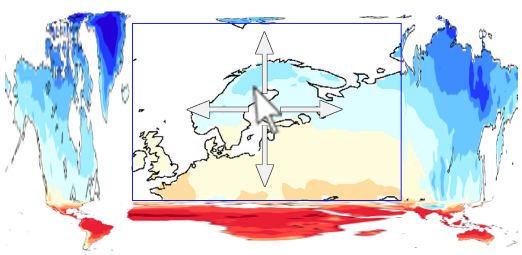
Zur *Definition einer RoI* wird durch einen Zeiger ein spezifischer rechteckiger Bereich einer *View* markiert. Diese so definierte *RoI* wird dann automatisch durch eine der drei Anzeigestrategien dargestellt. Als Standard wird hier die Anzeige der *RoI mit Fokus&Context* verwendet. Nach Rosenbaum [Ros06] zeigen *Fokus&Context* Techniken relevante Mikro- und Makroinformationen gleichzeitig in derselben Darstellung und eignen sich daher am besten für die Anzeige. Die Darstellung lässt sich interaktiv umschalten, so dass auch *RoI ohne Hintergrund* und *RoI mit Detail&Overview* zur Anzeige ausgewählt werden können.

Um die *RoI* dynamisch zu verändern und dem aktuellen Diskussionsgegenstand anzupassen, wird die *Zoom&Pan* Interaktion angeboten (siehe Abbildung 4.17).

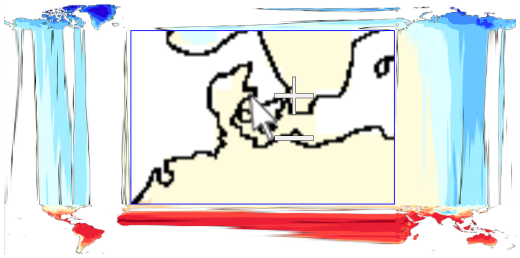
Auf Basis der *JPEG2000* Kodierung und mit Hilfe der Anzeigestrategien, der Definition und der Manipulation der *RoI* wird für den Presenter und die Nutzer die Möglichkeit geschaffen, die *View* Anzeige interaktiv zu manipulieren.



(a) *RoI* mit *Focus&Context*.



(b) *RoI* mit *Focus&Context*. Demonstration des *Panens*.



(c) *RoI* mit *Focus&Context*. Demonstration des *Zoomens*.

Abbildung 4.17.: *Pan&Zoom* demonstriert an *RoI* mit *Focus&Context* Darstellung.

Dabei muss nicht erst in den *View* generierenden Prozess eingegriffen werden, sondern es können on-the-fly alle dargestellten *Views* mit denselben Interaktionen manipuliert werden. Dies erweitert die derzeit bestehenden Möglichkeiten in einem Smart Meeting Room erheblich. Zu diesem Thema wird derzeit im Rahmen dieser Arbeit eine Publikation erstellt [RTNS14].

Trotzdem soll an dieser Stelle noch einen Schritt weiter gegangen und eine Interaktion zur Manipulation des Inhalts einer *View* entworfen werden. Dies wird auch von Waldner [Wal11] als ´´nächster logischer Schritt" bezeichnet.

## 4.4. Interaktion mit dem Inhalt der Views

In diesem Abschnitt wird die Interaktion mit dem Inhalt der *Views* adressiert. Nach der Identifikation der Probleme werden konkrete Lösungsansätze vorgestellt.

### 4.4.1. Problembeschreibung

Die Interaktion mit dem Inhalt der *Views* beschäftigt sich mit der interaktiven Anpassung der *Views* auf Ebene der *View* Generierung. Damit kann im Gegensatz zur Interaktion mit der Anzeige der *Views* auch das Dargestellte interaktiv modifiziert werden. Hierbei werden zwei Ziele verfolgt:

**Fokussierung von Inhalten** Die Manipulation der Anzeige, die im vorherigen Abschnitt besprochen wurde (vgl. Abschnitt 4.3), unterstützt die Anzeige verschiedener Genauigkeiten zur Fokussierung auf bestimmte Details einer *View*. Dies wird durch die Skalierung der *View* oder spezifischer Regionen der *View* erreicht.

Neben der Skalierung existieren aber noch weitere Methoden zur Fokussierung. Eine interessante Möglichkeit zur Fokussierung ist die Nutzung der Bewegung, da die Bewegung für den Menschen ein besonders herausstechendes Attribut der Wahrnehmung ist [HH05]. Deshalb wurde im Rahmen dieser Dissertation auch die Bewegung für die Fokussierung untersucht (siehe Abschnitt 4.4.2). Dies erfordert allerdings einen Eingriff in die *View* Generierung.

**Unterstützung der unterschiedlichen View Größen** In den vorherigen Abschnitten wurde die Skalierung von *Views* für verschiedene Zwecke verwendet – für die Darstellung von *Views* auf verschiedenen Displayflächen und für die Manipulation der Anzeige zur dynamischen Anpassung an die Bedürfnisse während einer Präsentation oder Diskussion. Durch die Skalierung können allerdings Informationen verloren gehen. So können z.B. durch das Vergrößern einer *View* mit einer *Scatterplot* Darstellung auf einer großen Displayfläche Punkte, die zu einem Cluster gehören, auseinandergezogen werden, so dass ihr Zusammenhang nicht mehr oder nur schwer erkennbar ist. Bei der Verkleinerung, etwa zur Ausgabe auf einer kleinen Displayfläche, können dagegen zusätzliche Überlagerungen auftreten, wenn z.B. Punkte eines *Scatterplots* durch die Verkleinerung auf eine Bildschirmkoordinate fallen, die vorher auf verschiedenen lagen. Ziel ist es deshalb, schon auf die *View* Generierung Einfluss zu nehmen, um die Darstellung der *View* bzgl. der Zielgröße anzupassen. Aus diesem Grund wurde im Rahmen dieser Arbeit das *Redundante Mapping* entwickelt (siehe Abschnitt 4.4.3).

Beide Ziele lassen sich durch eine Onlinemodifikation der *Views* umsetzen, wobei es sowohl dem Presenter als auch dem Nutzer erlaubt ist, mit jedem Interaktionsgerät für jeden *View* auf jeder Displayfläche in den Prozess der *View* Generierung einzugreifen.

Wie schon bei der Interaktion mit der Anzeige gilt auch bei dieser Modifikation die Anforderung, die Orientierung des Nutzers zu erhalten. Auf Basis der aktuellen Repräsentation schafft sich der Betrachter einen Überblick über die angezeigten *Views*. Um diesen Überblick und damit die Orientierung nicht zu verlieren, sollte demnach das grundsätzliche Erscheinungsbild der zu modifizierenden *View* erhalten bleiben. Wird beispielsweise ein Wechsel der Visualisierungstechnik vorgenommen (z.B. Austausch einer *Scatterplot Matrix* durch eine *parallele Koordinatendarstellung*), wird der Inhalt grundsätzlich anders dargestellt und die Orientierung des Betrachters geht verloren.

Daraus leitet sich die Anforderung für beide Ziele ab, die grundsätzliche Darstellung einer *View* zu erhalten, die Kommunikation des Inhalts aber zu verbessern. Dabei soll die Interaktion, wenn durch die entsprechende Applikation zugelassen, von Presenter und Nutzer gleichermaßen ausführbar sein.

### 4.4.2. Bewegung als visuelles Attribut

Die Bewegung ist beim Menschen ein hervorstechendes Merkmal der Wahrnehmung [HH05]. Aus diesem Grund eignet sie sich besonders für die Fokussierung der Aufmerksamkeit auf bestimmte Bereiche einer *View*.

Im Rahmen dieser Dissertation wurden drei Verfahren entwickelt, um das visuelle Attribut Bewegung einzusetzen. Diese wurden am Beispiel von *Scatterplots* genutzt, um für eine kurze Zeit die Aufmerksamkeit auf einen bestimmten Bereich zu lenken und damit die Wahrnehmbarkeit gezielt zu unterstützen. *Scatterplots* werden dafür verwendet, die Verteilung der Datenpunkte und deren Cluster zu analysieren. Wenn die Anzahl der Datenpunkte allerdings sehr groß und die Größe der *View* gering ist, verdecken sich die dargestellten Punkte untereinander. Dadurch wird es schwierig, die verschiedenen Cluster und ihre Form und Verteilung in einem *Scatterplot* zu unterscheiden.

Ziel des Einsatzes der Bewegung als visuelles Attribut in *Scatterplots* ist es, Presenter und Nutzer die Möglichkeit einer interaktiven Fokussierung zu ermöglichen. So kann z.B. während einer Diskussion der Presenter durch das interaktive Zuschalten der Bewegung den Fokus gezielt auf ein Cluster lenken und damit die Wahrnehmbarkeit dieses Clusters unterstützen. Die Bewegung kann auch interaktiv zugeschaltet werden, wenn beispielsweise die Unterscheidung von Clustern eingeschränkt ist, wie beispielsweise durch eine Verkleinerung einer *View*.

Um dies zu erreichen wurden drei verschiedene Arten der Bewegung verwendet: (1) *Jittering*, (2) *alternierende Weaving Muster* und (3) *scheinbare Bewegung*, die jeweils drei verschiedene Ziele verfolgen. Diese drei Techniken werden im Folgenden vorgestellt (vgl. [PRSB10]).

**Jittering** Das *Jittering* wird verwendet, um die Erkennbarkeit der Verteilung und Form eines ausgewählten Clusters zu unterstützen. Beim *Jittering* werden dafür die jeweiligen Datenpunkte geringfügig bewegt.

Um die Kommunikation der Form und Verteilung eines einzelnen Clusters zu verbessern, werden in einer kurzen Animation die jeweiligen Mitglieder eines Clusters um ihre ursprüngliche Position herum mit einer Frequenz von 17 Hz bewegt. Die Richtung der Verschiebung wird für jeden Datenpunkt individuell zufällig ermittelt. Eine sehr geringe Bewegungsweite von mindestens 20% der Größe des zu verschiebenden Objekts reicht dabei völlig aus, um eine wahrnehmbare Bewegung zu erreichen. Durch diese Bewegung wird eine Oszillation der einzelnen Punkte um ihre ursprüngliche Position herum erreicht.

Nach der Bewegung der Elemente eines Clusters können weitere Cluster auf dieselbe Art und Weise untersucht werden. Dadurch lassen sich jeweils Form und Verteilung eines spezifischen Clusters individuell analysieren.

Zu beachten ist dabei, das *Jittering* nicht dauerhaft zu verwenden, sondern interaktiv vom Nutzer aktivieren und deaktivieren zu lassen, um die Aufmerksamkeit der Nutzer nicht durch ein so starkes visuelles Attribut dauerhaft zu binden.

Zur Untersuchung der Nutzbarkeit des *Jitterings* wurde eine kleine Nutzerstudie durchgeführt. Es wurde ein Klimadatensatz mit ca. 40.000 Datenpunkten

verwendet, woraus zwei *Scatterplots* (A) mit 6 Clustern und (B) mit 8 Clustern generiert wurden. Dann wurden 14 Probanden mit Erfahrung in der Visualisierung dazu aufgefordert, die Anzahl der Cluster zu zählen. Um einen Lerneffekt zu verhindern, wurden die Probanden in zwei Gruppen geteilt. Gruppe 1 testete *Scatterplot* A ohne Bewegung und *Scatterplot* B mit Bewegung, Gruppe 2 testete genau anders herum. Die Hypothese der Studie war, dass durch den Einsatz von *Jittering* die Anzahl der Cluster mit höherer Erfolgsquote richtig eingeschätzt wird.

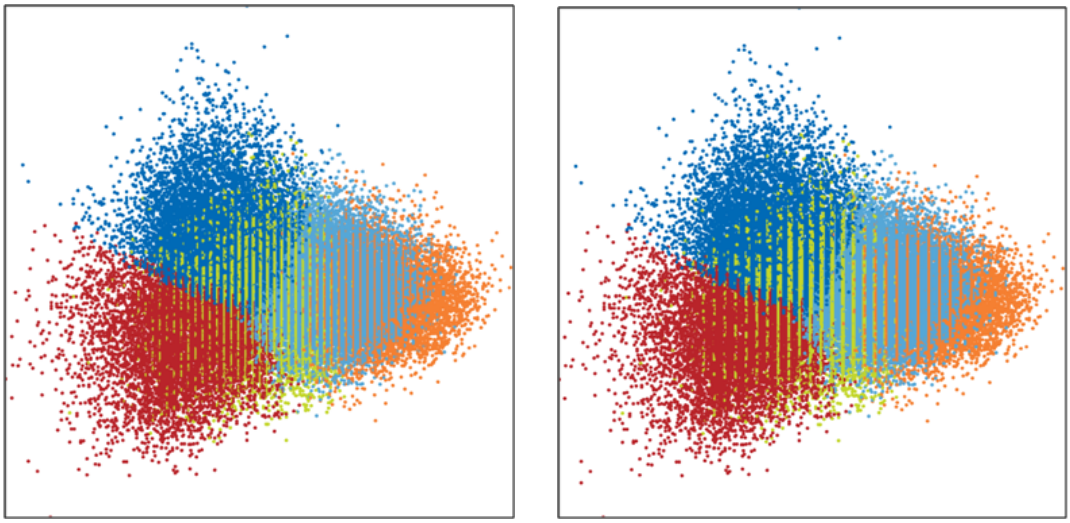
Das Ergebnis bestätigt die Hypothese der Studie. Bei der Befragung mit *Scatterplot* ohne Bewegung wurden bei *Scatterplot* A die 6 Cluster von ca. 71% der Probanden korrekt erkannt, bei *Scatterplot* B wurden die 8 Cluster nur zu ca. 28% korrekt erkannt. Wurde hingegen das *Jittering* verwendet, wurde die Anzahl der Cluster in allen Fällen korrekt erkannt.

Diese kleine Nutzerstudie kann zwar nicht als repräsentativ angesehen werden, gibt aber einen deutlichen Hinweis darauf, dass das *Jittering* die Erkennbarkeit der Cluster deutlich steigert.

**Alternierende Weaving Muster** Die Anzeige *alternierender Weaving Muster* ist die zweite Art der Bewegung. Wie bereits in Abschnitt 4.3.3.3 beschrieben, wird das *Weaving* verwendet, um die Verdeckung von *Views* zu behandeln.

Ziel der Nutzung *alternierender Weaving Muster* ist es nun, mit Hilfe dieses visuellen Attributs die Bereiche des *Scatterplots* hervorzuheben, an denen sich Cluster gegenseitig überdecken. Damit lassen sich auf einen Blick die Überlagerungsbereiche der Cluster gut erkennen.

Für den Anwendungsfall der *Scatterplots* wurde das *stack-aligned Weaving* ausgewählt (vgl. Abschnitt 4.3.3.3). Um nun eine Bewegung zu ermöglichen, wurde das *stack-aligned Weaving* Muster leicht modifiziert. Ursprünglich wurde für jede Spalte im Zielbild ein Stack der Ebenen erstellt, aus dem die Quellfarbe ausgewählt wird (siehe Abbildung 4.18a). Dieses Muster wurde dahingehend modifiziert, dass die Breite der Spalten angepasst werden kann. Dies wird dadurch erreicht, dass ein Stack nicht mehr nur für eine Spalte sondern für zwei oder mehr Spalten verwendet wird, bevor der Stack variiert wird (vgl. Abschnitt 4.3.3.3).



(a) (b)

Abbildung 4.18.: *Weaving* zur Erzeugung von Bewegung in den Bereichen der Überlagerung. Abbildung (a) zeigt das *stack-aligned Weaving* mit 1 Pixel breitem Muster, Abbildung (b) zeigt *stack-aligned Weaving* mit 2 Pixel breitem Muster. Eine alternierende Präsentation der Bilder erzeugt Bewegung im Bereich der Überlagerung.

Dadurch werden die Spalten breiter bei Anwendung des prinzipiell gleichen Musters (siehe Abbildung 4.18).

Die Bewegung wird nun dadurch erzeugt, dass zwei (oder mehr) *Weaving* Bilder mit verschiedenen Spaltenbreiten generiert und abwechselnd in einer Animation gezeigt werden. Die Anzeigezeit eines Bildes kann dabei interaktiv vom Nutzer eingestellt werden. Durch diese Animation wird im Bereich der Überlagerung Bewegung wahrgenommen.

Durch diese Art der Bewegung kann eine echte Bewegung angezeigt werden, ohne die Positionen der Datenpunkte im *Scatterplot* zu ändern.

Für die Anwendung des *Weavings* zur Erzeugung einer Bewegung wurde an dieser Stelle keine zusätzliche Nutzerstudie angefertigt. Zum einen wurde in [HSIHK06] das *Weaving* im Allgemeinen und zum anderen in [LRS10a] die neuen *Weaving* Muster untersucht (vgl. Abschnitt 4.3.3.3).

**Scheinbare Bewegung** Die *scheinbare Bewegung* durch die periphere Drift ist die dritte Art der Bewegung. Die periphere Drift ist eine optische Bewegungstäuschung, die durch eine falsche Interpretation durch das menschliche visuelle System hervorgerufen wird. Die menschliche Wahrnehmung erkennt helle Objekte zuerst und interpretiert dunkle Objekte danach. Ist ein Objekt mit einer hellen Kontur auf der einen, einer dunklen Kontur auf der anderen Seite und einer spezifischen Helligkeit in der Objektmitte versehen, führt dies zur Illusion dass die Objekte sich in eine bestimmte Richtung bewegen. Dieser Effekt wird durch die geringe Auflösung des peripheren Sichtfeldes des Menschen hervorgerufen [BO05].

Diese optische Täuschung wird nun dafür verwendet, eine Bewegung im *Scatterplot* zu kommunizieren. Um die periphere Drift zu erzeugen, wurden im *Scatterplot* die Punkte durch kleine Kreise ersetzt, die eine spezifische Kontur erhielten. Dabei wurden die Vorgaben von Backus et al. [BO05] verwendet. Hier wurde vorgeschlagen, die eine Kontur schwarz (0% Helligkeit), die entgegengesetzte Kontur weiß (100% Helligkeit) und das Objekt selbst mit einer Farbe mit 40% Helligkeit und den Hintergrund mit einer Farbe mit 80% Helligkeit (dafür wurde grau verwendet) zu kolorieren (siehe Abbildung 4.19a).

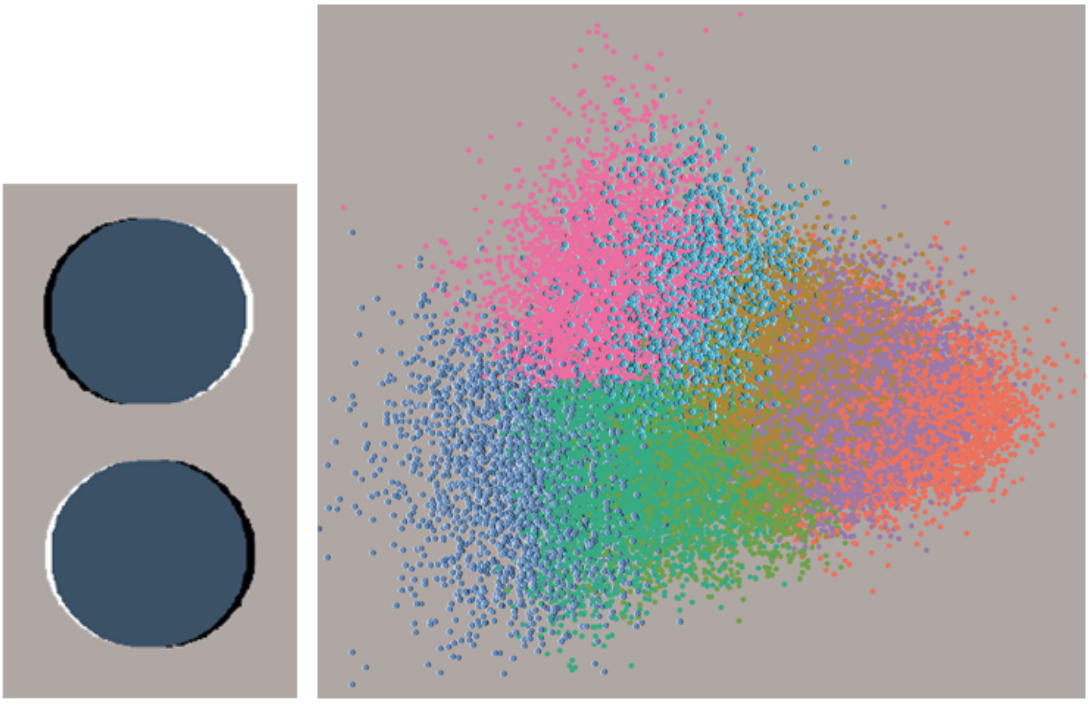
Für die Anwendung im *Scatterplot* können zwei bis vier Cluster ausgewählt werden, die dann mit diesem Effekt versehen werden können. Die Konturen der Mitglieder der Cluster werden dann bzgl. der gewünschten Bewegungsrichtung eingefärbt (siehe Abbildung 4.19b).

Bei der peripheren Drift gilt es allerdings zu beachten, dass die Wahrnehmung dieses Effekts von Mensch zu Mensch variiert. In [FW79] untersuchten Fraser und Wilcox dieses Phänomen. Das Ergebnis ihrer Untersuchungen zeigte bei 678 Testpersonen, dass 24,9% der Probanden keine Bewegung, 59% eine Bewegung von dunkel zu hell, 6,5% von hell zu dunkel und 9,6% abwechselnd eine Bewegung von hell zu dunkel und von dunkel zu hell sahen. Daher sollte die periphere Drift als Bewegung nur dann eingesetzt werden, wenn der Nutzer diesen Effekt auch wahrnehmen kann.

Für die Anwendung der scheinbaren Bewegung als visuelles Attribut wurde eine kleine Nutzerstudie durchgeführt. Es wurden 22 Probanden mit einem ähnlichen Setup wie beim *Jittering* getestet. Die Probanden wurden in zwei Gruppen geteilt. Bei dieser Studie wurden den Probanden *Scatterplots* mit 3 Clustern unterschieden durch verschiedene Grauwerte und 8 Cluster mit Farbe unterschieden gezeigt. Auch hier war die Hypothese, dass die Anwendung der scheinbaren Bewegung die Erfolgsquote bei der Erkennung der Clusteranzahl verbessert.

Im Ergebnis wurde eine ähnlich gute Verbesserung der Erfolgsrate bei der Nennung der Anzahl der Cluster erreicht. Sowohl bei 3 Clustern mit Grauwerten als auch bei 8 Clustern mit Farben verbesserte sich die Erfolgsquote um ca. 27%. Auch diese Studie kann nicht als repräsentativ angesehen werden, gibt aber einen Hinweis darauf, dass auch die scheinbare Bewegung eine Verbesserung der Erkennbarkeit ermöglicht.

Mit diesen drei Methoden zur Nutzung der Bewegung als visuelles Attribut in *Scatterplots* wird dem Presenter und den Nutzern die Möglichkeit eröffnet, spezifische Aspekte der Darstellung zu fokussieren. Mit dem *Jittering* wird die Verteilung und Form der Cluster hervorgehoben, durch die Nutzung *alternierender Weaving Muster* werden die Überlagerungsbereiche der Cluster fokussiert und durch den Einsatz der peripheren Drift kann die Unterscheidbarkeit der Cluster verbessert werden.



(a) Kolorierung der einzelnen Objekte.

(b) Anwendung der Kolorierung für die scheinbare Bewegung auf zwei Cluster im *Scatterplot*.

Abbildung 4.19.: Prinzip der scheinbaren Bewegung durch periphere Drift und der Anwendung im *Scatterplot*. Zur Erzeugung der scheinbaren Bewegung werden die Objekte wie folgt koloriert: Kontur schwarz (0% Helligkeit), die entgegengesetzte Kontur weiß (100% Helligkeit), Objektfarbe 40% Helligkeit und Hintergrundfarbe 80% Helligkeit.

Das visuelle Attribut Bewegung wurde an dieser Stelle in einem sehr speziellen Kontext getestet. Es wäre interessant, in künftigen Arbeiten weitere Darstellungstechniken mit Bewegung anzureichern und mit Hilfe des *Smart Interaction Managements* auszuwählen.

### 4.4.3. Redundantes Mapping

Im Rahmen dieser Arbeit wurde das *Redundante Mapping* entwickelt – nicht nur um einzelne Aspekte zu fokussieren, sondern auch um die Wahrnehmbarkeit von Informationen bei der Darstellung auf verschiedenen Displayflächen zu unterstützen.

In [TC05] wird das Problem der *Display Scalability* (Bildschirm Skalierbarkeit) beschrieben. Darunter verstehen Thomas und Cook die Herausforderung, Visualisierungstechniken und Interaktionstechniken zu entwickeln, die für verschiedenste Displayflächen von Powerwall bis PDA skalieren. Eines der Probleme der *Display Scalability* ist die konsistente Darstellung der gleichen Daten auf verschiedenen Displayflächen. Eine einfache Skalierung von *Views* reicht nicht immer aus, um dieses Problem zu lösen. Insbesondere kann es zu *Visual Clutter* (visuelles Durcheinander) kommen:

- Bei kleinen *Views* tritt *Visual Clutter* auf, da zu viele Daten auf zu wenig Displayfläche angezeigt werden sollen [ED07].
- Bei großen *Views* wird die Darstellungsfläche vergrößert, was dazu führen kann, dass visuelle Objekte auseinandergerissen werden. Dadurch kann der Zusammenhang zwischen den Daten verloren gehen oder Merkmale wie die Datendichte falsch interpretiert werden [BS04, BS05].

Bei der Interaktion mit der Anzeige der *Views* können nun genau diese Probleme auftreten. Werden *Views* verkleinert, kommt es zur Vermischung von Informationen, werden *Views* vergrößert, kann der Zusammenhang verloren gehen. Deshalb soll an dieser Stelle ein Verfahren eingeführt werden, das den Inhalt einer *View* so anpasst, dass diese Seiteneffekte nicht so stark auftreten.

Auch hier gilt wieder die Anforderung, dass die Orientierung des Nutzers erhalten bleiben muss. Dafür müssen die Informationen so dargestellt werden, dass das

ursprüngliche Erscheinungsbild kaum verändert, eine Verbesserung der Wahrnehmbarkeit aber trotzdem erreicht wird. Im Rahmen dieser Dissertation wurde hierfür das *Redundante Mapping* entwickelt [RLSS11].

Das Mapping spezifiziert die visuelle Kodierung der Daten und ist damit der wichtigste Schritt im Visualisierungsprozess. Hierbei werden visuelle Abstraktionen definiert, die graphisch repräsentiert werden. Die grundlegende Idee des *Redundanten Mappings* ist es nun, das Mapping in Schritte aufzuspalten:

**Primäre Mapping Funktion** Die primäre Mapping Funktion erzeugt die grundsätzliche Darstellung, so wie es der Intention der Visualisierung entspricht, z.B. erzeugt die primäre Mapping Funktion eine *Scatterplot* Darstellung. Dabei wird aber zusätzlich garantiert, dass die Größe der graphischen Elemente so angepasst wird, dass diese bezogen auf die Eigenschaft der Displayfläche und den Abstand des Nutzers garantiert wahrnehmbar sind.

**Sekundäre Mapping Funktion** In der sekundären Mapping Funktion werden zusätzliche visuelle Attribute genutzt, um die dargestellten Daten redundant zu kodieren und so die Wahrnehmbarkeit zu unterstützen. Während die Anpassung der Größe der graphischen Elemente automatisch berücksichtigt wird, lassen sich die zusätzlichen visuellen Attribute nach Bedarf interaktiv zuschalten.

Durch die Kombination dieser Mapping Funktionen wird eine einheitliche Darstellung geschaffen, welche die Sichtbarkeit der graphischen Elemente garantiert und die Eigenschaften der verschiedenen Displayflächen berücksichtigt.

Nachfolgend werden beide Mapping Funktionen genauer beleuchtet. Die Anwendung des *Redundanten Mappings* erfolgt am Beispiel von *Scatterplots* Den Hinweis auf die Anwendbarkeit dieser Mapping Strategie wird zudem mit Hilfe einer Nutzerstudie belegt.

#### 4.4.3.1. Primäre Mapping Funktion

Der erste Teil der primären Mapping Funktion entspricht dem klassischen Mapping Schritt in der Visualisierungspipeline [HM90]. Hier wird definiert, welche Datenwer-

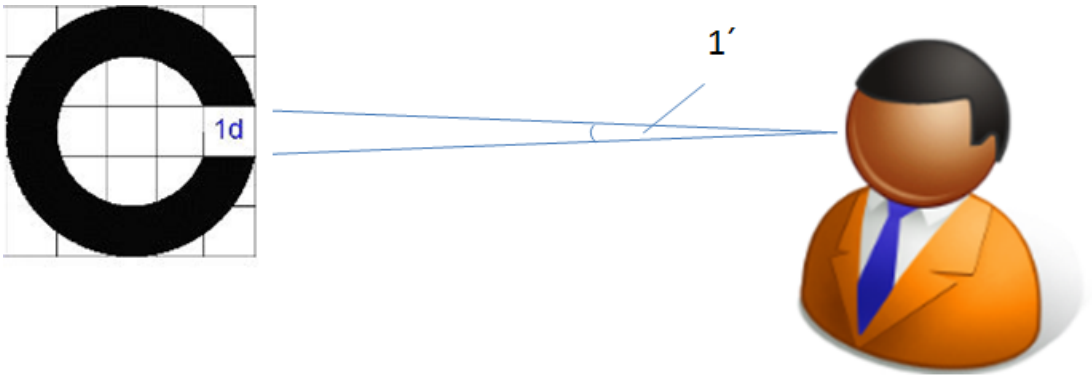



Abbildung 4.20.: Sehschärfe (Visus) von 1,0 ist gegeben, wenn ein Objekt mindestens eine Bogenminute vom Sichtfelds des Menschen überdeckt und dabei vom Hintergrund und anderen Objekten unterscheidbar ist.

te auf welche graphischen Elemente und in welcher Ausprägung abgebildet werden. Durch den ersten Schritt der primären Mapping Funktion wird die grundlegende Darstellung generiert, die dann weiter angepasst wird.

Der zweite Schritt der primären Mapping Funktion passt die Größe der graphischen Elemente so an, dass die Sichtbarkeit garantiert werden kann.

Die Adaption der Größe der graphischen Elemente basiert auf dem Verfahren zum Testen der Sehschärfe eines Menschen. Nach dem ISO Standard zur Sehschärfeprüfung [EN 96] ist bei einem Visus (Sehschärfe) von 1,0 ein Objekt vom Hintergrund und anderen Objekten unterscheidbar, wenn es mindestens eine Bogenminute ( $\frac{1}{60}^{\circ}$ ) des Sichtfelds des Menschen überdeckt (siehe Abbildung 4.20).

Daraus abgeleitet muss ein Objekt mindestens eine Bogenminute des Sichtfeldes eines Nutzers überdecken, um wahrnehmbar zu sein. Daher lässt sich die Mindestgröße eines graphischen Objekts ( $s$ ) wie folgt berechnen:

$$s = d \cdot \tan(\alpha) \tag{4.7}$$

Dabei ist $d$ der Abstand des Nutzers von der Displayfläche. Weiterhin ist ($\alpha = \frac{1}{60}^{\circ}$ der Winkel, den das Objekt vom Sichtfeld überdecken muss, um sichtbar zu sein.

Da eine Displayfläche nur eine diskrete Anzahl von Pixeln für die Darstellung eines Objekts nutzen kann, muss diese Mindestgröße auf die notwendige Anzahl von Pixeln abgebildet werden. Die Anzahl der notwendigen Pixel $p$ lässt sich folgendermaßen berechnen:

$$p \geq s \cdot r \tag{4.8}$$

Dabei ist $r$ die Pixeldichte der Displayfläche (in Pixel pro Zentimeter) und $s$ die zuvor berechnete Objektgröße (in Zentimeter). Die minimale Objektgröße entspricht demnach der nächsten Natürlichen Zahl größer als $p$. Wichtig ist hier, dass die Einheiten der einzelnen Faktoren beachtet werden. Distanzen werden in der Regel in Zentimetern angegeben, woraus folgt, dass auch die Objektgröße in Zentimetern berechnet wird. Die Pixeldichte bei Displayflächen wird dagegen in der Regel in *Pixel pro Inch* angegeben. Eine Umrechnung der Werte ist daher notwendig, um zu einem korrekten Ergebnis zu gelangen. Die Umrechnung Inch in Zentimeter ist $1in = 2,54cm$.

Außerdem wird die Sehstärke von der Helligkeit beeinflusst. Größere Helligkeiten erhöhen und niedrige Helligkeiten verringern dabei die Sehstärke. Dieser Zusammenhang wurde erstmals 1897 von Arthur König nachgewiesen [KÖ97]. Liegt die Lichtstärke allerdings bei mindestens 160 $cd/m^2$ wird die Sehstärke nicht negativ beeinflusst [Kau03]. Diese Helligkeit wird in der Regel von allen gängigen Displays deutlich übertroffen. Daher bedarf dieser Effekt keiner besonderen Berücksichtigung und die Formel 4.8 ist ausreichend.

#### 4.4.3.2. Sekundäre Mapping Funktion

Bei der sekundären Mapping Funktion werden Daten redundant enkodiert, um die Effektivität der visuellen Repräsentation auf verschiedenen Displayflächen zu erhöhen. Hier lassen sich prinzipiell alle visuellen Attribute nutzen, die bisher in der primären Mapping Funktion nicht verwendet wurden. Die sekundäre Mapping Funktion kann dabei nach Bedarf interaktiv zugeschaltet werden.

Zu beachten ist dabei allerdings, dass jede Visualisierungstechnik verschiedene visuelle Kodierungen verwendet. Damit bleibt für die sekundäre Mapping Funktion bei jeder Visualisierungstechnik eine unterschiedliche Menge visueller Attribute, die für die redundante Kodierung verwendet werden können. Weiterhin verlangen verschie-

dene Aufgaben und Ausgabegeräte unterschiedliche Kodierungen. Daher ist es nicht möglich, eine generelle Designrichtlinie vorzugeben. Die sekundären visuellen Attribute sollten auf Grundlage der verwendeten Visualisierungstechnik, also auf Grundlage der primären Mapping Funktion und des gegebenen Kontextes verwendet werden. Das lässt sich an dieser Stelle nicht für alle Visualisierungstechniken und verschiedenen Kontexte diskutieren.

Trotzdem kann ein grundsätzliches Vorgehen zur Auswahl von visuellen Attributen, die für die redundante Kodierung zur Verfügung stehen, angegeben werden. Als Grundlage kann die Liste der visuellen Attribute von Bertin [Ber81] ergänzt durch die Erweiterungen von Mackinlay [Mac86] verwendet werden:

- Position [position]
- Größe [size] (Länge [length], Fläche [area], Volumen [volume])
- Farbwert [color]
- Helligkeit [value]
- Form [shape]
- Orientierung [orientation] (Winkel [angle], Neigung [slope])
- Textur [texture]
- Dichte [desity]
- Verbindung [connection]
- Schachtelung [containment]

Hiervon werden durch die primäre Mapping Funktion gewisse visuelle Attribute genutzt, die für das sekundäre Mapping nicht mehr zur Verfügung stehen. Die restlichen visuellen Attribute können aber je nach gegebenem Kontext eingesetzt werden. Das soll im Folgenden an einem konkreten Beispiel der *Scatterplot* Darstellung diskutiert werden.

#### 4.4.3.3. Anwendung des Redundanten Mappings am Beispiel von Scatterplots

Hier wird exemplarisch die Anwendung des *Redundanten Mappings* an *Scatterplots* demonstriert. Die Aufgabe ist dabei die Clustererkennung. Dazu wird zunächst die primäre Mapping Funktion erklärt, gefolgt von der sekundären Mapping Funktion. Im Anschluss wird eine Nutzerstudie vorgestellt, die die Effektivität der redundanten Kodierung am genannten Beispiel untersucht.

**Primäre Mapping Funktion**

Im Beispiel der *Scatterplots* für die Darstellung von Clustern werden im primären Mapping zwei visuelle Attribute verwendet, um die grundlegende Darstellung zu erzeugen. Für jede der beiden visuellen Attribute wird jeweils eine Mapping Funktion definiert:

$p_1$ Bildet die Datenwerte der zwei darzustellenden Dimensionen auf eine Position ab.

$p_2$ Bildet die Clusterzugehörigkeit auf einen Farbwert ab.

Die primäre Mapping Funktionen $p_1$ erzeugt einen typischen *Scatterplot* und $p_2$ ermöglicht die Unterscheidung zwischen den Clustern (siehe Abbildung 4.21a). Im Beispiel wird der Farbwert für die Clusterzugehörigkeit verwendet, da dieses visuelle Attribut nach Mackinley [Mac86] für die Kodierung von nominalen Werten die größte Effektivität besitzt.

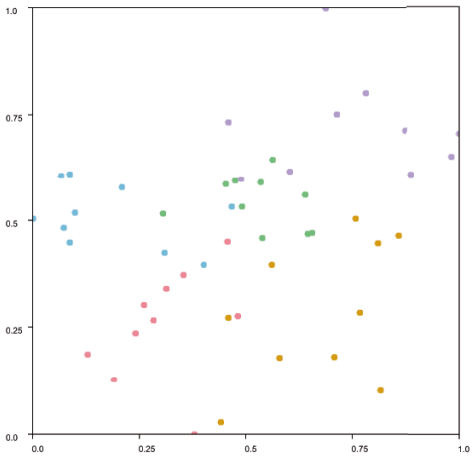
Um die Sichtbarkeit zu gewährleisten, wird weiterhin die primäre Mapping Funktion $p_3$ definiert:

$p_3$ Definiert die minimale Punktgröße abhängig von der Position des Nutzers (siehe Abschnitt 4.4.3.1).

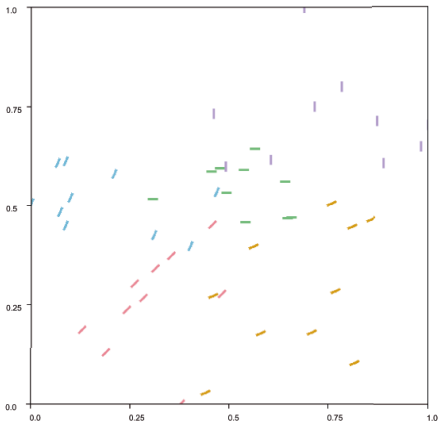
Auf diese Weise werden die visuellen Attribute Position, Farbe und Größe im primären Mapping verwendet.

**Sekundäre Mapping Funktion**

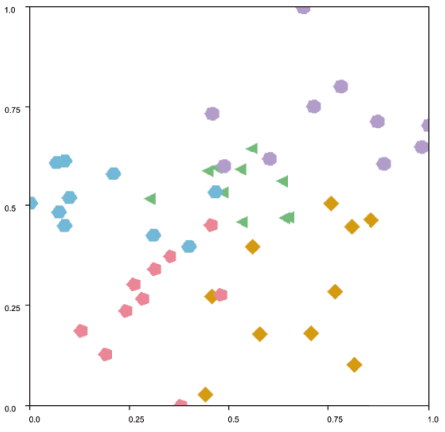
Für die sekundäre Mapping Funktion müssen nun visuelle Attribute identifiziert werden, die für eine redundante Kodierung der Clusterzugehörigkeit in Frage kom-



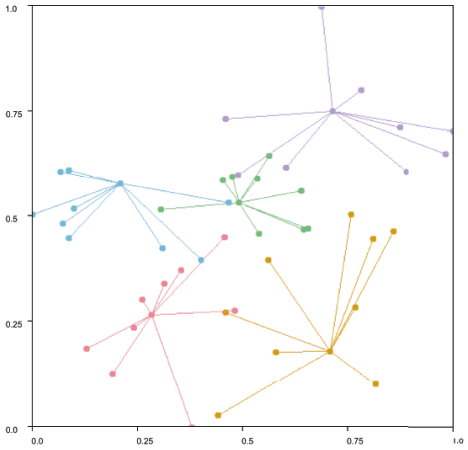

(a) Nur primäres Mapping.




(b) Redundantes Attribut: Orientierung.




(c) Redundantes Attribut: Form.




(d) Redundantes Attribut: Verbindung.

Abbildung 4.21.: *Scatterplot* mit unterschiedlicher redundanter Kodierung der Clusterzugehörigkeit.

men. Dafür werden im Folgenden zunächst die visuellen Attribute ausgeschlossen, bei der eine Kombination nicht in Frage kommt:

**Position, Farbwert und Größe** wurden bereits von den primären Mapping Funktionen $p_1$ bis $p_3$ verwendet und stehen daher für das sekundäre Mapping nicht mehr zur Verfügung.

**Helligkeit** beeinflusst die Wahrnehmung des Farbwerts. Eine redundante Kodierung wäre hier nicht als zusätzliches Attribut erkennbar.

**Textur, Schachtelung und Dichte** sollten auch ausgeschlossen werden, da die Größe der darzustellenden Objekte (Punktprimitive) erheblich zu klein ist. Eine zusätzliche Textur, Dichte oder Schachtelung wäre in diesem Fall nur schwer zu erkennen.

**Länge, Fläche, Volumen** Ebenso bringt die Unterteilung des visuellen Attributs Größe keine weiteren anwendbaren visuellen Attribute, da auch diese bei Punkten des *Scatterplots* nur sehr schwer unterscheidbar wären.

Nach diesem Ausschlussverfahren bleiben noch drei visuelle Attribute übrig, die in unserem Beispiel für eine redundante Kodierung verwendet werden können: Form, Orientierung und Verbindung. Hierfür werden die Mapping Funktionen $s_1$ bis $s_3$ eingeführt, die im Folgenden näher erläutert werden. Für die redundante Kodierung wird dann jeweils eine der drei sekundären Mapping Funktionen ausgewählt.

$s_1$**: Orientierung** Die Mapping Funktion $s_1$ mapped die Clusterzugehörigkeit auf das visuelle Attribut Orientierung. Da einzelne Punkte keine Orientierung besitzen, werden diese durch kleine Balken ersetzt. Die in $p_3$ berechnete minimale Größe wird hier auf die Breite des Balkens angewendet. Die Clusterzugehörigkeit wird auf einen Winkel $\gamma$ abgebildet, wobei $\gamma \in [0°, 90°]$. Die Balken werden dann bzgl. des $\gamma$ rotiert (siehe Abbildung 4.21b).

$s_2$**: Form** Die Mapping Funktion $s_2$ bildet die Clusterzugehörigkeit auf das visuelle Attribut Form ab. Die Punkte werden dafür durch regelmäßige Formen ersetzt, d.h. alle Seiten der Form haben die gleiche Länge und alle inneren Winkel sind gleich. Die Clusterzugehörigkeit wird auf die Anzahl der Eckpunkte abgebildet, wobei das Minimum der Ecken drei ist (d.h. ein Dreieck ist die Form mit der

kleinsten Anzahl an Eckpunkten). Der Datenwert wird durch den Mittelpunkt der Form repräsentiert (siehe Abbildung 4.21c).

$s_3$**: Verbindung** Die Mapping Funktion $s_3$ bildet die Clusterzugehörigkeit auf das visuelle Attribut Verbindung ab. Dabei verbindet $s_3$ die Punkte eines Clusters mit dem Zentroid dieses Clusters. Zur Berechnung des Zentroid wird der Punkt $P = (x, y)$ im *Scatterplot* berechnet, der das Zentrum des Clusters repräsentiert durch: $P = \left(\frac{x_{max}-x_{min}}{2} + x_{min}, \frac{y_{max}-y_{min}}{2} + y_{min}\right)$, wobei $x_{min}, y_{min}$ die minimalen $x$ bzw. $y$ Werte der Punkte im Cluster und $x_{max}, y_{max}$ die maximalen Werte darstellen. Der Zentroid ist dann der Punkt des Clusters, der die geringste euklidische Distanz zum Zentrum $P$ hat. Die einzelnen Punkte des Clusters werden mit dem Zentroid durch eine Linie verbunden (Abbildung 4.21d).

Die Resultate des *Redundanten Mappings* werden in Abbildung 4.21 dargestellt. Im folgenden Abschnitt wird anhand einer Nutzerstudie die Effektivität des *Redundanten Mappings* für verschiedene Displaygrößen untersucht.

#### 4.4.3.4. Nutzerstudie

Ziel der Nutzerstudie ist es, die Effektivität des *Redundanten Mappings* für verschiedene Displaygrößen am Beispiel der Clusterzugehörigkeit in *Scatterplots* zu untersuchen. Der Einsatz verschiedener Displaygrößen gibt dabei einen Hinweis darauf, bei welcher Kategorie von *View* Größen ob und wenn ja, welches Mapping verwendet werden sollte. Dafür wurden kleine, mittlere und große Displayflächen untersucht.

Zur Einschätzung der Resultate werden die gemessenen Erfolgsraten des *Redundanten Mappings* mit denen des klassischen *Scatterplots* verglichen. Die Größenanpassung (primäre Mapping Funktion $p_3$ ) wird dabei ebenfalls beim klassischen *Scatterplot* angewendet, um die Vergleichbarkeit zu gewährleisten. Im Vorfeld der Studie wurden zwei Hypothesen aufgestellt:

**H1** Die Kodierungen mit sekundärem Mapping liefern bessere Resultate auf den genutzten Displayflächen im Vergleich zum klassischen Mapping.

**H2** Die Displayflächen beeinflussen die Erfolgsrate des sekundären Mappings.

(a) Testgerät für die Nutzerstudie: iPod 1G.

(b) Testgerät für die Nutzerstudie: 61″ TV (Samsung HL61A750A1F).

Abbildung 4.22.: Displayflächen der Nutzerstudie.

Im Folgenden werden das Setup der Studie und die Resultate vorgestellt.

**Setup**

**Technische Basis** Es wurden drei Arten von Displayflächen verwendet: groß, mittel und klein. Für die Klasse der großen Displayflächen wurde ein Samsung HL61A750A1F TV mit 61″ und einer Auflösung von 1920x1080 Pixel (36 ppi (Pixel pro Inch)) verwendet (siehe Abbildung 4.22b). Das mittelgroße Display war ein 24″ Desktop Monitor mit einer Auflösung von 1920x1080 Pixel (94 ppi), wobei der präsentierte *Scatterplot* in einem 500x500 Pixel Fenster angezeigt wurde. Das kleine Display war ein Apple iPod touch 1G mit 3,5″ und einer Auflösung von 320x480 Pixel (163 ppi) (siehe Abbildung 4.22a). Der Betrachtungsabstand der Probanden wurde basierend auf einem Standard Nutzungsabstand aus [TQD09] festgelegt. Die Probanden waren 3m vor dem TV, 70cm vor dem Desktop Monitor und 40cm vor dem iPod touch positioniert.

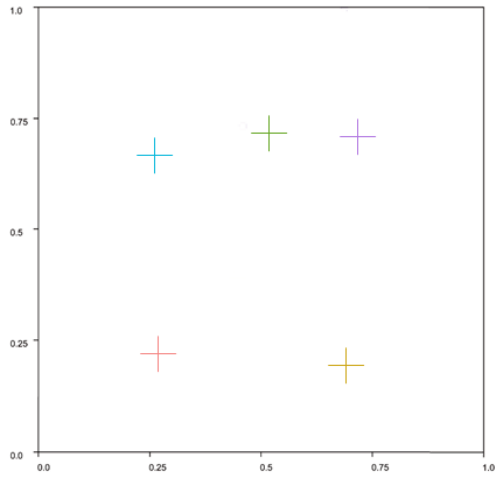
**Datenbasis** Als Datensatz wurden generierte Daten verwendet. Deren Erzeugung basiert auf zwei Maßen, die von Sips et al. vorgeschlagen wurden [SNLH09]. Diese beiden Maße werden eingesetzt, um die Trennbarkeit von Clustern in *Scatterplots* zu messen: der *Class Consistency* (Konsistenz der Klassen) und der *Class Density* (Klassendichte). Die *Class Consistency* gibt dabei an, wie weit die Zentroiden der Cluster voneinander entfernt sind, die *Class Density* hingegen, wie verstreut

die Mitglieder eines Clusters sind. Die Generierung der Daten lief wie folgt ab: Zuerst wurde die Anzahl der Cluster festgelegt. Dabei wurden in dieser Studie 4 bis 6 Cluster verwendet. Eine Vorstudie auf dem Desktop Monitor ergab, dass die Erfolgsquote bei der Erkennung von Clustern bei weniger als 4 zu hoch und bei mehr als 6 zu niedrig ist. Dann wurden die Positionen der Zentroiden festgelegt, indem verschiedene Werte bei der *Class Consistency* angenommen wurden (siehe Abbildung 4.23a). Um die Positionen der Punkte festzulegen, wurde eine *Class Density* festgelegt. Diese wird durch einen Radius um den Zentroiden repräsentiert. Durch den Radius und den Zentroiden wurde dann ein Kreis definiert (siehe Abbildung 4.23b). Die Punkte wurden dann zufällig in dem Kreis positioniert. Um die Vergleichbarkeit zu gewährleisten, wurden zum Schluss insgesamt 200 Datenpunkte verteilt (siehe Abbildung 4.23c). Für die Auswahl der Farben, die in der primären Mapping Funktion $p_2$ verwendet wurde, um die Clusterzugehörigkeit abzubilden, wurden sechs Farben definiert. Dazu wurde die Farbskala von Healey et al. [Hea96] genutzt, die nachgewiesener Weise eine sehr gute Farbskala für die Unterscheidung nominaler Daten ist.

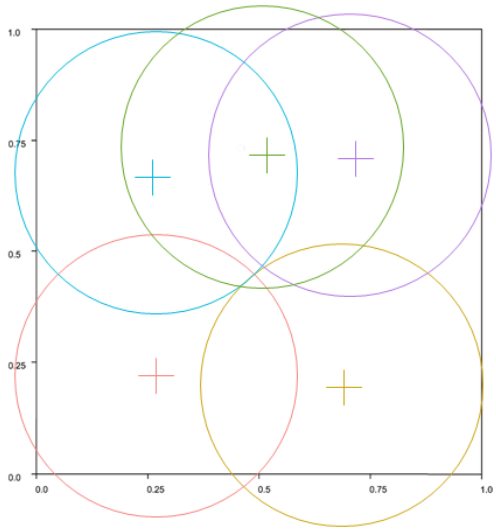
**Probanden** Die Probanden waren 23 Nichtvisualisierungsexperten, darunter 6 weibliche und 17 männliche mit einem Durchschnittsalter von ca. 34 Jahren, wobei der jüngste Teilnehmer 18 und der älteste 60 Jahre alt war.

**Aufgabe** Die Aufgabe bestand darin, die Anzahl der Cluster zu benennen, die in einem *Scatterplot* dargestellt wurden. Die Zeit war dabei beschränkt auf 15 Sekunden.

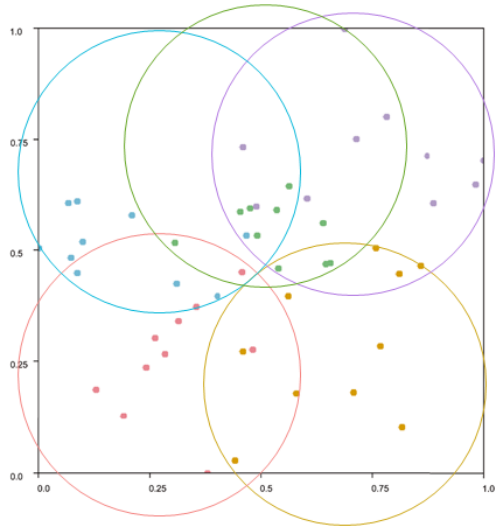
**Einweisung und Durchführung** Zur Einweisung wurden den Probanden Testbilder gezeigt und anhand derer die Aufgabe erklärt. Jedem Probanden wurden dann 20 *Scatterplots* pro Display Klasse mit verschiedenen Datenwerten gezeigt, dabei jeweils 5 pro visueller Kodierung (ohne *Redundantem Mapping*, mit Orientierung, mit Form und mit Verbindung als zusätzliches Attribut). Um einen Lerneffekt auszuschließen, wurden die Reihenfolgen der Displayflächen, Mappings und Anzahl der Cluster bei jedem Probanden zufällig variiert.



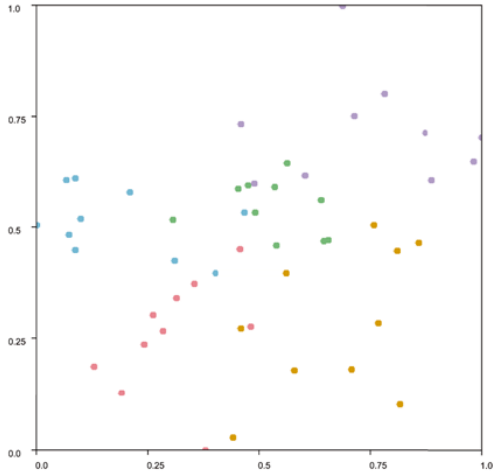

(a) Festlegung der Zentroiden der Cluster über das Maß der *Class Consistency*.




(b) Festlegung der Ausdehnung der Cluster über das Maß der *Class Density*.




(c) Gleichmäßige Verteilung der Datenwerte auf die verschiedenen Cluster innerhalb der Ausdehnung.




(d) Resultat der Generierung der Testdaten nach *Class Consistency* und *Class Density*.

Abbildung 4.23.: Ablauf der Generierung von Testdaten für die Nutzerstudie.

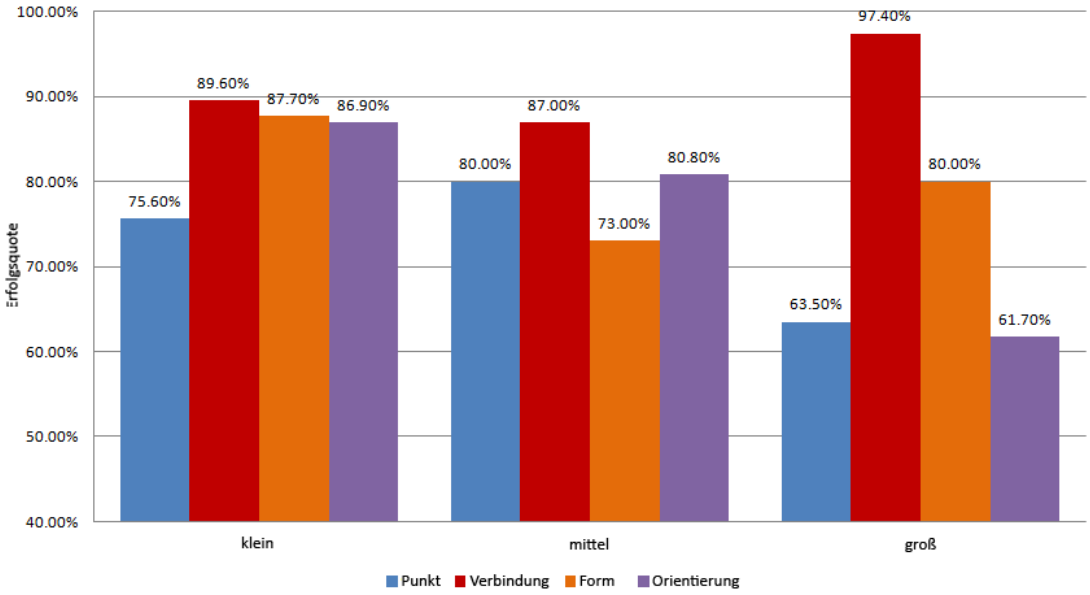



Abbildung 4.24.: Ergebnisse der Nutzerstudie zum *Redundanten Mapping*.

**Resultate**

Abbildung 4.24 zeigt die Erfolgsquoten der Studie. Im Folgenden werden die Ergebnisse, bezogen auf die im Vorfeld aufgestellten Hypothesen interpretiert. Auf eine Signifikanzanalyse wurde hier verzichtet, da die Stichprobenzahl nicht groß genug war, um eine signifikante Aussage treffen zu können. Die Ergebnisse geben allerdings einen deutlichen Hinweis.

**H1** Die primäre Mapping Funktion ohne zusätzliches *Redundantes Mapping* führte zu einer Erfolgsquote von 63,5% (große Displayfläche), 75,6% (kleine Displayfläche), und 80% (mittlere Displayfläche). Das *Redundante Mapping* verbesserte im Allgemeinen diese Erfolgsquoten erheblich. Bei kleinen Displayflächen wurde eine Verbesserung in jedem Fall gemessen (86,9% Orientierung, 87,8% Form, 89,6% Verbindung). Für mittle Displayflächen war Form etwas schlechter (73%), Orientierung war in etwa gleich gut (80,8%) und Verbindung war deutlich besser (87%). Bei großen Displayflächen war Orientierung in etwa gleich (61,7%), Form war deutlich besser (80%) und Verbindung war mit 97,4% außerordentlich

besser. Daraus lässt sich schlussfolgern, dass sich mit *Redundantem Mapping* eine Steigerung der Erkennbarkeit und damit eine Steigerung der Effektivität der Darstellung erreichen lässt. Diese erreichte Steigerung ist dabei abhängig von der Effektivität der einzelnen visuellen Attribute für die Darstellung der Daten. So entspricht die Steigerung der Erfolgsquoten in etwa der Reihenfolge der Effektivität der visuellen Attribute, wie sie Mackinlay [Mac86] beschreibt. Weiterhin lässt sich an den Ergebnissen ablesen, dass multiple Einflussfaktoren die Erfolgsquoten beeinflussen.

**H2** Die Ergebnisse zeigen (siehe Abbildung 4.24), dass Verbindung alle anderen Kodierungen in der Erfolgsquote übertrifft. Sie zeigen aber auch, dass es erhebliche Unterschiede bei der Anwendung auf verschiedenen Displaygrößen gibt. So schwankt die Erfolgsrate von Orientierung deutlich zwischen den Displayflächen.

**Diskussion**

Die Nutzerstudie gibt einen Hinweis darauf, dass die Anwendung des *Redundanten Mappings* die Effektivität der Darstellung unterstützt. Es ist auch deutlich geworden, dass multiple Einflussfaktoren die Ergebnisse beeinflussen. Daher wäre es wünschenswert, in Zukunft eine sehr umfangreiche Nutzerstudie durchzuführen, mit deren Hilfe alle Einflussfaktoren auf die Effektivität beschrieben und gemessen werden können.

Auf Basis der durchgeführten Studie lassen sich aber schon Hinweise für die Nutzung im speziellen Beispiel ableiten. So ist die Nutzung des visuellen Attributs *Verbindung* für die Kodierung der Clusterzugehörigkeit stets eine gute Wahl. Allerdings gilt es zu beachten, dass bei kleineren Displayflächen das *Overplotting* Problem zunimmt, d.h. es tritt mehr Verdeckung der graphischen Elemente untereinander auf. Da das visuelle Attribut *Verbindung* durch die zusätzlichen Linien noch weitere grafische Elemente hinzufügt, kann evtl. die Effektivität der Darstellung wiederum leiden.

Auf Basis der Studie lassen sich auch Hinweise für die interaktive Anpassung von *Views* mit *Scatterplots* ableiten, die im *Smart Interaction Management* realisiert wurden. Nach dem Vergrößern einer *View* wird die Nutzung der Verbindung als visuelles Attribut vorgeschlagen, die sich dann durch einen Mausklick umsetzen lässt. Wird eine *View* verkleinert, wird dagegen das zusätzliche Attribut Orientierung vorgeschlagen.

Bei der interaktiven Anpassung wird dem Presenter und dem Nutzer immer die Wahl zwischen den zusätzlichen Attributen gelassen. Auch das Ausschalten der zusätzlichen Kodierung wird angeboten.

### 4.4.4. Implementierung und Anwendung

#### 4.4.4.1. Implementierung

Die Implementierung des *Smart Interaction Managements* wurde auf die Implementierung des *Smart View Managements* aufgesetzt. Es wurde in Java [Ora13a] implementiert, um auch bei der Interaktion weitestgehende Unabhängigkeit vom verwendeten Betriebssystem zu erreichen. Beim *Smart Interaction Management* wurde ebenfalls die Helferlein Middleware [Ora13b] eingesetzt, um selbstorganisierende Softwarekomponenten zu ermöglichen.

Das *Smart Interaction Management* besteht aus drei Softwarekomponenten, die jeweils die funktionalen Komponenten widerspiegeln:

**Interaction Grabber** Der Interaction Gabber wertet die Interaktionen in syntaktischer Ebene aus und übersetzt sie in die Zwischenbeschreibung (Maus und Tastatur). Damit ist der *Interaction Grabber* die Schnittstelle des Interaktionsgeräts zum *Smart Interaction Management*. Für verschiedene Interaktionsgeräte sind dabei jeweils verschiedene *Interaction Grabber* notwendig. Für die Interaktionsgeräte, die nicht standardmäßig vom System erkannt werden (z.B. Maus und Tastatur), ist dabei in der Regel eine API notwendig, um die Interaktionen in syntaktischer Ebene zu registrieren. Für die Nutzung der Wiimote wurde im Rahmen dieser Dissertation die WiiuseJ API [Wii13] verwendet. Jeder *Interaction Grabber* registriert sich beim *Interaction Mapper* und erhält eine einzigartige ID, die mit einem separaten Zeiger assoziiert ist.

**Interaction Mapper** Der *Interaction Mapper* übernimmt zwei Funktionen. Zum einen wird damit die 2D Projektion für die Anordnung der Displayflächen erstellt. Zum anderen werden alle *Interaction Grabber* verwaltet.

Über die Middleware bekommt der *Interaction Handler* die Eigenschaften aller Displayflächen (Position, Ausrichtung, Größe) im Smart Meeting Room. Auf dieser Basis wird eine 2D Projektion erstellt, um eine relative Anordnung der Displayflächen zu erhalten. Ändern sich Eigenschaften, z.B. dass eine Displayfläche ausgeschaltet wird, kann dies on-the-fly berücksichtigt werden.

Die *Interaction Grabber* werden über eine einfache Liste verwaltet. Jeder *Interaction Grabber* erhält eine eindeutige ID. Außerdem wird jedem *Interaction Grabber* ein Zeiger im Raum zugeordnet. Dieser Zeiger ist wiederum auf genau einer Displayfläche. Der *Interaction Mapper* vermittelt die Kommunikation zwischen *Interaction Grabber* und der Displayfläche. Verlässt ein Zeiger eine Displayfläche, wird die nächste Displayfläche aus der relativen Anordnung berechnet und der Zeiger wird einer anderen Displayfläche zugeordnet.

Damit bildet der *Interaction Mapper* die Schnittstelle, um das Wissen über den Raum mit einzubeziehen. So ist der *Interaction Mapper* in der Lage, auf Änderungen dynamisch zu reagieren.

**Interaction Handler** Der *Interaction Handler* bildet die Schnittstelle zum *Smart View Management*. Die bisherigen Komponenten waren bzgl. der Implementierung unabhängig. Hier wird eine Schnittstelle für das *Smart View Management* geschaffen, um Interaktionen zu empfangen und auszuwerten. Außerdem ermöglicht der *Interaction Handler* die Interaktion mit der Anzeige der *Views* und die Interaktion mit dem Inhalt der *Views*. Dafür wurde der *Interaction Handler* sowohl in den *Metarenderer* als auch in die *View API* integriert.

Beim *Metarenderer* wird die Interaktion in Zwischenbeschreibung (Maus und Tastatur) direkt ausgeführt. Hier muss kein weiteres Mapping in eine Semantik stattfinden, da der *Metarenderer* die Semantik der Interaktion bereits in Maus und Tastatur Events beschreibt.

Bei der Integration in die V*iew API* wird eine Schnittstelle für die *View* generierenden Applikationen geschaffen, um *Views* Interaktionen zu empfangen. Integriert eine Applikation die Schnittstellen der API, so muss hier lediglich die Übersetzung von der Zwischenbeschreibung in die Semantik der Applikation implementiert werden.

Die Funktionalitäten der Anpassung der Anzeige von *Views* wurden im *Metarenderer* implementiert. Durch die Nutzung des *Interaction Handlers* kann auf die dort implementierten Interaktionen voll durch das *Smart Interaction Management* zugegriffen werden. Zur Manipulation der *JPEG2000* kodierten *Views* wurde die *kakadu SDK* verwendet [Sof13]. Die Funktionalitäten der Anpassung des Inhalts von *Views* wurden wiederum in separaten Applikationen implementiert.

#### 4.4.4.2. Anwendung

In [RNS11] wurde eine Anwendung des *Smart View Managements* in Zusammenarbeit mit Thomas Nocke vom Potsdamer Institut für Klimafolgenforschung vorgestellt (vgl. Abschnitt 3.7). Dabei wurde der *Vegetation Visualizer* [NHP+09] (PVV – Potsdam Vegetation Visualizer) verwendet, um *Views* zu generieren und mittels *Smart View Management* im Smart Meeting Room anzuzeigen.

Das *Smart Interaction Management* wurde im Rahmen dieser Kooperation ebenfalls eingesetzt. Neben der Möglichkeit der Interaktion für alle Nutzer mit allen Darstellungen und der Möglichkeit der interaktiven Anpassung der Anzeige der *Views* gab es zwei wesentliche Punkte der Interaktion, die hier fokussiert wurden: (1) Vergleichen der *Views* durch *Superimposition (überdecken)* mittels *Weaving* und (2) Manipulation der *Views* für die Fokussierung.

**Vergleichen von Views**

Eine wichtige Aufgabe in der Klimafolgenforschung ist es, verschiedene Informationen miteinander zu vergleichen, wie z.B. Attribute konkurrierender Klimamodelle oder verschiedene Attribute eines Klimamodells. Diese Informationen werden oft auf Karten visualisiert.

Das *Weaving* für die *Superimposition* von *Views* eignet sich gut für die Anforderungen der Klimafolgenforschung, da die Resultate der Klimamodelle mittels einer Farbskala auf den Kartendarstellungen visualisiert werden. Hier ist es von besonderem Interesse, dass Farben erhalten bleiben und klar einer *View* zugeordnet werden können.

**Manipulation der Views**

Für die bisherige Darstellung von Ergebnissen der Klimafolgenforschung wurde das PVV verwendet. Die damit generierten und dargestellten Bilder waren auf Grund der Anzeigemöglichkeiten in der Auflösung begrenzt. Trotz höherer Datenauflösung war es auch für die Darstellung im Smart Meeting Room mit Hilfe des *Smart View Managements* nicht möglich, höher aufgelöste Bilder anzuzeigen.

Durch die Manipulation der *Views* auf Basis der *JPEG2000* Enkodierung wird eine Darstellung höher aufgelöster Bilder ermöglicht. Darüber hinaus wird es den Nutzern ermöglicht, interaktiv Bereiche von Interesse auszuwählen und diese hochaufgelöst anzuzeigen und interaktiv zu manipulieren. Das ermöglicht es, Daten spezifischer in der Tiefe zu analysieren.

Durch den Einsatz des *Smart Interaction Managements* und den in diesem Kapitel vorgestellten Manipulationen der *Views* wird ein hohes Maß an Interaktivität in die Präsentation und Diskussion von Daten der Klimafolgenforschung gebracht. Daten können interaktiv diskutiert werden, Schwerpunkte in der Diskussion können dynamisch gesetzt und verändert werden und Daten können ohne spezifische Vorbereitung on-the-fly hochauflösend exploriert und diskutiert werden.

## 4.5. Zusammenfassung

In diesem Kapitel wurde die Interaktion im *Smart View Management* als Basis für die Interaktion mit *Views* im Smart Meeting Room vorgestellt.

Grundidee des *Smart Interaction Managements* ist es, die physikalische Interaktion von der Interpretation der Interaktion zu entkoppeln. Dazu wurden drei Komponenten verwendet: der *Interaction Grabber*, der *Interaction Mapper* und der *Interaction Handler*.

**Interaction Grabber** Wird auf dem Gerät verwendet, an dem das zu verwendende Interaktionsgerät angeschlossen ist. Dort wird die Interaktion ausgewertet, auf eine spezifische Zwischenbeschreibung abgebildet und zur Verfügung gestellt.

**Interaction Mapper** Weist diese Interaktion einer spezifischen Displayfläche zu und versendet die Interaktion zum *Metarenderer*, der auf dieser Displayfläche *Views* anzeigt.

**Interaction Handler** Ist dann zuständig für die Abbildung der spezifischen Zwischenbeschreibung auf die semantische Interpretation. Die Interpretation erfolgt dann entweder am *Metarenderer* selbst, wo die Anzeige der *Views* interaktiv verändert werden kann, oder die Interpretation erfolgt an der *View* generierenden Applikation, wo interaktiv die Generierung der *View* beeinflusst werden kann.

Durch das *Smart Interaction Management* werden neue Möglichkeiten der Interaktion mit *Views* bereitgestellt. Es ermöglicht **jedem Nutzer** eine Interaktion mit **allen dargestellten Views** auf **allen Displayflächen** mit **jedem Interaktionsgerät**.

Dabei werden nicht nur persönliche Geräte mit einbezogen, sondern auch neue Interaktionsgeräte integriert. Weiterhin unterstützt das *Smart Interaction Management* dabei die ´´Unsichtbarkeit" der Interaktion, d.h. dass Nutzer nicht wissen müssen, wo *Views* generiert wurden, mit denen sie interagieren wollen, und zur Interaktion muss der Ablauf nicht bekannt sein.

Auf der technischen und konzeptionellen Basis des *Smart Interaction Managements* wurden die Interaktion mit der Anzeige von *Views* und die Interaktion mit dem Inhalt der *Views* in diesem Kapitel thematisiert.

Bei der **Interaktion mit der Anzeige von Views** werden Methoden vorgestellt, die Anzeige im Smart Meeting Room zu manipulieren. Mit diesen Methoden werden sowohl der Presenter als auch alle Nutzer in die Lage versetzt, die Anzeige dynamisch auf sich verändernde Schwerpunkte und Bedürfnisse der Teilnehmer anzupassen.

Die Anzeige der *Views* besteht aus der Zuordnung der *Views* zu den Displayflächen und der Anordnung der *Views* auf den jeweiligen Displayflächen. Zur interaktiven Manipulation der Anzeige werden drei Aufgaben unterstützt:

- Interaktive Zuweisung von *Views* zu einer Displayfläche
- Interaktive Positionierung und Skalierung von *Views*
- Interaktive Anpassung der Darstellung von *Views*

Durch die hierfür entwickelten Methoden wird es für jeden Nutzer möglich, Anpassungen der Anzeige auf jeder Displayfläche und für jeden *View* durchzuführen. Es wurde die Möglichkeit geschaffen, *Views* mit einem einheitlichen Interface zu manipulieren unabhängig von der *View* generierenden Applikation und durch die Beschränkung auf lokale Änderungen kann die Orientierung der Nutzer bei Anpassungen erhalten bleiben.

Durch die **Interaktion mit dem Inhalt der Views** wird es allen Nutzern ermöglicht, in die Generierung von *Views* interaktiv einzugreifen. Auf Basis des *Smart Interaction Managements* wird auch dafür ein einheitliches Interface geschaffen.

In diesem Kapitel wurden zwei Ansätze vorgestellt, die eine Interaktion mit dem Inhalt von *Views* thematisieren. Dabei werden zwei grundsätzliche Ziele bei der Präsentation bzw. Diskussion unterstützt:

- Fokussierung von Inhalten
- Unterstützung unterschiedlicher *Views* Größen

Durch die Nutzung der Bewegung als visuelles Attribut wurde unter Verwendung eines hervorstechenden Merkmals der menschlichen Wahrnehmung die Möglichkeit geschaffen, auf spezifische Aspekte der Darstellung zu fokussieren.

Mit dem *Redundanten Mapping* wurde ein Konzept präsentiert, das es ermöglicht, mittels zusätzlicher visueller Attribute einzelne Aspekte einer *View* zu fokussieren und die Wahrnehmbarkeit der dargestellten Informationen zu unterstützen.

Durch das *Smart Interaction Management* als Basis für die Methoden zur Anpassung der Anzeige und des Inhalts von *Views* wird den Nutzern die Möglichkeit gegeben, mit allen *Views* auf allen Displayflächen zu interagieren, um sie den aktuellen Bedürfnissen und Schwerpunkten anzupassen.

# 5. Zusammenfassung und Ausblick

## 5.1. Zusammenfassung

In dieser Arbeit wird ein grundlegendes Konzept für die Informationspräsentation in Smart Meeting Rooms entwickelt. Durch das *Smart View Management* werden Generierung, automatische Verteilung und automatische Anordnung von *Views* ermöglicht. Das *Smart Interaction Management* ermöglicht weiterhin die Interaktion aller Nutzer mit allen *Views* unabhängig vom verwendeten Interaktionsgerät. Basierend auf diesen beiden Konzepten werden Methoden vorgestellt, die Anzeige der *Views* interaktiv zu manipulieren, um dynamischen Anforderungen von Präsentationen und Diskussionen gerecht zu werden. Weiterhin werden Methoden vorgestellt, den Inhalt der *Views* zu manipulieren. Dadurch kann jeder Nutzer in den Generierungsprozess der *Views* jedes Gerätes, wenn es freigegeben ist, eingreifen, die Wahrnehmbarkeit der in den *Views* dargestellten Informationen interaktiv anpassen oder spezifische Bereiche fokussieren.

Damit ist ein leistungsfähiges System entstanden, das eine Informationspräsentation in Smart Meeting Rooms unterstützt.

Mit dem *Smart View Management* wird ein Konzept zur Bildbasierten Informationspräsentation vorgestellt, das den aktuellen Stand der Forschung erweitert. Neben der Flexibilität und weitgehender Betriebssystemunabhängigkeit, die auch andere bereits präsentierte Systeme zur Informationsanzeige vorweisen können (z.B., [WJF+09]), wurde im *Smart View Management* durch das Konzept der *Views* eine konzeptionelle Betrachtung der anzuzeigenden visuellen Ausgaben unabhängig von der Anwendung und Art der anzuzeigender Informationen erreicht.

Das *Smart View Management* bietet außerdem eine automatische Konfiguration der Anzeige durch die automatische Zuweisung der *Views* zu den Displayflächen und das automatische Layout von *Views* auf den jeweiligen Displayflächen. Die grundlegenden Konzepte dafür finden sich zwar in der Literatur, für den Einsatz im *Smart View Management* mussten sie allerdings angepasst und weiterentwickelt werden. Eine vollständige Integration in ein System zur Informationspräsentation wurde nach Wissen des Autors bisher nicht vorgestellt.

Das *Smart Interaction Management* ermöglicht die Interaktion mit *Views* im Smart Meeting Room. Durch die Grundidee der Entkopplung der physischen Interaktion von der semantischen Interpretation wurde ein Konzept geschaffen, das den aktuellen Stand der Forschung um Optionen erweitert, gleichzeitig verschiedenste Interaktionsgeräte (Maus, Tastatur und neue Interaktionsgeräte) zu nutzen.

Dies bildet die Grundlage für die interaktive Anpassung der Anzeige und des Inhalts von *Views*.

Bei der Anzeige von *Views* wurde die Möglichkeit geschaffen, die Anzeige interaktiv anzupassen. So kann die Zuordnung der *Views* zu den Displayflächen modifiziert werden und die *Views* können auf den Displayflächen interaktiv positioniert und skaliert werden. Eine überlagernde Darstellung wird dabei mittels eines neuen *Weaving* Verfahrens ermöglicht. Auf Basis von *JPEG2000* können außerdem Regionen von Interesse zur Fokussierung spezifisch angezeigt werden. Damit werden hinsichtlich des Stands der Forschung bei der Anzeige keine neuen Methoden hinzugefügt. Vielmehr wird hier durch die Integration in die Informationsanzeige ein Interface geschaffen, das es unabhängig von der *View* erzeugenden Applikation ermöglicht, alle angezeigten *Views* von jedem Nutzer interaktiv den aktuellen Anforderungen der Präsentation oder Diskussion anzupassen.

Die Manipulation des Inhalts einer *View* für die Anzeige in einer Multi-Display-Umgebung wurde bereits von Waldner [Wal11] als offenes Forschungsthema identifiziert und als "nächsten logischen Schritt" der Anpassung bezeichnet. Im Rahmen dieser Dissertation werden zwei Ansätze vorgestellt, den Inhalt von *Views* anzupassen. Zum einen wird die Bewegung als visuelles Attribut eingesetzt, um spezifische Aspekte einer Scatterplott Darstellung zu fokussieren. Zum anderen wird das Konzept

des *Redundanten Mappings* vorgestellt, das nicht nur die Fokussierung erlaubt, sondern auch die Möglichkeit bietet, die Wahrnehmbarkeit der angezeigten Informationen zu unterstützen.

Weitere Forschung ist hier in vielfältigen Richtungen möglich und wünschenswert.

## 5.2. Ausblick

Im Folgenden wird ein Ausblick über die Informationspräsentation in Smart Meeting Rooms gegeben. Der Vorschlag für Themenstellungen nachfolgender Arbeiten orientiert sich dabei an der Struktur dieser Arbeit. Zuerst werden mögliche Erweiterungen auf technischer Ebene präsentiert, danach mögliche Erweiterungen zum Leistungsumfang von *Views* und Interaktionen. Am Schluß werden dann mögliche Themenstellungen für eine nutzerbasierte Anpassung vorgestellt.

**View Management**

Das *Smart View Management* umfasst (1) die Generierung von *Views*, (2) das Zusammenfassen von *Views* zu *View Packages*, (3) die automatische Zuweisung von *Views* zu Displays und (4) das automatische Anordnen von *Views* auf den Displayflächen.

Das Prinzip des *Smart View Managements* basiert auf der räumlichen Anordnung von *Views* im Smart Meeting Room. Eine automatisierte Abfolge verschiedener Konfigurationen ist dabei bisher nicht vorgesehen. Deshalb wäre es von Vorteil, wenn neben der räumlichen Verteilung der *Views* auch die zeitliche Anordnung untersucht würde. Diese Fragestellung wird derzeit im Rahmen des Graduiertenkollegs MuSAMA in der Folgepromotion von Christian Eichner untersucht. Ziel dieser neuen Promotion ist es unter anderem, eine Planungskomponente für den zeitlichen Ablauf zu entwickeln. Auf Grundlage dieser Planung lässt sich sowohl die Zuordnung der *Views* zu den Displayflächen als auch die Anordnung auf den Displayflächen beeinflussen.

Weiterhin wurde bisher der Inhalt von *Views* für die Verteilung und Anordnung im Smart Meeting Room nicht näher berücksichtigt. Ein erster Schritt in diese Richtung wurde durch die interaktive Gruppierung von *Views* zu *View Packages* realisiert. Hier werden *Views* interaktiv gruppiert, die inhaltlich zusammen gehören und ent-

sprechend in räumlicher Nähe angezeigt werden sollen. Für weitere Arbeiten wäre es wünschenswert, diese Gruppierung zu automatisieren. Dafür bedarf es allerdings einer automatischen Analyse der dargestellten Informationen oder vom Nutzer spezifizierter Metadaten der *Views*. So könnten z.B. die Art der visuellen Repräsentation (Text, Bild, Visualisierung, ...), der dargestellte Datensatz, die dargestellten Attribute oder der Bezugsraum analysiert und spezifiziert werden. Dadurch ließen sich nicht nur automatisch *View Packages* zusammenstellen, sondern auch die Anordnung der *Views* durch die automatische Ableitung von Constraints anpassen. Auf Basis solcher Metadaten könnte weiterhin der Inhalt von *Views* dynamisch verlinkt werden, was zusätzliche Möglichkeiten der Anpassung der Anzeige ermöglicht. So könnte z.B. visual Linking verwendet werden, um die Beziehungen von *Views* untereinander darzustellen. Grundlegende Ansätze für das visual Linking über Applikationen hinweg wurden bereits von Waldner et al. vorgestellt [WPL+10, SWS+11].

Auch zusätzliches Wissen über die Umgebung und den Nutzer könnten in Folgearbeiten dazu verwendet werden, die Anzeige von *Views* weiter anzupassen. So könnte z.B. die Vorhersage von Nutzerverhalten oder Wissen über die Umgebung die Konfiguration der Anzeige beeinflussen. Die Analyse und Prädiktion von Netzwerkauslastungen könnte z.B. in das Display Mapping einbezogen werden, um die Netzwerkauslastung zwischen *View* bereitstellenden Geräten und Displayfläche zu berücksichtigen.

Spätere Arbeiten können außerdem die Generierung von *Views* weiter ausbauen. So präsentieren Oehlberg et al. [OSJ+12] ein Interface, das es erlaubt, beliebige Dateien auf ein sogenanntes *drop target* zu ziehen. Dann wird die Datei lokal geöffnet und auf einem großen Display dargestellt. Eine Kombination dieses Prinzips mit der *View* Erzeugung im *Smart View Management* würde die Generierung von *Views* deutlich erweitern. Anzuzeigende Informationen, wie z.B. PDFs oder Slides aber auch Datensätze, die in Dateien gespeichert sind, würden auf ein *drop target* gezogen. In der Folge würde das *Smart View Management* eine Anfrage an alle *View* generierenden Applikationen stellen, ob sie aus dieser Datei eine oder multiple *Views* generieren können. Die Datei wird automatisch übers Netzwerk kopiert, von der Applikation werden die *Views* erzeugt und stehen dann zur Anzeige bereit. Damit wird für den Nutzer selbst die *View* generierende Applikation ´´unsichtbar". Durch die Selektion der Datei könnten somit *Views* generiert werden, alle Nutzer könnten mit dieser *View* interagieren und im Fall,

dass die Applikation dies anbietet, wäre selbst ein interaktiver Eingriff in die *View* Generierung möglich.

Für weitere Arbeiten besteht außerdem die Möglichkeit, das automatische Display Mapping und die automatischen Layouts des *Smart View Managements* zu erweitern. Neben der zusätzlichen zeitlichen Komponente, der Nutzung von Metadaten der *Views* und weiterem Wissen über Nutzer und Raum kann hier auch die aktuelle Aufgabe der Nutzer für die Steuerung der Anzeige verwendet werden. Im Rahmen dieser Dissertation wurde ein erster Schritt dazu in [RFS12] publiziert. Ziel war es hier, basierend auf einer Task Beschreibung die Anzeige von *Views* automatisch zu manipulieren. Dafür waren drei grundsätzliche Schritte notwendig: (1) Beschreibung der Aufgaben, (2) Auswahl der Aufgaben und (3) Konfiguration der Anzeige. Als Basis zur Beschreibung der Aufgaben wurde das Aufgabenmodell von Andrienko & Andrienko [AA05] verwendet. Hier werden Aufgaben in *elementare* und *synoptische* Aufgaben unterteilt. In Bezug auf das *Smart View Management* betreffen *elementare* Aufgaben einen einzelnen *View* während *synoptische* Aufgaben eine Menge von *Views* betreffen. Wie von Tominski et al. [TFS08] weiter ausgeführt wird, unterteilen sich diese Aufgaben weiter in *Vergleich*, *Identifikation* und *Lokalisation*:

Um diese Aufgaben zu spezifizieren werden drei Möglichkeiten vorgestellt: (1) automatisch, (2) semiautomatisch und (3) interaktiv.

*Automatisch* ist die weitgehendste Unterstützung. Dafür wird eine Workflow Beschreibungen als Grundlage vorausgesetzt. Sie beschreibt den Ablauf wie z.B. den eines Visual Analysis Meetings. Das auf Basis solch einer Beschreibung das Ableiten der aktuellen Aufgabe möglich ist, wurde unter anderem von Giersich et al. [GFF+07] und Burghardt et al. [BWB+11] gezeigt. Dabei wird z.B. die Position des Nutzers für die Erkennung genutzt. Ein großer Nachteil dieser automatischen Unterstützung ist, dass eine detaillierte Workflow Beschreibung vorhanden sein muss. Wird von diesem Ablauf abgewichen, z.B. während einer Diskussion, führt die automatische Konfiguration zu einer nicht erwünschten Anzeige. Daher sind die heutigen Ansätze zumeist auf lineare Workflows begrenzt, bei denen Aufgaben Schritt für Schritt ausgeführt werden [BWB+11]. Außerdem wird in [HK07] beschrieben, dass eine vollständig automatische Konfiguration Beobachtungen zu Folge von einigen Nutzern als zu aufdringlich empfunden wird.

*Semiautomatisch* bedeutet, dass der Nutzer die Aufgabe und die entsprechenden *Views* interaktiv spezifiziert, die Konfiguration der Anzeige dann aber automatisch ausgeführt wird.

*Interaktiv* bedeutet, dass die Konfiguration der *View* Anzeige vollständig interaktiv vom Nutzer konfiguriert wird.

An Hand der Aufgaben, unabhängig wie sie spezifiziert wurden, wird dann automatisch die Anzeige angepasst.

**Identifikation** *Views* werden im Layout automatisch größer dargestellt. Dadurch wird die *View* prominenter und erleichtert die Identifikation der Informationen.

**Lokalisation** Betroffene *Views* werden mit einem roten Rahmen dargestellt, was das Auffinden erleichtert.

**Vergleich** Zu vergleichende *Views* werden in ein *View Package* zusammengefasst. Mittels des anschließenden Display Mappings werden sie in räumlicher Nähe zueinander angezeigt.

Mit der aufgabenbasierten Anpassung der Anzeige wird ein erster Schritt für die automatische Anpassung geliefert. Durch weiteres Wissen und die Prädiktion des Verhaltens der Nutzer lassen sich hier in weiteren Arbeiten Anpassungen durchführen.

**Interaktion**

Im Rahmen des *Smart Interaction Managements* wurde zwar die konzeptionelle und technische Basis entwickelt, die sozialen Aspekte der Interaktion, vor allem bei der simultanen Interaktion multipler Nutzer, allerdings nicht thematisiert. Bei der Kollaboration von Nutzern gibt es in der Regel einen Konflikt bei der Privatheit von Inhalten, wobei zwischen privaten und öffentlichen Inhalten unterschieden wird. Dies beeinflusst nicht nur, welche Inhalte angezeigt werden sollen, sondern auch, wer wie damit interagieren sollte. In weitergehenden Arbeiten könnte daher untersucht werden, ob und wenn ja welche Systeme zur Zugriffskontrolle eingesetzt werden könnten. Solche Systeme werden z.B. in [BPG+11, FL08, SLC+00] diskutiert. Zum Einsatz bedarf es allerdings tiefgründiger Untersuchungen zur kollaborativen Arbeit in Smart Meeting Rooms und in typischen Smart Meeting Room Situationen.

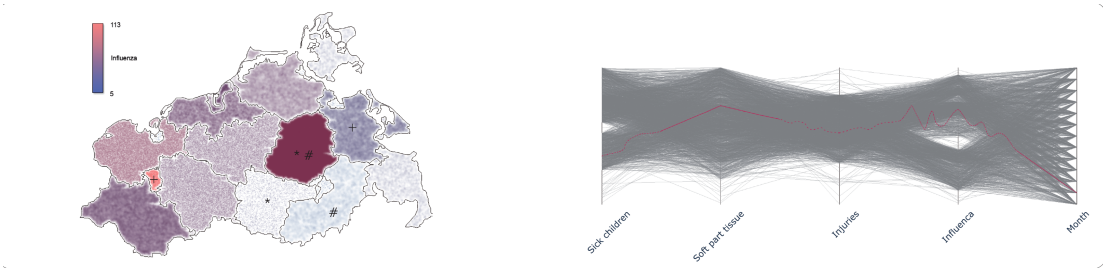



Abbildung 5.1.: Ergebnisse der Aquarell Simulation auf Karten (links) und der Anwendung von Strichelungen auf eine Parallele Koordinaten Darstellung. Beide Visualisierungen zeigen Krankheitsdaten in Mecklenburg/Vorpommern. Die Unsicherheiten *fehlende Daten* und *Standardabweichung* wurden jeweils in NPR Parameter kodiert.

Im Bereich der interaktiven Anpassung der Anzeige können ebenfalls weitere Arbeiten angeschlossen werden, um die Präsentation von Informationen zu erleichtern. Eine Möglichkeit dabei wäre, das Zeigen auf interessante Bereiche einer *View* zu erleichtern. Hier kann es durch die Anordnung der *Views* im *Smart View Management* dazu kommen, dass *Views* dupliziert auf verschiedenen Displayflächen dargestellt werden. Das macht das Zeigen von Information z.B. mit dem Zeiger des persönlichen Interaktionsgeräts schwierig. Eine Möglichkeit dies zu erleichtern wäre, den Zeiger auf alle identischen *Views* zu duplizieren. Dieses Prinzip ließe sich weiter ausbauen, indem die Metadaten der *Views* herangezogen werden. So kann das Duplizieren des Zeigers nicht nur auf identische *Views* angewendet werden, sondern auch auf *Views* z.B. mit dem gleichen Bezugsraum. Dies würde auch den Vergleich von *Views*, die in Juxtaposition (nebeneinander) angeordnet sind, erleichtern. Für die Fokussierung bzw. Anpassung des Inhalts von *Views* können neben der vorgestellten redundanten Kodierung von Informationen (vgl. Abschnitt 4.4.3) und der Anwendung von Bewegung (vgl. Abschnitt 4.4.2) weitere Techniken und visuelle Attribute verwendet werden. So kann z.B. das Non Photorealistic Rendering (NPR) dazu verwendet werden, zusätzliche Informationen zu kommunizieren. Ein Beispiel dazu wurde im Rahmen der Arbeit [LRS10b] vorgestellt. Hier wurden beispielhaft zwei NPR Techniken für die Kodierung von Unsicherheiten in Visualisierungen verwendet.

Zum einen wurde eine Aquarell Simulation [Sma91] verwendet, um Unsicherheiten auf einer Karte darzustellen. Dabei wurden die fehlenden Daten und die Standardabweichung der Daten der jeweiligen Regionen durch Parameter der Aquarell Simulation kodiert. Dazu mussten multiple Parameter der Aquarell Simulation untersucht werden. Abbildung 5.1 zeigt das Resultat der Aquarell Simulation.

Zum anderen wurden gestrichelte Linien dazu verwendet, die Unsicherheiten in parallelen Koordinaten darzustellen. Auch hier wurden multiple Parameter der Strichelungen untersucht. Die fehlenden Daten und die Standardabweichung der einzelnen Attribute in den parallelen Koordinaten wurden dann auf den Abstand der Strichelungen und die Amplitude des Ausschlags der Striche abgebildet (siehe Abbildung 5.1).

Diese beiden Beispiele zeigen, dass multiple Möglichkeiten existieren, NPR für die Kodierung von Daten zu verwenden. Eine aktuelle Dissertation von Martin Luboschik widmet sich diesem Thema ausführlich. Im Rahmen der Fokussierung und Anpassung von *Views* könnten weitere NPR Techniken untersucht werden. Auch die Anwendung weiterer Techniken ist hier denkbar.

Im Sinne der Konzepte, die in dieser Dissertation vorgestellt werden, ließen sich weitere in folgenden Arbeiten entwickeln. So könnte z.B. die Informationspräsentation durch ein Konzept zur Soundausgabe erweitert werden. Durch ein Smart Sound Management ließen sich nicht nur visuelle Ausgaben sondern auch akustische Ausgaben von Applikationen im Smart Meeting Room machen. Neben den im Smart Meeting Room angebrachten Lautsprechern ließe sich auch für Nutzer individualisierter Sound über beispielsweise Bluetooth-Kopfhörer ausgeben. Für ein Smart Sound Management bedarf es allerdings weitergehender konzeptioneller Arbeit.

# A. Bilder der Nutzerstudie zur Evaluation von Layout Mechanismen für die Anordnung von Views

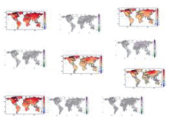
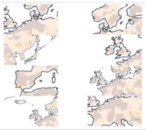
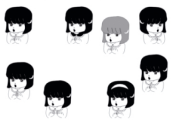
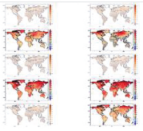
| Layout für | Zu testende Fragestellung | Erstelltes Layout |
|---|---|---|
| **Den Test und zum Üben des Ablaufes** | Zeigen sie auf das gelbe Objekt. |  |
| **Gitterbasiert Identifizieren Mädchen** | Zeigen Sie auf das Mädchen mit einem weit geöffneten Mund. |  |
| **Federkraftbasiert Vergleich Stumme Karte** | Zeigen Sie auf die beiden identischen Kartenausschnitte |  |
| **Federkraftbasiert Vergleich Klimakarten mit unterschiedlichen Skalen, die sich nicht ähneln** | Zeigen sie auf die Weltkarte mit der maximalsten Temperatur in Australien. |  |
| **Gitterbasiert Identifizieren Klimakartenausschnitte** | Zeigen Sie auf die Kartenausschnitte, auf denen Dänemark vollständig zu sehen ist. |  |
| **Gitterbasiert Identifizieren Stumme Karte** | Zeigen Sie auf den Kartenausschnitt, auf dem Spanien zu sehen ist. |  |
| **Federkraftbasiert Vergleich Mädchen** | Zeigen Sie auf die beiden identischen Mädchen. |  |
| **Gitterbasiert Identifizieren Klimakarten mit unterschiedlichen Skalen, die sich ähneln** | Zeigen sie auf die Weltkarte mit der maximalsten Temperatur in Afrika. |  |
| **Federkraftbasiert Identifizieren Klimakartenausschnitte** | Zeigen Sie auf die Kartenausschnitte auf denen Großbritannien vollständig zu sehen ist. |  |
| **Gitterbasiert Vergleich Stumme Karte** | Zeigen Sie auf die beiden identischen Kartenausschnitte |  |

Abbildung A.1.: Liste der Bilder aus der Nutzerstudie zur Evaluation von Layouts (aus [Hol13]).

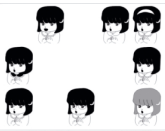
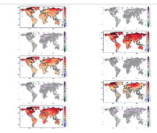
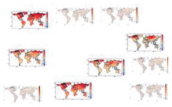
| | | |
|---|---|---|
| **Federkraftbasiert Identifizieren Mädchen** | Zeigen Sie auf das Mädchen mit einer Brille. |  |
| **Gitterbasiert Vergleich Klimakarten mit unterschiedlichen Skalen, die sich nicht ähneln** | Zeigen sie auf die Weltkarte mit der maximalsten Temperatur in Süd Amerika |  |
| **Federkraftbasiert Identifizieren Stumme Karte** | Zeigen Sie auf den Kartenausschnitt, auf dem Belgien zu sehen ist. |  |
| **Gitterbasiert Vergleich Mädchen** | Zeigen Sie auf die beiden identischen Mädchen |  |
| **Federkraftbasiert Vergleich Klimakarten mit unterschiedlichen Skalen, die sich ähneln** | Zeigen sie auf die Weltkarte mit der maximalsten Temperatur in Skandinavien. |  |

Abbildung A.2.: Liste der Bilder aus der Nutzerstudie zur Evaluation von Layouts (aus [Hol13]).

# Literaturverzeichnis


[AA05] ANDRIENKO, Natalia ; ANDRIENKO, Gennady: *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer-Verlag New York, Inc., 2005. – ISBN 3540259945

[Abo99] ABOWD, G. D.: Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment. In: *IBM Systems Journal* 38 (1999), Dezember, Nr. 4, S. 508–530

[AEa06] AARTS, E. H. L. ; ENCARNAÇÃO, J. L.: *True Visions: The Emergence of Ambient Intelligence*. Springer-Verlag New York, Inc., 2006. – ISBN 3540289720

[AHF+02] ARNSTEIN, Larry ; HUNG, Chia-Yang ; FRANZA, Robert ; ZHOU, Qing H. ; BORRIELLO, Gaetano ; CONSOLVO, Sunny ; SU, Jing: Labscape: A Smart Environment for the Cell Biology Laboratory. In: *IEEE Pervasive Computing* 1 (2002), Juli, Nr. 3, S. 13–21

[AHFS08] ALI, Kamran ; HARTMANN, Knut ; FUCHS, Georg ; SCHUMANN, Heidrun: Adaptive Layout for Interactive Documents. In: *Proceedings of the 9th International Symposium on Smart Graphics*, Springer-Verlag, 2008 (SG '08), S. 247–254

[Ali09] ALI, Kamran: *Adaptive Layout for Interactive Documents*, University of Rostock, Diss., 2009

[AP80] AXENFELD, Matthias ; PAU, Hans: *Lehrbuch und Atlas der Augenheilkunde*. Gustav Fischer Verlag, 1980

[ASWC04] ANDREWS, Christopher R. ; SAMPEMANE, Geetanjali ; WEILER, Andrew

; CAMPBELL, Roy H.: Clicky: User-centric input for active spaces. In: *University of Illinois AT Urbana-Champaign Dept. of CS Technical Report UIUCDCS-R-2004* 2469 (2004)

[ATK+05] AHLBORN, Benjamin A. ; THOMPSON, David ; KREYLOS, Oliver ; HAMANN, Bernd ; STAADT, Oliver G.: A practical system for laser pointer interaction on large displays. In: *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, 2005 (VRST '05), S. 106–109

[BB04] BIEHL, Jacob T. ; BAILEY, Brian P.: ARIS: An Interface for Application Relocation in an Interactive Space. In: *Proceedings of Graphics Interface 2004*, Canadian Human-Computer Communications Society, 2004 (GI '04), S. 107–116

[BB06] BIEHL, Jacob T. ; BAILEY, Brian P.: Improving Interfaces for Managing Applications in Multiple-device Environments. In: *Proceedings of the Working Conference on Advanced Visual Interfaces*, ACM, 2006 (AVI '06), S. 35–42

[BBB+08] BIEHL, Jacob T. ; BAKER, William T. ; BAILEY, Brian P. ; TAN, Desney S. ; INKPEN, Kori M. ; CZERWINSKI, Mary: Impromptu: A New Interaction Framework for Supporting Collaboration in Multiple Display Environments and Its Field Evaluation for Co-located Software Development. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2008 (CHI '08), S. 939–948

[BBM+06] BADI, Rajiv ; BAE, Soonil ; MOORE, J. M. ; MEINTANIS, Konstantinos ; ZACCHI, Anna ; HSIEH, Haowei ; SHIPMAN, Frank ; MARSHALL, Catherine C.: Recognizing User Interest and Document Value from Reading and Organizing Activities in Document Triage. In: *Proceedings of the 11th International Conference on Intelligent User Interfaces*, ACM, 2006 (IUI '06), S. 218–225

[BCHG04] BAUDISCH, Patrick ; CUTRELL, Edward ; HINCKLEY, Ken ; GRUEN, Robert: Mouse ether: accelerating the acquisition of targets across multi-monitor displays. In: *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, ACM, 2004 (CHI EA '04), S. 1379–1382

[Bea85] BEACH, Richard J.: *Setting tables and illustrations with style*, University of Waterloo Computer Science Department, Waterloo, Ontario, Canada, May 1985. Also issued as University of Waterloo Computer Science Department Technical Report CS-85-45 (May 1985) and Xerox Palo Alto Research Center Technical Report CSL-85-3 (May 1985), Diss., 1985

[Ber81] BERTIN, Jacques: *Graphics and Graphic Information-Processing*. de Gruyter, 1981

[BETT98] BATTISTA, Giuseppe D. ; EADES, Peter ; TAMASSIA, Roberto ; TOLLIS, Ioannis G.: *Graph drawing: algorithms for the visualization of graphs*. Prentice Hall PTR, 1998

[BF07] BENKO, Hrvoje ; FEINER, Steven: Pointer warping in heterogeneous multi-monitor environments. In: *Proceedings of Graphics Interface 2007*, ACM, 2007 (GI '07), S. 111–117

[BFLA02] BOOTH, Kellogg S. ; FISHER, Brian D. ; LIN, Chi Jui R. ; ARGUE, Ritchie: The MMighty MouseMMulti-screen Collaboration Tool. In: *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, ACM, 2002 (UIST '02), S. 209–212

[BG04] BAUDISCH, Patrick ; GUTWIN, Carl: Multiblending: displaying overlapping windows simultaneously without the drawbacks of alpha blending. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2004 (CHI '04), S. 367–374

[BK09] BURGHARDT, Christoph ; KIRSTE, Thomas: A Probabilistic Approach for Modeling Human Behavior in Smart Environments. In: *Proceedings of the 2Nd International Conference on Digital Human Modeling: Held As Part of HCI International 2009*, Springer-Verlag, 2009 (ICDHM '09), S. 202–210

[BKN05] BARATTO, Ricardo A. ; KIM, Leonard N. ; NIEH, Jason: THINC: A Virtual Display Architecture for Thin-client Computing, ACM, Oktober 2005, S. 277–290

[BMK+00] BRUMITT, Barry ; MEYERS, Brian ; KRUMM, John ; KERN, Amanda ; SHA-

FER, Steven A.: EasyLiving: Technologies for Intelligent Environments. In: *Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing*, Springer-Verlag, 2000 (HUC '00), S. 12–29

[BN12] BADER, Sebastian ; NYOLT, Martin: A context-aware publish-subscribe middleware for distributed smart environments. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on* IEEE, 2012, S. 100–104

[BO05] BACKUS, Benjamin T. ; ORUÇ, İpek: Illusory motion from change over time in the response to contrast and luminance. In: *Journal of Vision* 5 (2005), Nr. 11

[BPG+11] BRYDEN, Aaron ; PHILLIPS, George N. Jr. ; GRIGUER, Yoram ; MOXON, Jordan ; GLEICHER, Michael: Improving collaborative visualization of structural biology. In: *Proceedings of the 7th international conference on Advances in visual computing - Volume Part I*, Springer-Verlag, 2011 (ISVC'11), S. 518–529

[BRH+08] BURGHARDT, Christoph ; REISSE, Christiane ; HEIDER, Thomas ; GIERSICH, Martin ; KIRSTE, Thomas: Implementing Scenarios in a Smart Learning Environment. In: *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, IEEE Computer Society, 2008 (PERCOM '08), S. 377–382

[BRK10] BADER, Sebastian ; RUSCHER, Gernot ; KIRSTE, Thomas: A Middleware for Rapid Prototyping Smart Environments. In: *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*. Copenhagen, Denmark : ACM, SEP 2010, S. 355–356

[BRTF02] BORCHERS, Jan ; RINGEL, Meredith ; TYLER, Joshua ; FOX, Armando: Stanford interactive workspaces: a framework for physical and graphical user interface prototyping. In: *Wireless Communications, IEEE* 9 (2002), Nr. 6, S. 64–69

[BS04] BERTINI, Enrico ; SANTUCCI, Giuseppe: Quality metrics for 2d scatterplot graphics: automatically reducing visual clutter. In: *Proceedings of 4th In-*

*ternational Symposium on Smart Graphics, Banff, Canada* Springer, 2004 (SG '04), S. 77–89

[BS05] BERTINI, Enrico ; SANTUCCI, Giuseppe: Improving 2d scatterplots effectiveness through sampling, displacement, and user perception. In: *Proceedings of Ninth International Conference on Information Visualisation* IEEE, 2005, S. 826–834

[BWB+11] BURGHARDT, Christoph ; WURDEL, Maik ; BADER, Sebastian ; RUSCHER, Gernot ; KIRSTE, Thomas: Synthesising Generative Probabilistic Models for High-Level Activity Recognition. In: *Activity Recognition in Pervasive Intelligent Environments* Bd. 4. Atlantis Press, 2011, S. 209–236

[C+98] COEN, Michael H. u. a.: Design principles for intelligent environments. In: *Proceedings of the National Conference on Artificial Intelligence* JOHN WILEY & SONS LTD, 1998, S. 547–554

[CD04] COOK, Diane ; DAS, Sajal: *Smart environments: Technology, protocols and applications*. Bd. 43. Wiley-Interscience, 2004

[CD07] COOK, Diane J. ; DAS, Sajal K.: How Smart Are Our Environments? An Updated Look at the State of the Art. In: *Pervasive Mob. Comput.* 3 (2007), März, Nr. 2, S. 53–73

[CLB+03] CHIU, Patrick ; LIU, Qiong ; BORECZKY, John ; FOOTE, Jonathan ; FUSE, Tohru ; KIMBER, Don ; LERTSITHICHAI, Surapong ; LIAO, Chunyuan: Manipulating and annotating slides in a multi-display environment. In: *Proceedings of INTERACT* Bd. 3 Citeseer, 2003, S. 583–590

[CLWB01] CLAYPOOL, Mark ; LE, Phong ; WASED, Makoto ; BROWN, David: Implicit Interest Indicators. In: *Proceedings of the 6th International Conference on Intelligent User Interfaces*, ACM, 2001 (IUI '01), S. 33–40

[Com13] COMPIZ: *Compiz Fenstermanager*. http://www.compiz.org/, 2013. – (Zugriff 31.07.2013)

[CRKH09] CORBET, Jonathan ; RUBINI, Alessandro ; KROAH-HARTMAN, Greg: *Linux device drivers*. O'reilly, 2009

[CWM09] CHUANG, Johnson ; WEISKOPF, Daniel ; MOLLER, Torsten: Hue-Preserving Color Blending. In: *IEEE Transactions on Visualization and Computer Graphics* 15 (2009), November, Nr. 6, S. 1275–1282

[DE98] DIX, Alan ; ELLIS, Geoffrey: Starting simple: adding value to static visualisation through simple interaction. In: *Proceedings of the working conference on Advanced visual interfaces*, ACM, 1998 (AVI '98), S. 124–134

[DFAB03] DIX, Alan ; FINLAY, Janet E. ; ABOWD, Gregory D. ; BEALE, Russell: *Human-Computer Interaction (3rd Edition)*. Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 2003. – ISBN 0130461091

[DKQ13] DOSTAL, Jakub ; KRISTENSSON, Per O. ; QUIGLEY, Aaron: Subtle gaze-dependent techniques for visualising display changes in multi-display environments. In: *Proceedings of the 2013 international conference on Intelligent user interfaces* ACM, 2013, S. 137–148

[Dol07] DOLFING, H.: *A visual analytics framework for feature and classifier engineering*, University of Konstanz, Diplomarbeit, 2007

[EaK05] ENCARNAÇÃO, José L. ; KIRSTE, Thomas: From Integrated Publication and Information Systems to Virtual Information and Knowledge Environments. (2005), S. 261–270

[ED07] ELLIS, Geoffrey ; DIX, Alan: A taxonomy of clutter reduction for information visualisation. In: *Visualization and Computer Graphics, IEEE Transactions on* 13 (2007), Nr. 6, S. 1216–1223

[EN 96] Norm ISO 8596:1996-05 1996. *Augenoptik – Sehschärfeprüfung – Das Normsehzeichen und seine Darbietung, EN ISO 8596:1996-05*

[Fei98] FEINER, Steven: Readings in Intelligent User Interfaces. (1998), S. 249–255

[FES+06] FORLINES, Clifton ; ESENTHER, Alan ; SHEN, Chia ; WIGDOR, Daniel ; RYALL, Kathy: Multi-user, Multi-display Interaction with a Single-user, Single-display Geospatial Application. In: *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, ACM, 2006 (UIST '06), S. 273–276

[FFH00] FLACHSBART, Joshua ; FRANKLIN, David ; HAMMOND, Kristian: Improving human computer interaction in a classroom environment using computer vision. In: *Proceedings of the 5th international conference on Intelligent user interfaces*, ACM, 2000 (IUI '00), S. 86–93

[FL08] FORLINES, Clifton ; LILIEN, Ryan: Adapting a Single-user, Single-display Molecular Visualization Application for Use in a Multi-user, Multi-display Environment. In: *Proceedings of the Working Conference on Advanced Visual Interfaces*, ACM, 2008 (AVI '08), S. 367–371

[FR91] FRUCHTERMAN, Thomas M. J. ; REINGOLD, Edward M.: Graph Drawing by Force-directed Placement. In: *Softw. Pract. Exper.* 21 (1991), November, Nr. 11, S. 1129–1164. – ISSN 0038–0644

[FTS+10] FUKAZAWA, Ryo ; TAKASHIMA, Kazuki ; SHOEMAKER, Garth ; KITAMURA, Yoshifumi ; ITOH, Yuichi ; KISHINO, Fumio: Comparison of multimodal interactions in perspective-corrected multi-display environment. In: *Proceedings of the 2010 IEEE Symposium on 3D User Interfaces*, IEEE Computer Society, 2010 (3DUI '10), S. 103–110

[FW79] FRASER, Alex ; WILCOX, Kimerly J.: Perception of illusory movement. In: *Nature* 281 (1979), S. 565–566

[GDI13] GDI, Windows: *Windows GDI (Graphics Device Interface)*. http://msdn.microsoft.com/en-us/library/aa969176.aspx, 2013. – (Zugriff 30.04.2013)

[GFF+07] GIERSICH, Martin ; FORBRIG, Peter ; FUCHS, Georg ; KIRSTE, Thomas ; REICHART, Daniel ; SCHUMANN, Heidrun: Towards an integrated approach for task modeling and human behavior recognition. In: *Proceedings of the 12th international conference on Human-computer interaction: interaction design and usability*, Springer-Verlag, 2007 (HCI'07), S. 1109–1118

[GK07] GIERSICH, Martin ; KIRSTE, Thomas: Effects of Agendas on Model-based Intention Inference of Cooperative Teams. In: *Proceedings of the 2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing*, IEEE Computer Society, 2007 (COLCOM '07), S. 456–463

[Gmb10] GMBH, Future-Shape: SensFloor / Future-Shape GmbH. 2010. – Forschungsbericht

[Goo13] GOOGLE: *Google Earth*. http://earth.google.com/, 2013. – (Zugriff 03.05.2013)

[HBGK09] *Kapitel* Model-based Inference Techniques for detecting High-Level Team Intentions. In: HEIN, Albert ; BURGHARDT, Christoph ; GIERSICH, Martin ; KIRSTE, Thomas: *Ambient Intelligence and Smart Environments*. Bd. 3: *Model-based Inference Techniques for detecting High-Level Team Intentions*. IOS Press, 2009, S. 257 – 288

[HBL+06] HALLER, Michael ; BRANDL, Peter ; LEITHINGER, Daniel ; LEITNER, Jakob ; SEIFRIED, Thomas ; BILLINGHURST, Mark: Shared design space: Sketching ideas using digital pens and a large augmented tabletop setup. In: *Advances in artificial reality and tele-existence* Bd. 4282. Springer, 2006, S. 185–196

[HBR+09] HALLER, Michael ; BRANDL, Peter ; RICHTER, Christoph ; LEITNER, Jakob ; SEIFRIED, Thomas ; GOKCEZADE, Adam ; LEITHINGER, Daniel: Interactive displays and next-generation interfaces. In: *Hagenberg Research*. Springer, 2009, S. 433–472

[Hea96] HEALEY, Christopher G.: Choosing Effective Colours for Data Visualization. In: *Proceedings of the 7th Conference on Visualization '96*, IEEE Computer Society Press, 1996 (VIS '96), S. 263–ff.

[Hei09] HEIDER, Thomas: *A Unified Distributed System Architecture for Goal-based Interaction with Smart Environments*, University of Rostock, Diss., 2009

[HH05] HUBER, Daniel E. ; HEALEY, Christopher G.: Visualizing Data with Motion. In: *IEEE Visualization*, 2005 (VIS 2005), S. 67

[HK05a] HEIDER, Thomas ; KIRSTE, Thomas: Multimodal Appliance Cooperation Based on Explicit Goals: Concepts & Potentials. In: *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies*, ACM, 2005 (sOc-EUSAI

'05), S. 271–276

[HK05b] HEIDER, Thomas ; KIRSTE, Thomas: Smart environments and self-organizing appliance ensembles. In: *Mobile Computing and Ambient Intelligence* 5181 (2005)

[HK07] HEIDER, Thomas ; KIRSTE, Thomas: Automatic vs. manual multi-display configuration: a study of user performance in a semi-cooperative task setting. In: *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it - Volume 2*, British Computer Society, 2007 (BCS-HCI '07), S. 43–46

[HK08] HEIDER, Thomas ; KIRSTE, Thomas: Evaluating the effect of automatic display management on user performance in a Smart Meeting Room. In: *Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2008), München,Germany*, 2008

[HM90] HABER, Robert B. ; MCNABB, David A.: Visualization idioms: A conceptual model for scientific visualization systems. In: *Visualization in scientific computing* 74 (1990), S. 93

[HNC+09] HERVÁS, Ramón ; NAVA, SalvadorW. ; CHAVIRA, Gabriel ; VILLARREAL, Vladimir ; BRAVO, José: PIViTa: Taxonomy for Displaying Information in Pervasive and Collaborative Environments. In: *3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008* Bd. 51, Springer, 2009 (Advances in Soft Computing), S. 293–301

[HNJ+06] HARRINGTON, Steven J. ; NAVEDA, Jose F. ; JONES, Rhys P. ; SARR, Nathan ; THAKKAR, Nishant A. ; ROETLING, Paul G.: *System and method for measuring and quantizing document quality*. November 14 2006. – Google Patents

[Hol13] HOLZ, Anne: *Evaluation von Layoutmechanismen für die Anordnung von Views im Smart Meeting Room*. Studienarbeit, Universität Rostock, 2013

[hom13] HOMEPAGE, Musama: *Multimodal Smart Appliance Ensembles for Mobile Applications*. http://www.musama.de, 2013. – (Zugriff 06.05.2013)

[HSIHK06] HAGH-SHENAS, Haleh ; INTERRANTE, Victoria ; HEALEY, Christopher ; KIM, Sunghee: Weaving versus blending: a quantitative assessment of the information carrying capacities of two alternative methods for conveying multivariate data with color. In: *Proceedings of the 3rd symposium on Applied perception in graphics and visualization*, ACM, 2006 (APGV '06), S. 164–164

[IBR+03] IZADI, Shahram ; BRIGNULL, Harry ; RODDEN, Tom ; ROGERS, Yvonne ; UNDERWOOD, Mia: Dynamo: A Public Interactive Surface Supporting the Cooperative Sharing and Exchange of Media. In: *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, ACM, 2003 (UIST '03), S. 159–168

[JE12] JAVED, Waqas ; ELMQVIST, Niklas: Exploring the Design Space of Composite Visualization. In: *Proceedings of the IEEE Pacific Symposium on Visualization*, 2012 (PacificVis), S. 1–8

[JF02] JOHANSON, Brad ; FOX, Armando: The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In: *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, IEEE Computer Society, 2002 (WMCSA '02), S. 83–

[JFW02] JOHANSON, Brad ; FOX, Armando ; WINOGRAD, Terry: The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. In: *IEEE Pervasive Computing* 1 (2002), April, Nr. 2, S. 67–74

[JFW04] JOHANSON, Brad ; FOX, Armando ; WINOGRAD, Terry: *The Stanford Interactive Workspaces Project*. 2004

[JHW00] JOHANSON, Brad ; HUTCHINS, Greg ; WINOGRAD, Terry: Pointright: A system for pointer/keyboard redirection among multiple displays and machines / tech. report CS-2000-03, Stanford Univ. 2000. – Forschungsbericht

[JLS+03] JACOBS, Charles ; LI, Wilmot ; SCHRIER, Evan ; BARGERON, David ; SALESIN, David: Adaptive grid-based document layout. In: *ACM Transactions on Graphics* 22 (2003), Nr. 3, S. 838–847

[Jmo13] JMOL: *Jmol: an open-source Java viewer for chemical structures in 3D*. http://jmol.sourceforge.net/, 2013. – (Zugriff 03.05.2013)

[KÖ97] KÖNIG, Arthur: Die Abhängigkeit der Sehschärfe von der Beleuchtungsintensität. In: *Sitzungsbericht der Königlich Preussischen Akademie der Wissenschaften zu Berlin 26 (1897) pp. 559 – 575*. 1897

[Kau03] KAUFMANN, Herbert: *Strabismus*. Georg Thieme Verlag, 2003

[KHG03] KOSARA, Robert ; HAUSER, Helwig ; GRESH, Donna L.: An interaction view on information visualization. In: *State-of-the-Art Report. Proceedings of EUROGRAPHICS* (2003)

[KI07] KURIHARA, Kazutaka ; IGARASHI, Takeo: A Flexible Presentation Tool for Diverse Multi-display Environments. (2007), S. 430–433

[KRC02] KINDLMANN, Gordon ; REINHARD, Erik ; CREEM, Sarah: Face-based luminance matching for perceptual colormap generation. In: *Proceedings of IEEE Visualization 2002* IEEE, 2002 (VIS 2002), S. 299–306

[KS05] KARAM, Maria ; SCHRAEFEL, M. C.: A Taxonomy of Gestures in Human Computer Interactions. (2005)

[Lee13] LEE, Johnny C.: *Wii Projects*. http://johnnylee.net/projects/wii/, 2013. – (Zugriff 05.08.2013)

[LES95] LÜDERS, Peter ; ERNST, Rolf ; STILLE, Stefan: An approach to automatic display layout using combinatorial optimization algorithms. In: *Software: Practice and Experience* 25 (1995), Nr. 11, S. 1183–1202

[LGMLC01] LE GAL, Christophe ; MARTIN, Jérôme ; LUX, Augustin ; CROWLEY, James L.: Smart Office: Design of an Intelligent Environment. In: *IEEE Intelligent Systems* 16 (2001), Juli, Nr. 4, S. 60–66

[LKZH05] LIU, Qiong ; KIMBER, Don ; ZHAO, Frank ; HUANG, Jeffrey: Framework for effective use of multiple displays. In: *Optics East 2005* International Society for Optics and Photonics, 2005, S. 60150W–60150W

[LRS10a] LUBOSCHIK, Martin ; RADLOFF, Axel ; SCHUMANN, Heidrun: A New

Weaving Technique for Handling Overlapping Regions. In: *Proceedings of the International Conference on Advanced Visual Interfaces*, ACM, 2010 (AVI '10), S. 25–32

[LRS10b] LUBOSCHIK, Martin ; RADLOFF, Axel ; SCHUMANN, Heidrun: Using NPR-Rendering Techniques for the Visualization of Uncertainty. In: *IEEE Information Visualization Poster*, 2010 (InfoVis'10)

[LS12] LEHMANN, Anke ; STAADT, Oliver G.: Distance-adapted 2D Manipulation Techniques for Large High-Resolution Display Environments. In: *Proceedings of the International Conference on Computer Graphics Theory and Applications and International Conference on Information Visualization Theory and Applications*, 2012 (GRAPP/IVAPP), S. 387–394

[LSST11] LEHMANN, Anke ; SCHUMANN, Heidrun ; STAADT, Oliver ; TOMINSKI, Christian: Physical Navigation to Support Graph Exploration on a Large High-resolution Display. In: *Proceedings of the 7th International Conference on Advances in Visual Computing - Volume Part I*. Springer-Verlag, 2011 (ISVC'11), S. 496–507

[Mac86] MACKINLAY, Jock: Automating the Design of Graphical Presentations of Relational Information. In: *ACM Trans. Graph.* 5 (1986), April, Nr. 2, S. 110–141

[MB81] MÜLLER-BROCKMANN, Josef: *Grid systems in graphic design: a visual communication manual for graphic designers, typographers and three dimensional designers*. Verlag Arthur Niggli, 1981

[MBN+02] MYERS, Brad A. ; BHATNAGAR, Rishi ; NICHOLS, Jeffrey ; PECK, Choon H. ; KONG, Dave ; MILLER, Robert ; LONG, A. C.: Interacting at a distance: measuring the performance of laser pointers and other devices. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2002 (CHI '02), S. 33–40

[Mel10] MELO, Guido de: *Modellbasierte Entwicklung von Interaktionsanwendungen.*, University of Ulm, Diss., 2010

[MLM+12] MIROLL, Jochen ; LÖFFLER, Alexander ; METZGER, Julian ; SLUSALLEK, Philipp ; HERFET, Thorsten: Reverse Genlock for Synchronous Tiled Display Walls with Smart Internet Displays. In: *Proceedings of the 2nd IEEE International Conference on Consumer Electronics*, 2012 (ICCE-Berlin)

[NHP+09] NOCKE, Thomas ; HEYDER, Ursula ; PETRI, Stefan ; VOHLAND, Katrin ; WROBEL, Markus ; LUCHT, Wolfgang: Visualization of Biosphere Changes in the Context of Climate Change. In: *Wohlgemuth, V. (ed.). Information Technology and Climate Change, 2nd International Conference IT for empowerment* (2009), S. 29–36

[Nin13] NINTENDO: *Nintendo Unternehmensgeschichte*. http://www.nintendo.de/Unternehmen/Unternehmensgeschichte/Nintendo-Geschichte-625945.html, 2013. – (Zugriff 02.08.2013)

[Nor99] NORMAN, Donald A.: *The invisible computer: why good products can fail, the personal computer is so complex, and information appliances are the solution*. The MIT press, 1999

[NSC+06] NACENTA, Miguel A. ; SALLAM, Samer ; CHAMPOUX, Bernard ; SUBRAMANIAN, Sriram ; GUTWIN, Carl: Perspective cursor: perspective-based interaction for multi-display environments. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2006 (CHI '06), S. 289–298

[NSY+07] NACENTA, Miguel A. ; SAKURAI, Satoshi ; YAMAGUCHI, Tokuo ; MIKI, Yohei ; ITOH, Yuichi ; KITAMURA, Yoshifumi ; SUBRAMANIAN, Sriram ; GUTWIN, Carl: E-conic: A Perspective-aware Interface for Multi-display Environments. In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, ACM, 2007 (UIST '07), S. 279–288

[Ora13a] ORACLE: *Java*. https://www.java.com/de/, 2013. – (Zugriff 27.08.2013)

[Ora13b] ORACLE: *Java Web Start*. http://www.oracle.com/technetwork/java/javase/javawebstart/index.html, 2013. – (Zugriff 27.08.2013)

[OS02] OH, Ji-Young ; STUERZLINGER, Wolfgang: Laser pointers as collaborative

pointing devices. In: *Graphics Interface* Bd. 2002 Citeseer, 2002, S. 141–149

[OSJ+12] OEHLBERG, L. ; SIMM, K. ; JONES, J. ; AGOGINO, A. ; HARTMANN, B.: Showing is sharing: building shared understanding in human-centered design teams with Dazzle. In: *Proceedings of the Designing Interactive Systems Conference* ACM, 2012, S. 669–678

[Pec01] PECK, Choon H.: Useful parameters for the design of laser pointer interaction techniques. In: *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, ACM, 2001 (CHI EA '01), S. 461–462

[PLF+01] PONNEKANTI, Shankar ; LEE, Brian ; FOX, Armando ; HANRAHAN, Pat ; WINOGRAD, Terry: ICrafter: A service framework for ubiquitous computing environments. In: *Ubicomp 2001: Ubiquitous Computing* Springer, 2001, S. 56–75

[PRSB10] PRITZKAU, Albert ; RADLOFF, Axel ; SCHUMANN, Heidrun ; BARTZ, Dirk: Scattering and Jittering: Using Real and Illusionary Motion for Better Visual Scatterplot Analysis. In: *IEEE Information Visualization Poster*, 2010 (InfoVis'10)

[PWS09] PIRCHHEIM, Christian ; WALDNER, Manuela ; SCHMALSTIEG, Dieter: Deskotheque: Improved spatial awareness in multi-display environments. In: *IEEE Virtual Reality Conference 2009* IEEE, 2009 (VR 2009), S. 123–126

[RC00] ROMAN, Manuel ; CAMPBELL, Roy H.: Gaia: enabling active spaces. In: *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, ACM, 2000 (EW 9), S. 229–234

[Rea13] REALVNC: *remote access software for desktop and mobile platforms*. http://www.realvnc.com/, 2013. – (Zugriff 30.04.2013)

[RFS12] RADLOFF, Axel ; FUCHS, Georg. ; SCHUMANN, Heidrun: Supporting Visual Analysis in Smart Meeting Rooms. In: *Proceedings of the EuroVis Workshop on Visual Analytics*, 2012 (EuroVA)

[RHC+02a] ROMÁN, Manuel ; HESS, Christopher K. ; CERQUEIRA, Renato ; RANGANATHAN, Anand ; CAMPBELL, Roy H. ; NAHRSTEDT, Klara: Gaia: A Middleware Infrastructure to Enable Active Spaces. In: *IEEE Pervasive Computing* (2002), Oct–Dec, S. 74–83

[RHC02b] ROMAN, Manuel ; HO, Herbert ; CAMPBELL, Roy H.: Application mobility in active spaces. In: *1st International Conference on Mobile and Ubiquitous Multimedia, Oulu, Finland*, 2002

[RLS11] RADLOFF, Axel ; LUBOSCHIK, Martin ; SCHUMANN, Heidrun: Smart Views in Smart Environments. In: *Smart Graphics* Bd. 6815, Springer, 2011 (Lecture Notes in Computer Science), S. 1–12

[RLSS11] RADLOFF, Axel ; LUBOSCHIK, Martin ; SIPS, Mike ; SCHUMANN, Heidrun: Supporting Display Scalability by Redundant Mapping. In: *Proceedings of the 7th International Conference on Advances in Visual Computing - Volume Part I*, Springer-Verlag, 2011 (ISVC'11), S. 472–483

[RLSS12] RADLOFF, Axel ; LEHMANN, Anke ; STAADT, Oliver ; SCHUMANN, Heidrun: Smart Interaction Management: An Interaction Approach for Smart Meeting Rooms. In: *Proceedings of the 8th International Conference on Intelligent Environments*. Guanajuato, Mexico : IEEE Computer Society, June 2012 (IE'12), S. 228–235

[RMSF10] RAMOS, Carlos ; MARREIROS, Goreti ; SANTOS, Ricardo ; FREITAS, Carlos-Filipe: Smart Offices and Intelligent Decision Rooms. (2010), S. 851–880

[RNS11] RADLOFF, A. ; NOCKE, T. ; SCHUMANN, H.: Supporting Climate Impact Research by a Smart View Management. In: *IEEE Information Visualization Poster*, 2011 (InfoVis'11)

[Ros06] ROSENBAUM, Rene: *Mobile Image Communication using JPEG200*, University of Rostock, Diss., 2006

[RRC+06] RAMOS, Gonzalo ; ROBERTSON, George ; CZERWINSKI, Mary ; TAN, Desney ; BAUDISCH, Patrick ; HINCKLEY, Ken ; AGRAWALA, Maneesh: Tumble! Splat! helping users access and manipulate occluded content in 2D

drawings. In: *Proceedings of the working conference on Advanced visual interfaces*, ACM, 2006 (AVI '06), S. 428–435

[RSFWH98] RICHARDSON, Tristan ; STAFFORD-FRASER, Quentin ; WOOD, Kenneth R. ; HOPPER, Andy: Virtual Network Computing. In: *IEEE Internet Computing* 2 (1998), Januar, Nr. 1, S. 33–38

[RTNS14] RADLOFF, Axel ; TOMINSKI, Christian ; NOCKE, Thomas ; SCHUMANN, Heidrun: Supporting Presentation and Discussion of Visualization Results in Smart Meeting Rooms. In: *In der Erstellung für: Journal of Informatics – Special Issue "Interactive Visualizations: Design, Technologies, and Applications"* (2014)

[SER03] SHEN, Chia ; EVERITT, Katherine ; RYALL, Kathleen: UbiTable: Impromptu face-to-face collaboration on horizontal interactive surfaces. In: *UbiComp 2003: Ubiquitous Computing* Bd. 2864 Springer, 2003 (Lecture Notes in Computer Science), S. 281–288

[SG86] SCHEIFLER, Robert W. ; GETTYS, Jim: The X Window System. In: *ACM Transactions on Graphics* 5 (1986), April, Nr. 2, S. 79–109

[Sha13] SHAPE, Future: *SensFloor Sensorboden*. http://www.future-shape.de/de/technologies/11/sensfloor, 2013. – (Zugriff 06.05.2013)

[SLC+00] SU, Simon ; LOFTIN, R. B. ; CHEN, David T. ; FANG, Yung chin ; LIN, Ching yao: Distributed Collaborative Virtual Environment: Paulingworld. In: *Proceedings of the 10th International Conference on Artificial Reality and Telexistence*, 2000

[Sma91] SMALL, David: Modeling Watercolor by Simulating Diffusion, Pigment, and Paper Fibers. In: *In SPIE Proceedings*, 1991, S. 140–146

[SNLH09] SIPS, Mike ; NEUBERT, Boris ; LEWIS, John P. ; HANRAHAN, Pat: Selecting good views of high-dimensional data using class consistency. In: *Proceedings of the 11th Eurographics / IEEE - VGTC conference on Visualization*, Eurographics Association, 2009 (EuroVis'09), S. 831–838

[Sof13] SOFTWARE kakadu: *JPEG2000 developer toolkit*.

http://www.kakadusoftware.com/, 2013. – (Zugriff 27.08.2013)

[ST06a] SLAY, Hannah ; THOMAS, Bruce: Interaction and visualisation across multiple displays in ubiquitous computing environments. In: *Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*. New York, NY, USA : ACM, 2006 (AFRIGRAPH '06), S. 75–84

[ST06b] SLAY, Hannah ; THOMAS, Bruce H.: Evaluation of a universal interaction and control device for use within multiple heterogeneous display ubiquitous environments. In: *Proceedings of the 7th Australasian User interface conference - Volume 50*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2006 (AUIC '06), S. 129–136

[SWMB09] SCHREIBER, Michael ; WILAMOWITZ-MOELLENDORFF, Margeritta ; BRUDER, Ralph: New Interaction Concepts by Using the Wii Remote. In: *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*, Springer-Verlag, 2009, S. 261–270

[SWS+11] STEINBERGER, Markus ; WALDNER, Manuela ; STREIT, Marc ; LEX, Alexander ; SCHMALSTIEG, Dieter: Context-Preserving Visual Links. In: *IEEE Transactions on Visualization and Computer Graphics* 17 (2011), Dezember, Nr. 12, S. 2249–2258

[SXX+03] SHI, Yuanchun ; XIE, Weikai ; XU, Guangyou ; SHI, Runting ; CHEN, Enyi ; MAO, Yanhua ; LIU, Fang: The smart classroom: merging technologies for seamless tele-education. In: *IEEE Pervasive Computing* 2 (2003), Nr. 2, S. 47–55

[Syn13] SYNERGY: *Synergy – Mouse and keyboard sharing software*. http://synergy-foss.org, 2013. – (Zugriff 31.07.2013)

[TC05] THOMAS, James J. ; COOK, Kristin A.: *Illuminating the path: The research and development agenda for visual analytics*. IEEE Computer Society Press, 2005

[Tea13] TEAMVIEWER: *Fernwartung und Online Meeting*. http://www.teamviewer.com, 2013. – (Zugriff 30.04.2013)

[Ten00] TENNENHOUSE, David: Proactive computing. In: *Communications of the ACM* 43 (2000), Nr. 5, S. 43–50

[TFS08] TOMINSKI, Christian ; FUCHS, Georg ; SCHUMANN, Heidrun: Task-driven color coding. In: *Proceedings of 12th International Conference on Information Visualisation* IEEE, 2008 (IV'08), S. 373–380

[TGSP06] TAN, Desney S. ; GERGLE, Darren ; SCUPELLI, Peter ; PAUSCH, Randy: Physically large displays improve performance on spatial tasks. In: *ACM Transactions on Computer-Human Interaction* 13 (2006), Nr. 1, S. 71–99

[Thi10] THIEDE, Conrad: *Visuelle Informationsdarstellung in Smart Environments*, University of Rostock, Diss., 2010

[Tig13] TIGHTVNC: *free remote control software*. http://www.tightvnc.com/, 2013. – (Zugriff 30.04.2013)

[TMC04] TAN, Desney S. ; MEYERS, Brian ; CZERWINSKI, Mary: WinCuts: Manipulating Arbitrary Window Regions for More Effective Use of Screen Space. In: *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, ACM, 2004 (CHI EA '04), S. 1525–1528

[TQD09] TERRENGHI, Lucia ; QUIGLEY, Aaron ; DIX, Alan: A taxonomy for and analysis of multi-person-display ecosystems. In: *Personal and Ubiquitous Computing* 13 (2009), Nr. 8, S. 583–598

[Ubi13a] UBISENSE: *Ubisense Series 7000*. http://www.ubisense.net/en/resources/factsheets/series-7000-ip-sensors.html, 2013. – (Zugriff 05.06.2013)

[Ubi13b] UBISENSE: Ubisense Series 7000 IP Sensors Precision ultra – wideband measurement devices for industrial environments – Manual / Ubisense. 2013. – Forschungsbericht

[UIM+03] URNESS, Timothy ; INTERRANTE, Victoria ; MARUSIC, Ivan ; LONGMIRE, Ellen ; GANAPATHISUBRAMANI, Bharathram: Effectively Visualizing

Multi-Valued Flow Data using Color and Texture. In: *Proceedings of the 14th IEEE Visualization*, IEEE Computer Society, 2003 (VIS '03), S. 16–

[Ult13] ULTRAVNC: *Remote Control Software*. http://www.uvnc.com/, 2013. – 30.04.2013

[VDGR99] VAN DANTZICH, Maarten ; GOROKHOVSKY, Vadim ; ROBERTSON, George: Application redirection: hosting Windows applications in 3D. In: *Proceedings of the 1999 workshop on new paradigms in information visualization and manipulation in conjunction with the eighth ACM internation conference on Information and knowledge management* ACM, 1999, S. 87–91

[Wal11] WALDNER, Manuela: *WIMP Interfaces for Emerging Display Environments*, Graz University of Technology, Diss., 2011

[Wei91] WEISER, Mark: The computer for the 21st century. In: *Scientific American* 265 (1991), Nr. 3, S. 94–104

[Wei12] WEIGANDT, Valerius: *Device Driven Layouts for Multiple Views*. Bachelor Thesis, University of Rostock, Germany, 2012

[WGM+08] WANG, Lujin ; GIESEN, Joachim ; MCDONNELL, Kevin T. ; ZOLLIKER, Peter ; MUELLER, Klaus: Color Design for Illustrative Visualization. In: *IEEE Transactions on Visualization and Computer Graphics* 14 (2008), November, Nr. 6, S. 1739–1754

[Wii13] WIIUSEJ: *Java API for Wiimotes*. https://code.google.com/p/wiiusej/, 2013. – (Zugriff 28.10.2013)

[WJF+09] WIGDOR, Daniel ; JIANG, Hao ; FORLINES, Clifton ; BORKIN, Michelle ; SHEN, Chia: WeSpace: The Design Development and Deployment of a Walk-up and Share Multi-surface Visual Collaboration System. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2009 (CHI '09), S. 1237–1246

[WKS10] WALDNER, Manuela ; KRUIJFF, Ernst ; SCHMALSTIEG, Dieter: Bridging gaps with pointer warping in multi-display environments. In: *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending*

*Boundaries*, ACM, 2010 (NordiCHI '10), S. 813–816

[WL07] WALLACE, Grant ; LI, Kai: Virtually Shared Displays and User Input Devices. In: *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, USENIX Association, 2007 (ATC'07), S. 31:1–31:6

[WLSS09] WALDNER, M. ; LEX, A. ; STREIT, Marc ; SCHMALSTIEG, Dieter: Design considerations for collaborative information workspaces in multi-display environments. In: *Proceedings of Workshop on Collaborative Visualization on Interactive Surfaces, in conjunction with VisWeek* (2009), S. 36

[WP03] WILSON, Andrew ; PHAM, Hubert: Pointing in Intelligent Environments with the WorldCursor. In: *INTERACT*, 2003

[WPKS10] WALDNER, Manuela ; PIRCHHEIM, Christian ; KRUIJFF, Ernst ; SCHMALSTIEG, Dieter: Automatic configuration of spatially consistent mouse pointer navigation in multi-display environments. In: *Proceedings of the 15th international conference on Intelligent user interfaces*, ACM, 2010 (IUI '10), S. 397–400

[WPL+10] WALDNER, Manuela ; PUFF, Werner ; LEX, Alexander ; STREIT, Marc ; SCHMALSTIEG, Dieter: Visual links across applications. In: *Proceedings of Graphics Interface 2010*, Canadian Information Processing Society, 2010 (GI '10), S. 129–136

[WPS08] WALDNER, Manuela ; PIRCHHEIM, Christian ; SCHMALSTIEG, Dieter: Multi projector displays using a 3d compositing window manager. In: *Proceedings of the 2008 workshop on Immersive projection technologies/Emerging display technologiges* ACM, 2008, S. 14

[WPT03] WANT, Roy ; PERING, Trevor ; TENNENHOUSE, David L.: Comparing autonomic and proactive computing. In: *IBM Systems Journal* 42 (2003), Nr. 1, S. 129–135

[WS03] WILSON, Andrew ; SHAFER, Steven: XWand: UI for intelligent spaces. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing*

*Systems*, ACM, 2003 (CHI '03), S. 545–552

[WS10] WALDNER, Manuela ; SCHMALSTIEG, Dieter: Experiences with mouse control in multi-display environments. In: *Workshop on coupled display visual interfaces 2010 (PPD'10), in conjunction with AVI conference*, ACM, 2010 (AVI '10), S. 6–10

[WSGS11] WALDNER, Manuela ; STEINBERGER, Markus ; GRASSET, Raphael ; SCHMALSTIEG, Dieter: Importance-driven compositing window management. In: *Proceedings of the 2011 annual conference on Human factors in computing systems* ACM, 2011, S. 959–968

[WW94] WEITZMAN, Louis ; WITTENBURG, Kent: Automatic Presentation of Multimedia Documents Using Relational Grammars. In: *Proceedings of the Second ACM International Conference on Multimedia*, ACM, 1994 (MULTIMEDIA '94), S. 443–451

[YHHC05] YOUNGBLOOD, G. M. ; HEIERMAN, Edwin O. ; HOLDER, Lawrence B. ; COOK, Diane J.: Automation Intelligence for the Smart Environment. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers Inc., 2005 (IJCAI'05), S. 1513–1514

[YKSJ07] YI, Ji S. ; KANG, Youn a. ; STASKO, John ; JACKO, Julie: Toward a Deeper Understanding of the Role of Interaction in Information Visualization. In: *IEEE Transactions on Visualization and Computer Graphics* 13 (2007), November, Nr. 6, S. 1224–1231

[ZLL+04] ZHANG, Hangjin ; LIU, Qiong ; LERTSITHICHAI, Surapong ; LIAO, Chunyuan ; KIMBER, Don: A presentation authoring tool for media devices distributed environments. In: *Proceedings of 2004 IEEE International Conference on Multimedia and Expo* Bd. 3 IEEE, 2004 (ICME'04), S. 1755–1758

# Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die eingereichte Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Diese Arbeit hat keiner anderen Fakultät oder Universität zur Prüfung vorgelegen.

Rostock, 21. Juli 2014

.....

Axel Radloff

# Lebenslauf

## Persönliche Daten

| | |
|---|---|
| Name: | Axel Radloff |
| Wohnort: | Rostock |
| geboren am: | 01.01.1983 |
| geboren in: | Rostock |
| Nationalität: | deutsch |

## Werdegang

| | |
|---|---|
| seit 11/2013 | ECS Engineering Consulting & Solutions GmbH |
| 10/2009 - 10/2013 | Promotion an der Universität Rostock:<br>Thema: Smart Views in Smart Environments<br>im GrK MuSAMA |
| 09/2008 - 10/2009 | ECS Engineering Consulting & Solutions GmbH |
| 10/2003 - 09/2008 | Diplomstudium Informatik, Universität Rostock |
| 2002 | Abitur, Erasmus Gymnasium, Rostock |

# Wissenschaftliche Veröffentlichungen

## Journal Veröffentlichung

- RADLOFF, Axel ; TOMINSKI, Christian ; NOCKE, Thomas ; SCHUMANN, Heidrun: Supporting Presentation and Discussion of Visualization Results in Smart Meeting Rooms. Will appear in: Journal of Visual Computer, 2014

## Begutachtete Konferenzbeiträge

- LUBOSCHIK, Martin ; RADLOFF, Axel ; SCHUMANN, Heidrun: A New Weaving Technique for Handling Overlapping Regions. In: *Proceedings of the International Conference on Advanced Visual Interfaces*, ACM, 2010 (AVI'10), S. 25–32, doi: 10.1145/1842993.1842999
- RADLOFF, Axel ; LUBOSCHIK, Martin ; SCHUMANN, Heidrun: Smart Views in Smart Environments. In: Smart Graphics Bd. 6815, Springer, 2011 (Lecture Notes in Computer Science), S. 1–12, doi: 10.1007/978–3–642–22571–0_1
- RADLOFF, Axel ; LUBOSCHIK, Martin ; SIPS, Mike ; SCHUMANN, Heidrun: Supporting Display Scalability by Redundant Mapping. In: Proceedings of the 7th International Conference on Advances in Visual Computing – Volume Part I, Springer-Verlag, 2011 (ISVC'11), S.472–483, doi: 10.1007/978–3–642–24028–7_44
- RADLOFF, Axel ; LEHMANN, Anke ; STAADT, Oliver ; SCHUMANN, Heidrun: Smart Interaction Management: An Interaction Approach for Smart Meeting Rooms. In: Proceedings of the 8th International Conference on Intelligent

Environments. Guanajuato, Mexico : IEEE Computer Society, June 2012 (IE'12), S. 228–235, doi: 10.1109/IE.2012.34

- RADLOFF, Axel ; FUCHS, Georg. ; SCHUMANN, Heidrun: Supporting Visual Analysis in Smart Meeting Rooms. In: Proceedings of the EuroVis Workshop on Visual Analytics, 2012 (EuroVA), doi: 10.2312/PE/EuroVAST/EuroVA12/001–005

## Begutachtete sonstige Beiträge

- LUBOSCHIK, Martin ; RADLOFF, Axel ; SCHUMANN, Heidrun: Using NPR-Rendering Techniques for the Visualization of Uncertainty. In: *IEEE Information Visualization Poster*, 2010 (InfoVis'10)
- PRITZKAU, Albert ; RADLOFF, Axel ; SCHUMANN, Heidrun ; BARTZ, Dirk: Scattering and Jittering: Using Real and Illusionary Motion for Better Visual Scatterplot Analysis. In: *IEEE Information Visualization Poster*, 2010 (InfoVis'10)
- RADLOFF, Axel ; NOCKE, Thomas ; SCHUMANN, Heidrun: Supporting Climate Impact Research by a Smart View Management. In: IEEE Information Visualization Poster, 2011 (InfoVis'11)

## Fachvorträge

- Smart Views in Smart Environments. Smart Graphics (GS'11), Bremen, Germany, 18.07.2011
- Supporting Display Scalability by Redundant Mapping. Int. Conference on Advances in Visual Computing (ISVC'11), Las Vegas, USA, 27.09.2011

# Thesen

1. Smart Meeting Rooms sind Multi-Display-Umgebungen, die auf die Zusammenarbeit heterogener Geräte unter Einbeziehung der persönlichen Geräte der Nutzer fokussieren. Ziel solcher Umgebungen ist es, Nutzer bei ihrer Arbeit zu unterstützen, indem Informationen über Nutzer und Umgebung gesammelt und ausgewertet werden.

2. Die Eigenschaften eines Smart Meeting Rooms stellen die Informationspräsentation vor besondere Herausforderungen. Neben den heterogenen Geräten, den heterogenen Anwendungen, heterogenen Displayflächen und den Eigenschaften der Nutzer müssen die Charakteristika der Informationsrepräsentationen bei der Anzeige berücksichtigt werden. Weiterhin ist die Interaktion in solchen Umgebungen von großer Wichtigkeit, um dynamisch auf sich ändernde Schwerpunkte bei Präsentationen oder Diskussionen reagieren zu können.

3. Das direkte Verbinden von persönlichen Geräten und Displayflächen ist ungeeignet für die Anzeige von Informationen in einem Smart Meeting Room. Auch Ansätze in der Literatur, die eine Informationsanzeige über multiple Displayflächen bereitstellen, adressieren in der Regel weder die automatische Assistenz der Nutzer noch die Dynamik und Flexibilität solcher Umgebungen.

4. Das *Smart View Management* erlaubt es, visuelle Ausgaben von verschiedensten Applikationen (*Views*) auf verschiedenen Geräten zu erzeugen, diese *Views* zu gruppieren, automatisch verschiedenen Displayflächen zuzuweisen und auf diesen Displayflächen automatisch anzuordnen. Damit ist die feste Zuordnung eines Gerätes und seiner visuellen Ausgaben zu einer Displayfläche nicht mehr erforderlich. Dies ermöglicht es, *Views* frei im Raum zu kombinieren und anzuzeigen.

5. Die Zuordnung von *Views* zu Displayflächen wird durch das *Smart Display Mapping* vorgenommen. Dafür werden Position und Blickrichtung der Nutzer, sowie Eigenschaften und Zusammengehörigkeit der *Views* automatisch ausgewertet.

6. Für die Anordnung der *Views* auf den Displayflächen wurden zwei Layout Mechanismen entwickelt. Auf Basis der Eigenschaften der *Views* und der spezifischen Displayfläche werden entweder mit einem Gitterbasierten oder mit einem Federkraftbasierten Layout Mechanismus *Views* automatisch positioniert und skaliert.

7. Das *Smart Interaction Management* ermöglicht es jedem Nutzer, durch die Entkopplung von syntaktischer und semantischer Interaktion mit verschiedenen Interaktionsgeräten mit allen *Views* auf allen Displayflächen zu interagieren.

8. Auf Basis des *Smart Interaction Managements* wurde ein einheitliches Interface zur interaktiven Manipulation der Anzeige von *Views* geschaffen. Damit wird es unabhängig von der *View* generierenden Applikation ermöglicht, *Views* interaktiv Displayflächen zuzuweisen, Views bzgl. der verwendeten Layout Strategie zu positionieren und zu skalieren, sowie Regionen von Interesse auf Basis der *JPEG2000* Kodierung spezifisch anzuzeigen.

9. Das *Smart Interaction Management* ermöglicht es, den Inhalt von *Views* zu manipulieren und damit in den Prozess der *View* Generierung einzugreifen. Dadurch kann eine Fokussierung auf spezifische Aspekte der Daten und eine Unterstützung der Wahrnehmbarkeit der dargestellten Informationen erreicht werden.

10. Das *Smart View Management* und das *Smart Interaction Management* stellen neue Konzepte für die Informationspräsentation in Smart Meeting Rooms dar. Auf Basis einer Bildbasierten Informationspräsentation wird eine automatische Konfiguration der Anzeige unter Berücksichtigung der Eigenschaften und Ziele von Smart Meeting Rooms ermöglicht. Auf Basis der Interaktion wird es außerdem ermöglicht, die Anzeige vollständig interaktiv zu manipulieren.