HeatSpace: Automatic Placement of Displays by Empirical Analysis of User Behavior

Andreas Fender[1], **David Lindlbauer**[2], **Philipp Herholz**[2], **Marc Alexa**[2], **Jörg Müller**[1]
[1]Department of Computer Science, Aarhus University, Denmark
[2]TU Berlin, Germany

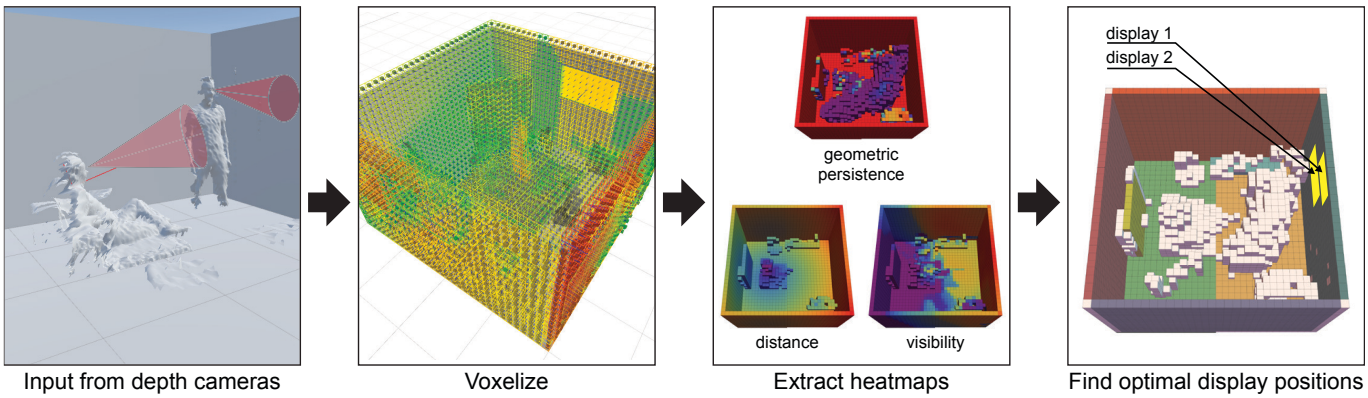

Figure 1. HeatSpace captures the environment and users including their viewing direction with multiple depth cameras for empirical analysis of user behavior. Properties such as geometric persistence, distance and visibility are extracted. HeatSpace takes the number of displays and their resolution as input and outputs their optimal position and size. If users additionally specify the size of displays, HeatSpace will only optimize their position. Furthermore, users can constrain the position of displays to a specific surface. HeatSpace will then find optimal display positions on that surface.

Abstract

We present HeatSpace, a system that records and empirically analyzes user behavior in a space and automatically suggests positions and sizes for new displays. The system uses depth cameras to capture 3D geometry and users' perspectives over time. To derive possible display placements, it calculates volumetric heatmaps describing geometric persistence and planarity of structures inside the space. It evaluates visibility of display poses by calculating a volumetric heatmap describing occlusions, position within users' field of view, and viewing angle. Optimal display size is calculated through a heatmap of average viewing distance. Based on the heatmaps and user constraints we sample the space of valid display placements and jointly optimize their positions. This can be useful when installing displays in multi-display environments such as meeting rooms, offices, and train stations.

Introduction

Multi-display environments (MDEs) are used in various scenarios, ranging from individual multi-display work, via collaborative work, to command-and-control rooms. Indeed, an increasing number of contemporary workplaces can already be considered as multi-display environments. A practical concern can be unfavorable display placement, with displays being installed in areas where nobody normally looks, or being occluded by furniture or people.

Spaces which will be equipped with multi-display systems may be shared among groups with different usages. This leads to the optimal display placement being non-obvious since no single person might have a complete overview of how the space is actually used in practice. We approach this problem inspired by three scenarios:

Scenario 1 — Open-plan office: An open-plan office space for 30 workers will be equipped with 5 newly purchased full-HD information displays. The displays will be mounted on walls and their positions should be chosen so they are visible from as many desks as possible. Furthermore, the displays should be located in a way to avoid occlusion from passers-by.

Scenario 2 — Meeting room: A meeting room with rearrangeable desks and chairs will be equipped with 3 large-scale interactive whiteboards. While the resolution of the whiteboards is fixed through the projectors, their size is variable. The room is used by different groups, for example for presentations on the existing 50" display or for brainstorming sessions on flip charts.

Scenario 3 — Train station: The lobby of a train station will be equipped with 10 information displays. Pas-

UIST 2017, October 22–25, 2017, Quebec City, QC, Canada

 ISBN 978-1-4503-4981-9/17/10…$15.00
DOI: https://doi.org/10.1145/3126594.3126621

senger behavior changes throughout the day and week. All displays should have maximum visibility and should not be occluded by passers-by.

In all these scenarios, various factors have to be taken into account, including users' position, field of view, viewing angle between users and a potential display surface, and occlusions. These factors also change over time, which makes finding optimal display placements challenging. The problem is exacerbated when the person installing the display (the *installer*) has little to no knowledge about the actual usage of the space.

We present HeatSpace, a system that analyzes user behavior over time and proposes display placements based on this empirical data. The installer equips a space with one or more commodity depth cameras (e. g. Microsoft Kinect V2) so that their combined view covers a good portion of the space and calibrates them using a procedure similar to RoomAlive [10]. HeatSpace continuously records user activity by incorporating automatic geometry reconstruction and skeleton tracking.

The retrieved depth data is used to calculate geometric persistence (static versus moving parts of the space), distance between the users and geometry in space, and surface planarity. HeatSpace continuously evaluates users' viewing behavior to calculate the visibility of all areas in the space based on occlusions, users' field of view and viewing angle. All this information is discretized and stored in multiple 3D voxel grids. These voxel grids are *integrated over time* and analyzed to generate heatmaps of display surface quality. Our system takes potential disturbances into account, for example occlusion through physical objects or users.

HeatSpace takes the number of displays and their resolution as input, specified by the installer. It then automatically calculates and outputs the positions and sizes of display surfaces based on aforementioned heatmaps. As additional input, the installer can specify the size of displays, leading to HeatSpace only optimizing the position of the display surfaces. Furthermore, the installer can set initial positions (e. g. on a specific wall). HeatSpace then only optimizes the positions locally.

After installing the displays, the depth cameras can be removed. Alternatively, they can remain in the space to continue validating the display placement for correction at a later stage or for interaction purposes.

HeatSpace aims at alleviating the problem of display placement by eliminating guesswork and trial-and-error, and replaces them with empirical analysis of actual user behavior. In general, HeatSpace can also be used for positioning objects other than digital displays. By analyzing user flow and viewing behavior, it could for example be used to place exit signs in train stations so that they are not occluded by passers-by.

Contributions

- A method for automatic generation of volumetric heatmaps that enables the empirical analysis of user behavior based on various properties including geometric persistence of the space, distance and visibility.
- A method for automatically optimizing the position and size of display surfaces based on users' behavior, suitable for a wide range of single-user and multi-user scenarios, where optimal display positions are non-obvious.

RELATED WORK

Multi-display environments

Multi-display systems are used for a variety of scenarios, from single-user to multi-user collaboration. These systems often consist of a set of private and public displays with different sizes and arrangements, for example in the Stanford iRoom [9] or i-Land [24]. MDEs allow exploiting the benefits of large displays, for example increased productivity [21] and better visibility, and combine them with private displays for example for collaborative sensemaking (e. g. [27]). MDEs were studied for a wide range of properties, including visual attention [20], display arrangement [12], distraction [4] and cross-device interaction [17].

All these works share the common assumption that the displays in a particular space *are already installed*. The systems were most often set up by experts with extensive knowledge of which groups work in a specific room as well as their needs and habits. We believe, however, that there is a wide range of scenarios where this knowledge is not available.

The problem of automatically positioning displays in a multi-display environment has not yet been addressed. Prior work such as work by Bell et al. [1] addressed the problem of content placement in a given space, in their case in an augmented reality scenario. The data gathered through HeatSpace could not only be used for positioning displays but also for automatic content placement, since properties such as visibility or field of view are available.

Besides multi-display environments, other work focused on fully covering spaces with display surfaces, for example through projection. Jones et al. [10] developed RoomAlive, a living room equipped with multiple projector-camera units to create an immersive environment for displaying content. Their system would also allow displaying arbitrary contents on any surface in a space. CAVE systems have similar capabilities [5]. Traditionally, however, conventional spaces (e. g. offices, meeting rooms, public spaces) are equipped only with a small number of displays, mainly due to practical reasons such as cost. Pinhanez [18] proposed the everywhere displays projector, which is essentially a steerable projector unit. For HeatSpace, such a device would allow for automatically positioning the display surfaces based on the gathered data without mounting individual displays.

Space Syntax and Isovist

Our work is related to work in architecture around Space Syntax and Isovist [2]. Space Syntax aims to computationally analyze the structure of cities and buildings based on graphs describing connectivity, visibility relationships etc. In particular, the concept of Isovist describes all points in a space that are visible from a certain vantage point. The main difference to our work is that, to our knowledge, most work in Space Syntax is based on the built structure, without recordings of actual user activity. While most work in Space Syntax describes 2D structure using graphs, some newer work takes 3D structure into account [26].

Automatic camera and projection placement

The problem of automatically placing displays in an environment is related to the camera placement problem. The goal of work such as by Ghanem et al. [7] is to automatically place cameras in a room to obtain maximum visual coverage. This is related to the Art Gallery Problem [15], which was shown to be NP-hard. There are a variety of possible solutions to this problem, we refer readers to a literature survey by Mavrinac and Chen [13]. HeatSpace solves a related problem, i. e. given a visual coverage for a room, find the optimal placement for a set of display surfaces.

Bimber et al. [3] worked on various correction techniques for improving projection, taking parameters such as color or surface geometry into account. Lischke et al. [12] used similar methods for automatically determining whether a surface is suitable for projection. In our work, we analyze user behavior to find out where to place displays. Their insights could serve as additional parameters in our optimization procedure.

Analyzing viewing behavior

Users' viewing behavior is analyzed for a wide range of applications, including measuring visual attention of online content (e. g. [14]) or public displays (e. g. [25, 28]). Those measurements are not limited to 2D surfaces but have also been performed on real-world 3D objects (e. g. [16]) and immersive virtual environments (e. g. [23]).

In multi-display environments, gaze analysis is typically performed to find out where users are looking, for example through passive sensing [22] or head-mounted eye trackers [11]. The data is then used e. g. to minimize distraction by displaying contents only when users are looking at a specific display (e. g. Dostal et al. [6]).

For analyzing and visualizing the data gathered by HeatSpace we use heatmaps (e. g. [19, 29]) in a volumetric 3D representation. 3D representations have been used previously for saliency maps for example by Smith et al. [22] for psychophysical experiments, however mostly for calculating data points directly on the geometry. In contrast, we track viewing behavior not only on existing geometry but also include information on occlusion and geometric persistence in our volumetric data representation.

HEATSPACE OVERVIEW

HeatSpace consists of two main components: a heatmap generator and a placement optimizer. The heatmap generator takes data from multiple commodity depth cameras (Microsoft Kinect V2) as input, three in the examples shown in the paper. The cameras are mounted and calibrated to cover as much space as possible. This setting allows us to gather a complete 3D representation of the space as well as to track users including their head position and orientation.

HeatSpace then continuously monitors the space and tracks the geometry and users, for example for several days or weeks to capture as many situations in a space as possible. It extracts 3D voxel grids (the heatmaps) based on multiple parameters such as geometric persistence, distance between users and the geometry, and visibility based on occlusion, field of view and viewing angle. The generation of these heatmaps is detailed below.

The placement optimizer samples the heatmaps for potential display surface positions and ranks the positions according to the heatmaps. HeatSpace suggests display surface positions and display sizes in the space based on the number of the displays that have been specified by the installer. If the installer chooses to also specify the size of the displays, HeatSpace only searches for optimal positions of the display while leaving their size fixed. Once the displays have been placed, the depth cameras can be either removed or can remain to monitor the space for changes in behavior or structure, or interaction purposes.

DATA STRUCTURES AND PROCESSING

This section briefly introduces our data structures and definitions used throughout the rest of the paper.

Capturing geometry and users

Multiple depth cameras are used to continuously capture the environment from various viewing angles. For every frame we use the depth streams and the extrinsics and intrinsics of the cameras to reconstruct a mesh including vertices, faces and normals, denoted as $M_i = \{F_i, V_i, N_i\}$, shown in Figure 2. This also incorporates geometry of users and other moving objects. In addition, we extract a set of user skeletons using the Kinect

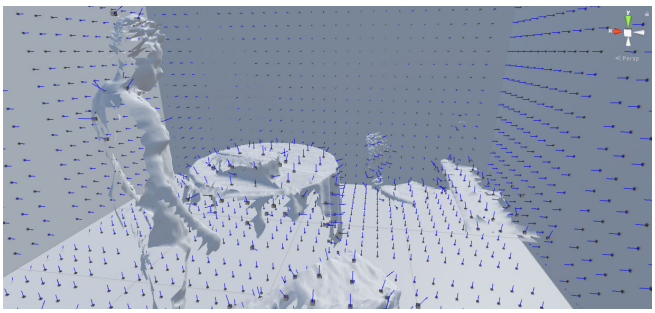

Figure 2. The input from the depth cameras is combined at every frame into a single mesh M_i.

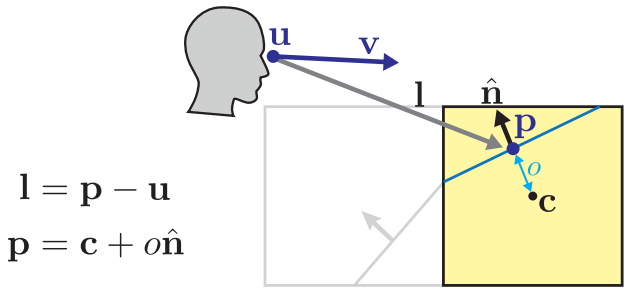

Figure 3. Data calculated and stored per frame for one voxel (yellow). The figure shows a plane that approximates the surface which cuts through the voxel (depicted here as oblique line inside the voxel), a user situated relative to the voxel, and a neighboring voxel (gray).

SDK for tracking head position and orientation, combined from all depth cameras. If a user is tracked by multiple depth cameras, we merge the data to a single representation in world coordinates.

Figure 3 provides an overview of all the information apart from the heatmaps we calculate and store per frame. We store the user's head position $\mathbf{u}$ and the view direction $\mathbf{v}$ pointing forward according to the head orientation. We approximate the geometry that is included in each voxel with a plane, for example to simplify calculating if adjacent voxels lie on the same surface. For each voxel, we define the following variables related to the geometry within a voxel: the voxel center $\mathbf{c}$ in world coordinates; the unit normal vector of the plane $\hat{\mathbf{n}}$; the plane offset $o \in \mathbb{R}$ indicating the distance from the voxel center to the plane; the point $\mathbf{p}$ on the plane, which is shifted by the plane offset from the voxel center along the normal. Lastly, we define the line of sight $\mathbf{l}$ between $\mathbf{u}$ and $\mathbf{p}$. We use 3D voxel grids to store the calculated data.

Heatmap data

We define a heatmap h as a uniform 3D voxel grid $h \in [0..1]^{m \times m \times m}$. The number of voxels m per dimension governs the resolution of the heatmap. Each heatmap is updated on a per-frame basis averaged over time t as

$$h = \sum_{i=0}^{t} \frac{1}{t} h_i. \qquad (1)$$

h_i denotes the heatmap values at time i, and h the accumulated value.

HEATMAP GENERATION

HeatSpace generates and continuously updates three different heatmaps, namely *geometric persistence*, *distance* and *visibility*. These are summarized in Figure 5. The geometric persistence heatmap allows distinguishing between static surfaces (e. g. walls, tables) and dynamic surfaces (e. g. passers-by, movable chairs). The distance heatmap accumulates the distance between users and the geometry at the current frame M_i for each voxel. If multiple users are present, this heatmap stores average distance values. A visibility heatmap serves as an accumulated quality measure of how well each voxel can be seen by users. We combine several criteria, specifically the occlusion of the voxel from the point of view, where the voxel is located in the user's field of view, and the viewing angle with respect to the surface normal. For multiple simultaneous users, we only store the minimum visibility value for each voxel to make sure to later find display positions that are well visible for all users.

Geometric persistence

The geometric persistence heatmap h_g contains the probability that the voxel contains geometry, shown in Figure 4. This heatmap is used in the optimization to find potential candidates for display surfaces. For each frame i, we calculate the heatmap $h_{g,i}$ by intersecting the voxel grid with M_i. h_g takes all geometry into account, including users that are in the space.

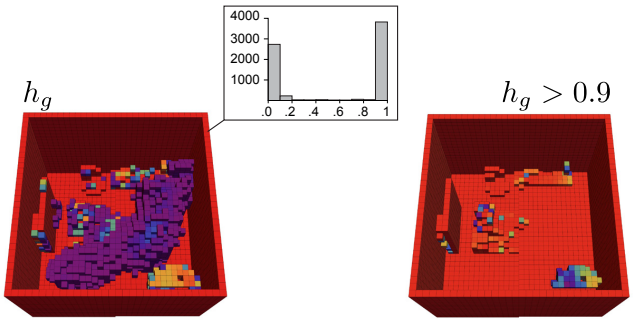

Figure 4. Heatmap for geometric persistence. Red indicates high persistence. *Left* shows all voxels that include geometry and the corresponding persistence values, *right* shows voxels with persistence > 0.9.

Distance

The distance heatmap h_d encodes the distance between every voxel and the head positions of all users. Figure 5 (c) shows an example of a distance heatmap. It is used by the placement optimizer to calculate the perceived resolution of a potential display surface when inferring the optimal size of a display as discussed below. Furthermore, it enables favoring close display surfaces over distant ones, while ensuring that a display is not too close to a user (e. g. avoiding that a 50" display is placed only 10 cm away from a user). This is enforced as part of the energy minimization procedure, discussed later.

Visibility

The visibility heatmap h_v encodes three different factors: *occlusion*, *field of view* and *viewing angle*. At each frame, all these factors are calculated individually per voxel and combined to h_v, as described below. Figure 5 (d) shows an illustration of the factors and the resulting visibility heatmap. After calculation, the factors are multiplied per voxel to account for the fact that a low score in any factor should result in an overall poor display surface quality. For example, a good viewing angle can not make up for an occluded voxel.

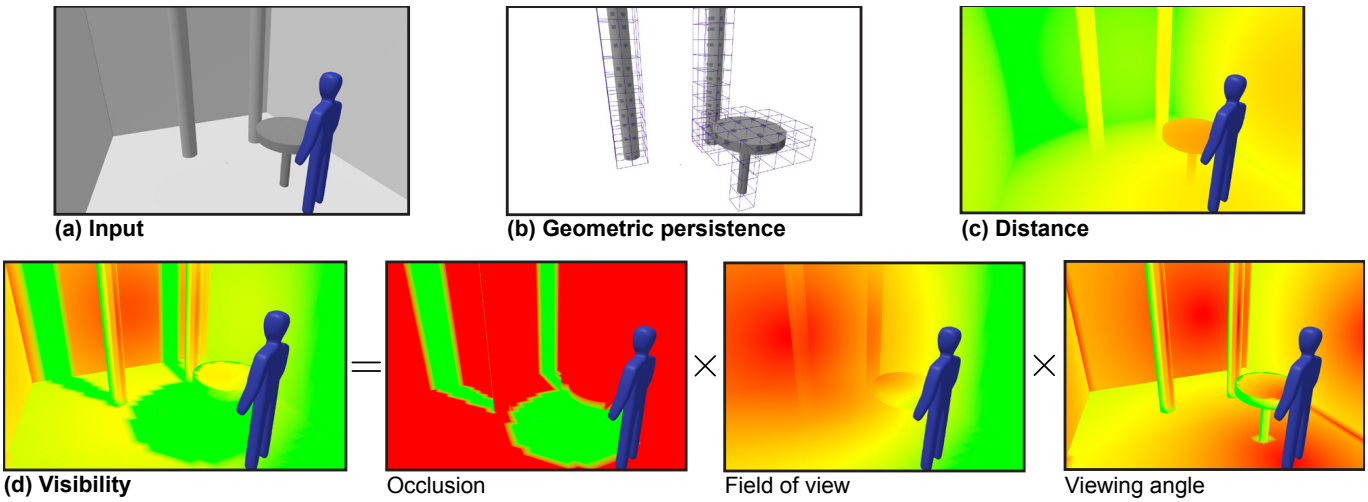

Figure 5. Simple example showing how voxel grids and heat maps are generated. Note that while our heatmaps are created as volumetric voxel grids, this figure illustrates their values on the surface geometry for better understanding. (a) A user (blue) looks at the wall while standing at a table in front of two pillars. (b) Voxels are generated based on geometric persistence (wall geometry is excluded in the image). (c) The distance heatmap stores the distance from each voxel to the user's head. (d) The visibility heatmap $h_{v,i}$ at time i is calculated from occlusion, field of view, and viewing angle. Occlusion only yields the value 0 (occluded, green) or 1 (visible, red). The value for field of view is high close to user's line of sight, and decreases towards user's periphery and is 0 behind the user. The value for viewing angle decreases with increasing angle between user and geometry.

Occlusion

For each voxel we calculate if it is visible from each user's point of view by casting a ray from $\mathbf{u}$ to $\mathbf{p}$. If a ray sent from $\mathbf{u}$ intersects M_i before reaching $\mathbf{p}$, the value in the voxel is set to 0, otherwise it is set to 1.

Field of view

HeatSpace calculates where each voxel is located with respect to the user's observable world. Voxels closer to the user's central line of sight are rated higher than voxels in the peripheral field of view. Voxels located behind the user are rated 0. This is quantified for each voxel as $\max(0, -(\mathbf{l} / \|\mathbf{l}\|)^\intercal \cdot \mathbf{v})^{e_f}$. This essentially encodes how easy or "natural" it is for users to look at a certain location, meaning if they look at or close to a location anyways, potentially even without the presence of a display. The exponent e_f is chosen to set the importance within the field of view. With a high exponent only surfaces right in front of the user are considered to have a high visibility, whereas with a small exponent also points in the periphery are considered as well visible. In our current implementation we set $e_f = 0.3$, as discussed later.

Viewing angle

HeatSpace takes the angle between the line of sight and the geometry normal at the intersection point with M_i into account. If the viewing angle is very steep, the display quality would be decreased, even though a surface might be visible for users. We calculate this for each voxel as $\max(0, -(\mathbf{l} / \|\mathbf{l}\|)^\intercal \cdot \hat{\mathbf{n}})^{e_a}$. The exponent e_a can be chosen similarly to e_f, in our current implementation $e_a = 0.35$.

DISPLAY PLACEMENT

In the following we detail our algorithm for automatically placing display surfaces in a space given the heatmaps for geometric persistence, distance and visibility, denoted h_g, h_d and h_v, respectively.

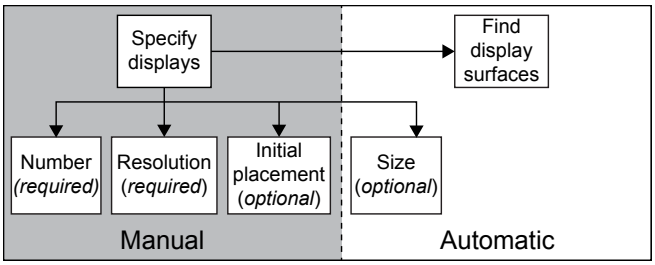

Figure 6. The installer specifies the number of displays and their resolution (i. e. number of pixels), all other measures can be computed automatically by our software.

Input and output

The output of the display placement algorithm depends on the input of the installer, illustrated in Figure 6. As a minimum input, the installer specifies (1) *the number of displays* that should be placed in a space and (2) *their resolution*. We refer to the number of horizontal and vertical pixels on a display as the resolution of a display. Conversely, we later optimize the size of a display surface based on the *perceived resolution* of a display. We use the term perceived resolution to describe the *maximum size of a pixel with which a user cannot see the individual pixels on a display*, governed by human visual acuity of 1 minute of arc. This gives us the size of a display given a resolution defined by the installer. With this information, HeatSpace will find the optimal display surface position *and* their size. Optionally, the installer can specify (3) *the size of the displays*, leading to HeatSpace only suggesting optimal display surface positions. As a last optional parameter, the installer can specify (4) *initial display placements*, for example on which wall the

displays should be located. HeatSpace will then position the display surfaces on the same surface as those initial placements (e. g. same wall). If size or position are specified, we use them as constraints in our optimization.

Optimization

We approach the problem of automatic placement based on the following constraints.

(C1) Display surfaces should be located on geometrically persistent regions.

(C2) Display surfaces should be located on planar regions.

(C3) Display surfaces should be located in visible regions.

(C4) Display sizes should yield specified screen sizes for average user distance.

Finding suitable display surfaces

The first step in our display placement procedure is to find planar and geometrically persistent candidate regions (C1 and C2) for display placement. We identify suitable areas by filtering the voxel graph G which connects each voxel to its up to six neighbors. We use a graph representation since this allows us to easily compare the geometric persistence and planarity of adjacent voxels. To account for (C1) we filter the graph based on geometric persistence, yielding a subgraph G' defined by the edges connecting persistent voxels i, j:

$$G' : \{(i,j) \in G,\, h_g(i) > \tau \,\&\, h_g(j) > \tau\}. \tag{2}$$

τ denotes the threshold for geometric persistence, typically in the range of 0.9 to account for noise from the depth cameras. In the next step, we filter the graph G' based on planarity, i.e. we remove edges between voxels if they do not lie on the same surface plane. This subgraph G'' can be defined as

$$G'' : \{(i,j) \in G' : \alpha(i,j) > \rho\}, \tag{3}$$

where $\alpha(i,j)$ describes the angle between the planes of voxels i and j. ρ is the threshold for disconnecting the nodes of G'. In our case $\rho = 10°$ to account for noise in the depth cameras. Candidate regions are subsequently identified as connected components $C = (c_0, c_1, \cdots, c_k)$ of G''. As a simple example, in an empty space each member of C would cover one mostly planar region, e.g. a wall, floor or ceiling if captured by the depth cameras. This also means that most members of C cover a larger area than necessary for positioning a display.

Finding positions and sizes for display surfaces

If the installer specifies the number of displays and their resolutions (e.g. 1920 × 1080 pixels), HeatSpace will automatically place displays on candidate regions in order to optimize with respect to (C3) and (C4). First, we sample potential display positions. For each voxel in a candidate region the voxel center is projected onto the corresponding plane and serves as display center. Using information from the distance heatmap h_d, the display width is computed using the human visual acuity

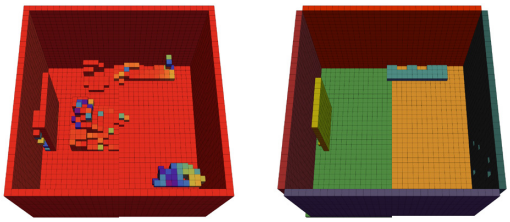

Figure 7. ***Left*** **shows G' with all voxels filtered for geometric persistence.** ***Right*** **shows all candidates in C, with each color representing one connected component (i. e. voxels on the same plane).**

and display resolution, i.e. a pixel should on average be about 1 arc minute in width. Therefore, the width of the display is computed as

$$width = \tan(1') \cdot distance \cdot resolution_x \tag{4}$$

The height is calculated from the aspect ratio of the display. This measure could be replaced also taking other parameters such as contrast and readability into account (cf. Ziefle [30] or Healey and Sawant [8]).

Given position and dimensions we can place the display on the surface keeping the x-axis of the display parallel to the floor. In order to check for (C1) and (C2) we have to ensure that all voxels intersected by the display belong to the same planar component c_i. If this is the case we sum the per-voxel energy

$$\mathcal{E} = h_v \cdot f_d(h_d) \tag{5}$$

over all intersected voxels and store this data along with the display geometry as potential display. The function f_d is designed to penalize small distances ($< 0.5\,\mathrm{m}$) and h_v favors visible areas. Note that we have to keep independent samples for each specified display resolution.

To find the position of n displays we first place them greedily in an arbitrary order by picking the highest scoring candidate from the samples which do not overlap with already placed displays. This strategy is of course not optimal. If two displays end up in the same connected component, moving both of them into a less optimal position could still result in better total energy. To deal with this problem we jointly optimize all display positions per connected component using an alternating gradient descent approach if two or more displays happen to reside on the same component.

Starting with a valid, overlap-free but random placement for each display we repeatedly move the displays to closeby samples in order to individually maximize the energy function. Eventually the process converges and a local optimum is found. We repeat the process for several starting conditions (five in all our examples) and pick the solution with highest energy.

Installer constraints

If the installer wishes to fix display sizes the sampling procedure will simply skip the optimal size computation

and use the pre-defined display geometry. If initial positions are specified, only samples for the corresponding candidate (i. e. the connected component the initial position is part of) are considered for that display, effectively constraining the position to a specific surface.

IMPLEMENTATION

This section briefly outlines the implementation of our software, which we are planning to release as open source. HeatSpace consists of three components: the heatmap generator, the display placement optimizer and the editor for installers.

Heatmap generator

The heatmap generator is implemented as a Unity 5 plugin in C#, running on a commodity gaming computer with Windows 10. This computer also acts as a server retrieving the data from all depth cameras. Each Kinect camera is connected to an individual computer that streams the data to this server, similar to RoomAlive [10]. The Unity application retrieves the depth streams and performs the geometry reconstruction and all other calculations discussed in Section *Data structures and processing*. For each frame, it calculates and stores the heatmaps for geometric persistence, distance and visibility.

Display placement

The placement optimizer is implemented as a Mathematica application, exchanging data with the Unity application. It retrieves the heatmaps and parameters specified by the installer and outputs the final display placements. We use Mathematica for filtering data, converting heatmaps into graphs and finding connected components. Furthermore, it holds the custom implementation of the alternating gradient descent optimization we are using for inferring the size and position of potential display surfaces.

Editor for installers

The editor for installers is implemented on top of the HeatSpace Unity 5 plugin. Installers define potential display positions and get realtime feedback on their visibility. The optimizer can be setup and triggered using the editor. Furthermore, installers can view the geometric persistence heatmap which also holds information on common user flow.

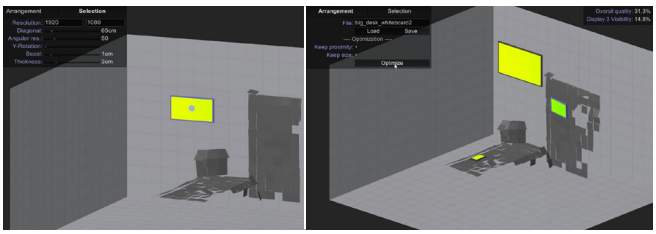

Figure 8. The HeatSpace editor allows the installer to add displays in a space and to set constraints such as initial placements or size (*left*). It automatically optimizes the position and size based on the installer's input (*right*).

EXAMPLES

As example of usage of HeatSpace we used the system while working on this paper. We recorded several hours of ourselves working, and rearranged the room several times for joint discussions on a whiteboard, individual paper writing, and proofreading. All examples are depicted in Figure 9.

Setup

We used 3 Kinects to capture a room with dimensions $4\,\mathrm{m} \times 4\,\mathrm{m} \times 4\,\mathrm{m}$. Since automatically placing displays does not require high resolution data, we opted for a real-world voxel side length of 12.5 cm. This results in $32^3 = 32768$ voxels in each of the heatmaps. The three examples yielded several hours of recording in total. The heatmaps for each example were 2.4 megabytes in size.

Weighting field of view and viewing angle

We chose a relatively small exponent as a parameter for the previously described field of view (typically $e_f = 0.3$). This means that surfaces in the periphery are still considered well suited for placing displays. We had in mind that users will not look at an empty wall, however this is likely to change if there was a display. In the periphery, while users have to take an effort to see a display (e. g. turn their head), they are likely to notice changes in content. If the goal was to minimize head rotation, the installer could choose a higher exponent. Similarly, the exponent for viewing angle was chosen to be small ($e_a = 0.35$). We believe that not looking completely straight at a display does not immediately reduce visibility. This would be different, e.g., with displays that have a low viewing angle range.

Example 1 — Room with couch: In the first example, one user was mostly seated on a couch in the center of the room, proofreading this paper. Occasionally, another person came through the door walking from the front of the room to the back for discussions. This is reflected in the geometric persistence heatmap h_g, shown in Figure 9, *bottom left*. The installer chose to add two big displays and a smaller one in the room, only constrained in resolution. Therefore, HeatSpace optimized the position and size of the displays. This resulted in two big displays positioned exactly in front of the user and the smaller display on the side (see Figure 9, *center left*). Qualitatively, this goes in line with the data from the visibility heatmap h_v, where the hotspot is in the corner where the displays were placed.

Example 2 — Two users at large desk: Two users were recorded sitting on opposite sides of a large desk (see Figure 9, *middle*), working on the text of this paper individually. Therefore, they were both blocking the views on the walls behind them. The installer added three displays, only constrained in resolution. For this example the installer configured the system to only optimize for one of the users. After optimization, the two displays close to the user have a smaller size than the display on the other side of the room.

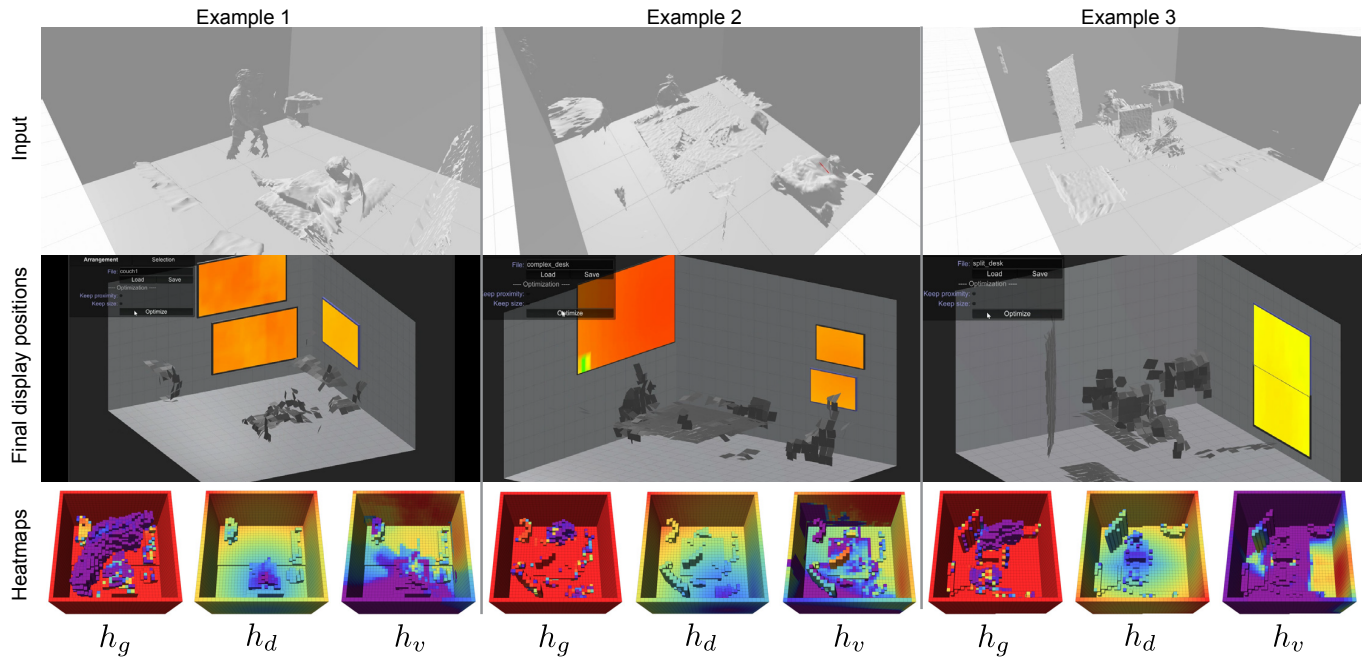

Figure 9. We recorded three examples with different number of users and different room configurations. *Top* shows the input mesh from the depth cameras, *center* the final display placements. *Bottom* hows the three calculated heatmaps for geometric persistence, distance and visibility, i. e. h_g, h_d and h_v respectively.

Example 3 — Two users at small desk: Two users who were sitting opposite each other at a small desk where recorded while working. The display size is unconstrained in this example. In contrast to example 2, the installer configured the system to optimize for both users. HeatSpace therefore optimized the position and size of the displays to be located on the wall, which is visible for both.

DISCUSSION AND FUTURE WORK

Nowadays, displays are usually placed by intuition. Especially for multi-purpose spaces that are shared among groups, "guessing" the correct display placement can be problematic. With HeatSpace, we automate finding suitable display surfaces by quantifying parameters such as distance and visibility. Note that our system trivially generalizes to non-digital displays, such as signs, artwork, noticeboards etc.

Scalability and capture time

Since all depth cameras are connected to individual computers, our setup can easily handle 5-10 and more depth cameras. The maximum number of cameras only depends on the computational power of the server running the Unity application. Alternatively, HeatSpace allows for recording all streams individually to combine and process them afterwards. This relaxes the realtime and network requirements, but increases the storage requirements considerably. The heatmaps itself require little storage, currently in the order of few megabytes, which does not increase over time. We imagine HeatSpace to run continuously over the course of several days or weeks to get a full picture of activity happening in a space.

Behavioral change

Installing displays changes the behavior of people. This is an inherent property of any planned change of the environment. Therefore, the best strategy would be to capture user behavior *before* any displays are installed. After display installation, the display areas would likely attract more views than before. One of the goals of HeatSpace is to position the displays so that there is a minimal behavioral change, i.e., to integrate displays seamlessly into the workflow and environment. Therefore, the placement of displays that intend to change the workflow, i.e., interactive touch-enabled displays, can not be optimized directly. However, display installations can be verified and refined.

Ceiling-suspended displays

Currently, HeatSpace automatically finds display placements on *existing geometry*. Our next step would be to also consider displays that are suspended from the ceiling or on stands. The key difficulty with such displays is that they can change the occlusion pattern in the room.

Digital content

The core concept can be used to optimize placement of digital content of already existing displays and projectors. This is especially interesting for dynamic content, like view-dependent perspective-corrected renderings. Extending our data structures would allow for optimizing placement of content shown through projection mapping.

Gaze tracking

While instrumenting users is impractical in our scenarios, the availability of accurate long-distance remote eye

trackers would enable an obvious extension of our system. Integrating gaze tracking in our system would be trivial, requiring no changes to data structures and algorithms. However, currently we optimize display placements to where they *could* easily be seen, assuming users would move their eyes to look at them. Optimizing for actual gaze direction would mean placing displays directly where users usually look, an approach that is not necessarily superior to our current approach.

Weighting of Events

HeatSpace currently averages heatmap data over time, which essentially ranks specific settings (e. g. 3 users in a room discussing a project) based on the relative amount of time they occur. However, some events, such as the presence of a manager —although occurring infrequently— might be very important and require a special room configuration. We plan to enable installers to rate the relative importance of recorded events.

Visibility parameters

Different exponents for field of view and viewing angle account for different types of content and scenarios. For instance, static content, like signs or paintings, require a large field of view exponent, since these are often overlooked in the periphery. Dynamic animated content on the other hand, is easily visible in the periphery. Currently, the installer has to set the importance of the visibility factors manually. Future iterations might investigate finding these parameters automatically based on color and movement of displayed content.

CONCLUSION

HeatSpace suggests the optimal position and size of display surfaces in a space based on user behavior. To do so, it gathers data from multiple depth cameras and calculates measures of visibility, geometric persistence, and distance with respect to users in the space. Data is stored in multiple volumetric heatmaps for fast computation and avoiding artifacts from noise in the data of the depth cameras. Display sizes and positions are optimized using alternating gradient descent. Our system allows to quantitatively evaluate display placements and aims at reducing guesswork for finding optimal display positions.

ACKNOWLEDGMENTS

This work has been supported by IFD grant no. 3067-00001B for the project entitled: MADE - A platform for future production.

REFERENCES

1. Blaine Bell, Steven Feiner, and Tobias Höllerer. 2001. View Management for Virtual and Augmented Reality. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST '01)*. ACM, New York, NY, USA, 101–110. DOI: http://dx.doi.org/10.1145/502348.502363
2. Michael L Benedikt. 1979. To take hold of space: isovists and isovist fields. *Environment and Planning B: Planning and design* 6, 1 (1979), 47–65.
3. Oliver Bimber, Daisuke Iwai, Gordon Wetzstein, and Anselm Grundhöfer. 2008. The Visual Computing of Projector-Camera Systems. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 2219–2245.
4. Jeremy Birnholtz, Lindsay Reynolds, Eli Luxenberg, Carl Gutwin, and Maryam Mustafa. 2010. Awareness Beyond the Desktop: Exploring Attention and Distraction with a Projected Peripheral-vision Display. In *Proceedings of Graphics Interface 2010 (GI '10)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 55–62. http://dl.acm.org/citation.cfm?id=1839214.1839225
5. Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. 1993. Surround-screen Projection-based Virtual Reality: The Design and Implementation of the CAVE. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93)*. ACM, New York, NY, USA, 135–142. DOI: http://dx.doi.org/10.1145/166117.166134
6. Jakub Dostal, Per Ola Kristensson, and Aaron Quigley. 2013. Subtle Gaze-dependent Techniques for Visualising Display Changes in Multi-display Environments. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces (IUI '13)*. ACM, 137–148. DOI: http://dx.doi.org/10.1145/2449396.2449416
7. Bernard Ghanem, Yuanhao Cao, and Peter Wonka. 2015. Designing Camera Networks by Convex Quadratic Programming. *Comput. Graph. Forum* 34, 2 (May 2015), 69–80. DOI: http://dx.doi.org/10.1111/cgf.12542
8. Christopher G. Healey and Amit P. Sawant. 2012. On the Limits of Resolution and Visual Angle in Visualization. *ACM Trans. Appl. Percept.* 9, 4, Article 20 (Oct. 2012), 21 pages. DOI: http://dx.doi.org/10.1145/2355598.2355603
9. B. Johanson, A. Fox, and T. Winograd. 2002. The Interactive Workspaces project: experiences with ubiquitous computing rooms. *IEEE Pervasive Computing* 1, 2 (April 2002), 67–74. DOI: http://dx.doi.org/10.1109/MPRV.2002.1012339
10. Brett Jones, Rajinder Sodhi, Michael Murdock, Ravish Mehra, Hrvoje Benko, Andrew Wilson, Eyal Ofek, Blair MacIntyre, Nikunj Raghuvanshi, and Lior Shapira. 2014. RoomAlive: Magical Experiences Enabled by Scalable, Adaptive Projector-camera Units. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, 637–644.

11. Christian Lander, Sven Gehring, Antonio Krüger, Sebastian Boring, and Andreas Bulling. 2015. GazeProjector: Accurate Gaze Estimation and Seamless Gaze Interaction Across Multiple Displays. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology (UIST '15)*. ACM, 395–404. DOI: http://dx.doi.org/10.1145/2807442.2807479

12. Lars Lischke, Sven Mayer, Katrin Wolf, Niels Henze, Harald Reiterer, and Albrecht Schmidt. 2016. Screen Arrangements and Interaction Areas for Large Display Work Places. In *Proceedings of the 5th ACM International Symposium on Pervasive Displays (PerDis '16)*. ACM, New York, NY, USA, 228–234. DOI: http://dx.doi.org/10.1145/2914920.2915027

13. Aaron Mavrinac and Xiang Chen. 2013. Modeling Coverage in Camera Networks: A Survey. *International Journal of Computer Vision* 101, 1 (2013), 205–226. DOI: http://dx.doi.org/10.1007/s11263-012-0587-7

14. Lori McCay-Peet, Mounia Lalmas, and Vidhya Navalpakkam. 2012. On Saliency, Affect and Focused Attention. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, 541–550. DOI: http://dx.doi.org/10.1145/2207676.2207751

15. Joseph O'Rourke. 1987. *Art Gallery Theorems and Algorithms*. Oxford University Press, Inc., New York, NY, USA.

16. Thies Pfeiffer. 2012. Measuring and Visualizing Attention in Space with 3D Attention Volumes. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12)*. ACM, 29–36. DOI:http://dx.doi.org/10.1145/2168556.2168560

17. Miguel A. Nacenta PhD, Carl Gutwin, Dzmitry Aliakseyeu, and Sriram Subramanian. 2009. There and Back Again: Cross-Display Object Movement in Multi-Display Environments. *Human–Computer Interaction* 24, 1-2 (2009), 170–229. DOI: http://dx.doi.org/10.1080/07370020902819882

18. Claudio S. Pinhanez. 2001. The Everywhere Displays Projector: A Device to Create Ubiquitous Graphical Interfaces. In *Proceedings of the 3rd International Conference on Ubiquitous Computing (UbiComp '01)*. Springer-Verlag, London, UK, UK, 315–331. http://dl.acm.org/citation.cfm?id=647987.741324

19. Marc Pomplun, Helge Ritter, and Boris Velichkovsky. 1996. Disambiguating Complex Visual Information: Towards Communication of Personal Views of a Scene. *Perception* 25, 8 (1996), 931–948. DOI:http://dx.doi.org/10.1068/p250931 PMID: 8938007.

20. Umar Rashid, Miguel A. Nacenta, and Aaron Quigley. 2012. Factors Influencing Visual Attention Switch in Multi-display User Interfaces: A Survey. In *Proceedings of the 2012 International Symposium on Pervasive Displays (PerDis '12)*. ACM, New York, NY, USA, Article 1, 6 pages. DOI:http://dx.doi.org/10.1145/2307798.2307799

21. George Robertson, Mary Czerwinski, Patrick Baudisch, Brian Meyers, Daniel Robbins, Greg Smith, and Desney Tan. 2005. The Large-Display User Experience. *IEEE Comput. Graph. Appl.* 25, 4 (July 2005), 44–51. DOI: http://dx.doi.org/10.1109/MCG.2005.88

22. Brian A. Smith, Qi Yin, Steven K. Feiner, and Shree K. Nayar. 2013. Gaze Locking: Passive Eye Contact Detection for Human-object Interaction. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, 271–280. DOI: http://dx.doi.org/10.1145/2501988.2501994

23. Sophie Stellmach, Lennart Nacke, and Raimund Dachselt. 2010. 3D Attentional Maps: Aggregated Gaze Visualizations in Three-dimensional Virtual Environments. In *Proceedings of the International Conference on Advanced Visual Interfaces (AVI '10)*. ACM, 345–348. DOI: http://dx.doi.org/10.1145/1842993.1843058

24. Norbert A. Streitz, Jörg Geißler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. 1999. i-LAND: An Interactive Landscape for Creativity and Innovation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 120–127. DOI: http://dx.doi.org/10.1145/302979.303010

25. Yusuke Sugano, Xucong Zhang, and Andreas Bulling. 2016. AggreGaze: Collective Estimation of Audience Attention on Public Displays. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 821–831. DOI: http://dx.doi.org/10.1145/2984511.2984536

26. Tasos Varoudis and Sophia Psarra. 2014. Beyond two dimensions: architecture through three dimensional visibility graph analysis. *The Journal of Space Syntax* 5, 1 (2014), 91–108.

27. James R. Wallace, Stacey D. Scott, and Carolyn G. MacGregor. 2013. Collaborative Sensemaking on a Digital Tabletop and Personal Tablets: Prioritization, Comparisons, and Tableaux. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 3345–3354. DOI: http://dx.doi.org/10.1145/2470654.2466458

28. Robert Walter, Andreas Bulling, David Lindlbauer, Martin Schuessler, and Jörg Müller. 2015. Analyzing Visual Attention During Whole Body Interaction with Public Displays. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. ACM, 1263–1267. DOI: http://dx.doi.org/10.1145/2750858.2804255

29. David S. Wooding. 2002. Fixation Maps: Quantifying Eye-movement Traces. In *Proceedings of the 2002 Symposium on Eye Tracking Research & Applications (ETRA '02)*. ACM, 31–36. DOI: http://dx.doi.org/10.1145/507072.507078

30. Martina Ziefle. 1998. Effects of Display Resolution on Visual Performance. *Human Factors* 40, 4 (1998), 554–568. DOI: http://dx.doi.org/10.1518/001872098779649355