Multimedia Data Processing in Heterogeneous Distributed Environment

EXTENDED ABSTRACT OF A PHD THESIS

Rudolf Kajan
http://www.fit.vutbr.cz/ ikajanr/

Department of Computer Systems
Faculty of Information Technology
Brno University of Technology

2015

It's not enough that we build products that function, that are understandable and usable, we also need to build products that bring joy and excitement, pleasure and fun, and yes, beauty to people's lives.

—Don Norman

Contents

1 Introduction

Current handheld devices have the display and computational capabilities of common desktop machines from several years ago. What is lacking are new methods of interacting with these ultramobile devices that uniquely suit mobile interaction, rather than derivatives of the desktop interface [11]. This work addresses the question about intuitive content sharing and information reaccess between user's personal device and another device. Its core part deals with the development of a interaction technique which supports unobtrusive content exchange between touch-enabled personal device and a large display – whether it is shared-private or public.

Much of the motivation for this work arises from a literature review in human-computer interaction, psychology and informative visualization. A study by Bales et al. [2] focused on methods and content of web information reaccess among personal devices. It showed that cross-device content reaccess is often very spontaneous and unplanned and that currently native applications play an important role in how users reaccess content. Unfortunately, contemporary solutions for content sharing and information reaccess are mainly document-centric and rely on complicated infrastructure, thus creating barriers for users trying to share information and collaborate [17]. Dearman et al. [8] state that for most users, the migration from PC to a mobile platform is more frustrating than any other means of follow-up. The main source of this frustration is the fact that users are often forced to use many creative, although very time-consuming, methods to enable content and task migration among their de-

vices, because of lack of support from software.

As Bales at al. [2] state, users would often use features that were made for different purposes as methods to find information later. Many tools, such as Context Clipboard, Evernote, and Dropbox, have attempted to address this problem by enabling easy capture and reaccess, such as saving a link to find later [10]. Although these tools are seamless and easy to use, they still require planning on the part of the user.

Dearman et al. [8] observed in their study, that among most commonly used means of content reaccess were: **Leaving browser tabs open** on the mobile device as a reminder to reaccess them on another device. **Using handwritten or printed information**, carried between the devices, and inputted on the second device. Utilizing **shared bookmark systems** to access data between devices. Using these systems users save a bookmark on one device and have it available on their other device automatically. Unplanned reaccesses are frequently executed by entering **search queries** into another device.

For a vast majority of applications, the initial assumption is still that users interact with just a single computing device throughout the day. The high quality of smartphone interfaces and always-on connectivity have changed how phones are used today, with many phones being used more like desktops [2]. The practical consequence of this assumption is the lack of collaboration among devices and user-centric activities that may span multiple devices as well as multiple applications.

While there are initial steps in this direction [3], these solutions must support a wider variety of activities and fully recognize the members of a user's device collection. Researchers have proposed supporting activities that span multiple applications [6]. However, we need to go farther and provide users with a lightweight solution for information transfer able to work with different types of information and contexts, to respect the need for privacy support and supports additional metadata

generated through interaction which is useful for future interactions on other devices. These findings suggest focusing on the user and his intent rather than on applications and devices, making devices aware of their roles and focusing on lighter-weight methods for information transfer and task synchronization / migration.

We are introducing ways for the mobile phone user to unobtrusively interact with another device (public display, desktop, tablet, ...). The user interaction needed to achieve the communication goal is done on the mobile device to spare the user the need to use two or more controlling appliances at a time. Our main contributions are:

- We present novel interaction techniques for continuous interaction across mobile and desktop platforms;
- We present method for real-time application state acquisition and reconstruction on target platform;
- We report on user studies focused on the usability of our prototypes and system performance evaluations.

For the purposes of this work, it is assumed that the sudden changes in network connectivity are addressed by the underlying infrastructure so that the reliable and persistent connection between the content provider display and a personal device can be established. Furthermore we assume that the communication channel is secured by a suite of protocols which provide data source authentication, data integrity, confidentiality and protection against attacks. Both of these assumptions can be met by utilization of already existing technologies (See Chapter 3), therefore it is safe to make these assumptions.

2 Interacting in Heterogeneous Environment

More than 15 years ago, Mark Weiser proposed ubiquitous computing, a paradigm in which the processing of information is linked with each activity or object as encountered, as the next era for interacting with computers. Ubiquitous computing focuses on learning by removing the complexity of computing and increases efficiency while using computing for different daily activities.

Weiser foresaw miniaturized and inter-connected devices embedded into our environment, playing a crucial role in our daily activities. Given today's broad availability and utilization of smartphones, tablets, sensor networks, public displays and intelligent wearable devices it is clear, that Weiser's vision is becoming a reality.

What is important to realize is that the vision of post-desktop model of computing introduced by Weiser, Brown and other researchers goes far beyond just devices and encompasses also their integration into the environment and describes interactions with these smart environments. Weiser predicted that technology would be seamlessly integrated into objects and spaces utilized by people on a daily basis, whether they are public or private, where it would be accessible at any time:

"*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life, until they are indistinguishable from it*"[24].

While we already have enabling technologies for mobile com-

puting and distributed systems, it is still necessary to fill the "missing link" between the conventional desktop metaphor and the multi-device era. To satisfy this need, researchers have explored the possibility of harnessing the input and output capabilities of different devices to improve user interaction with a large variety of devices.

Paak et al. observed that there are two major trends in research regarding public, or shared computing scenarios [19].

The first category focuses on large, individual displays with large high-resolution screens and simple input modality, given by the type of display and its primary purpose, usually keyboard, pointing device or touchscreen. These displays are commonly used as public content providers – providing navigation or information services. While they are usually equipped with advanced computing and networking capabilities, they frequently limit interaction to those who have direct access to the input device.

The second category focuses on mobile and ultramobile personal devices – smartphones, tablets or laptops. These devices act usually as both input and output devices and offer complex input modalities – computer vision, speech recognition, motion or orientation. They allow for interaction scenarios where interaction is distributed in space among a number of users. On the other hand, smaller screen sizes and usually low battery lifetime often hinder dynamic interaction [19].

What is missing from contemporary shared displays is interactivity and computation [19]. Event in cases where the display is interactive, usually only single simple input modality is available for user. Besides this, current shared and public display are making it very difficult for users to transfer and share their content and merge this publicly available content with private information (e.g., add publicly available event date to a private calendar or build list of preferences and store it on a private device).

Ultramobile devices and situated displays have a great potential for synergy, given by their complementary properties. Their coupling enables to combine personalised interactivity of a small (ultra)mobile and personal device with presentation space of a large, publicly available display [21]. Only with appropriate design, and by using technologies and interaction metaphors that parallel the way the user thinks about a task as closely as possible, are we able to achieve intuitive content sharing and information reaccess between user's personal device and a situated display. Enabling this type of interaction, all in the context of Weiser's vision of "Calm Technology", is the goal of our work.

3 On-Screen Marker Fields for Reliable Screen-To-Screen Task Migration

Our goal is to deliver unobtrusive task migration, typically between a desktop computer and a mobile device. We propose to overlay an aesthetically acceptable marker field across (a part of) the monitor screen. This marker field must be easily detectable even by a low-end ultramobile device, unobtrusive to the user, and easy to mix in the natural screen image. We show that Uniform Marker Fields are a good choice for this task and propose a methodology for inserting them into the screen image. The experimental results show that our solution provides reliable task migration on a video stream in interactive frame rates ($\sim$ 30 FPS marker detection, 340 ms whole processing time including wireless communication, HTC Desire from 2010). This substantially outperforms the existing solutions based on natural keypoints ($\sim$ 7 sec processing). Our user tests also led to selection and design of appropriate marker fields and their mixing parameters. This chapter is based on our published paper [16].

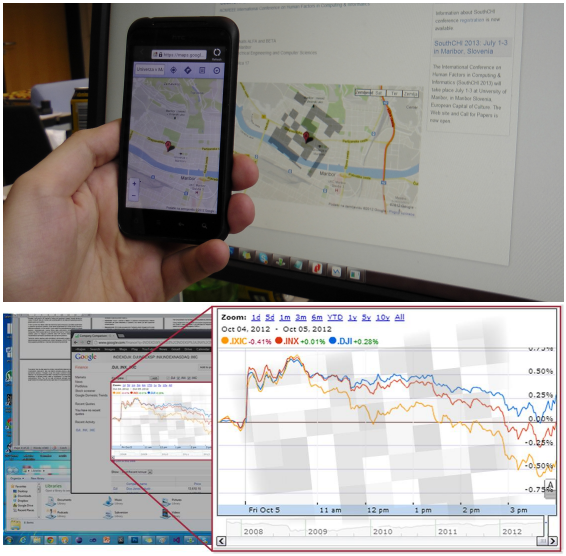

Figure 3.1: The goal of our work is to insert a subtle piece of information into the monitor screen that would be reliably detectable and could accurately establish the location within the screen observed by an ultramobile device.

3.1 High-Level Overview of the Task Migration System

The main objective for the design and implementation of our task migration system was to transfer tasks and information among large variety of devices while minimizing configuration time and being as intuitive as possible. Our system supports two main roles for devices – the *content provider* role and the

content requesters role. The device in the role of *content provider* is able to share the state of its applications with authenticated clients – *content requesters*. This device handles incoming connections, maintains context and streams requested content. The *content provider* device provides the state of its applications by either querying individual applications for their current work state (URL and internal settings for web applications, document along with current page number for document viewers, streamed multimedia content) or provides general services, e.g. providing high quality screenshots of a selected screen area or text from a selected area, possibly via optical character recognition. A typical *content provider* is expected to be a plugin into existing widely-used software: common web browsers, mail clients, office applications, possibly even integrated development environments. *Content requesters* are responsible for communication initiation with the target provider device, selection of screen region or application and selection of requested/offered content based on the user's intent. In a typical scenario, *content requesters* are mobile or ultra–mobile devices (mobile phone, smart phone, tablet, PDA). The communication between devices is realized through a WiFi connection. Messages which are exchanged among system components are structured key-value pairs based on Javascript Object Notation (JSON). Binary data in messages are encoded in Base64. Using standardized and open protocols ensures portability of provider / requester implementations.

The *marker field detector* on the *requester* device is used for fast and accurate client-side on-screen marker decoding and sending of within-screen localization information to the target device via the *network session manager*. This approach minimizes the amount of transferred data between the devices because only the unique ID of the currently detected part of the marker field and the homograph data is sent back to the *content provider* (unlike feature-based solutions where either feature vectors or

the whole camera stream are sent to the target device or to an intermediate server for processing and camera localization).

Based on the obtained camera-localization information, the *context manager* on the *content provider* device queries individual applications and gathers their status. In order to obtain the application status from web applications, we have implemented an extension for the Google Chrome browser which is able to forward application state requests from our system and return gathered state information for further processing.

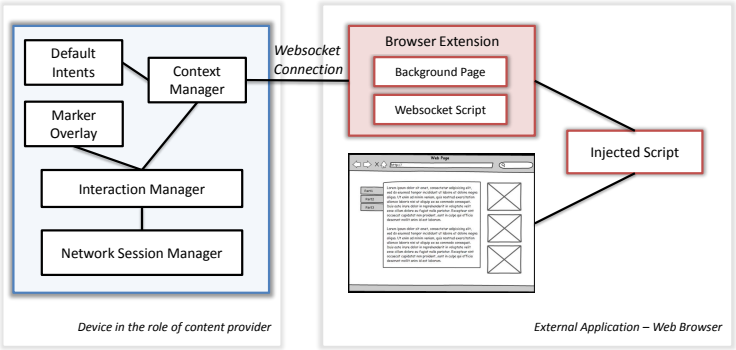

Figure 3.2: Example of communication with external application – web browser via browser extension. The implemented browser extension is able to extract state of web application through javascript code injected into the opened web page.

The Google Chrome extension is implemented as a persistent Chrome background page, which runs on the *content provider* device even when main browsers windows are closed. The extension communicates with the *context manager* via websocket – full-duplex single socket connection. After receiving a request, the extension finds the active browser window and the tab with

a web page and through code injection inserts javascript script into the page. This script is able to extract required parts of the web page - blocks of text, images, videos or links to other web-based resources and send them back to the *content provider*, which forwards them to the *requester device* (See Fig. 3.2).

If the selected application is unable to provide its state and metadata, only general intents are available. General intents include high-quality screenshots, text and phone numbers recognitions for the selected part of the screen.

The acquired application state is sent to the *intent manager* on the *requester device* which translates these JSON-encoded messages to intents directly usable on the requester platform (e.g., on the Android platform creates Android intents from JSON messages).

Afterwards, the *state manager* provides the user with a visual feedback and updates the GUI, based on available actions for the selected content. The options include resuming work with web application on current device, editing text in available text editor, manipulation and viewing of images or audio/video playback.

If the user decides to continue the task with a selected application (e.g. web browser or media player), the *application dispatcher* sends the intent along with context to the appropriate application on the *requester device*.

3.2 Results: User Testing and Technical Evaluation

We performed a user test to find out the most acceptable shape of the marker field and the parameters of mixing it into the desktop screen image. The technical evaluation involves tests of reliability of detection of the marker field on different screen

contents and under different viewing poses. Finally, computational performance is evaluated.

3.2.1 Marker Unobtrusiveness

We conducted an initial user study to observe how people would use our prototype. Our main goal was to find out how obtrusive was the usage of marker fields for task migration for participants and whether this approach is feasible also for inexperienced users.

The study we conducted consisted of 11 participants (8 male, 3 female). All attendees use at least one ultra-mobile device (mobile phone, smart phone, tablet, PDA) and one desktop computer or laptop on a daily basis. The average device count per participant was 2.9.

In the beginning, participants had to fill in a questionnaire. This questionnaire asked them about their technical expertise, their knowledge regarding mobile phones, as well as some demographic statements. Subsequently we introduced our system and three basic task migration scenarios – map state transfer, acquisition of short textual information from a web page and acquisition of an image from a long online article with a photo gallery.

In order to be able to compare feedback from participants, we have created ten marker presets divided into four categories (Fig. 3.3).

We provided the participants with an Android smartphone and a laptop with our system; the laptop also contained an application which allowed fast change of marker parameters from presets. Participants were asked to rate marker presets based on perceived obtrusiveness on a five point Likert scale (1–least obtrusive, 5–most obtrusive). The average rating across all presets was 2.47. Markers with high opacity were perceived as most obtrusive (average rating 3.76), while markers with 20%

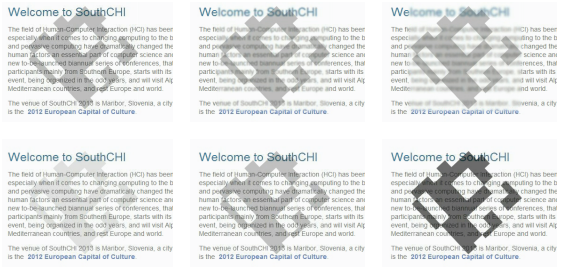

Figure 3.3: Examples of mixed-in marker field with varying presets. **top row:** different background blur radii. **bottom row:** varying opacity levels (20, 40, 60 %).

and 40% opacity had similar, significantly lower, average rating (1.67 and 2.12). Among our participants, the presence of blurred background or periodic changes in marker opacity had only minimal influence on perceived obtrusiveness. Application of blur on the background decreases the amount of natural edges present in the image and allows for the marker field edges to prevail. Similarly, the pulsing intensity of the marker allows for periodic appearances of highly opaque form of the marker field, which can be tracked afterwards or at least can provide localization information in discrete time frames. The fact that the users tend to tolerate these modes of mixing, offers truly reliable on-screen localization with acceptable levels of obtrusiveness.

In general, our system was perceived very positively, with 81% of participants stating that it would definitely help them with content reaccess. 63% of participants would use it to obtain information from public displays. In this case, the biggest concern were privacy issues – fear that a publicly available system could access private information stored in mobile devices

due to security flaws.

3.2.2 Marker Detection Reliability

We tested the accuracy and reliability of our marker detection algorithm, with the marker mixed into natural screen contents. We created a setup consisting of one device in the role of content provider and one device in the role of content requester. As a content provider device we used a laptop computer with a high-resolution (1920×1080) display, and an Android 4.0.4 smartphone HTC Incredible S for the role of information requester device. The requester device was attached to a base perpendicular to the floor, in a fixed height, focused at a chosen part of the screen. The experiment was conducted in a room with artificial (fluorescent) lighting. Devices were connected through a WiFi connection. The requester device was held at a distance of 10 (on graphs shown as near), 20 and 40 cm (on graphs shown as far) and a pitch angle of 70°, 90°, 110°, 130°, and 150°. On the content provider device a fullscreen webpage containing text, several smaller images, and a map was displayed during the experiment. The pulsing period for the pulsing marker was set to 1.5 seconds. The frame rate of the camera was 25 FPS.

Fig. 3.4 shows some of the reliability tests based on the display tilt and the distance between the camera and the display. For the static (non-pulsing) marker mixing we used the average transparency used in the case of the pulsing marker. We chose from each set with different pitch at least one histogram. Between 90° and 110° the results were consistent with the data presented in Fig. 3.4 and thus they are omitted. The results show that even for 150° pitch angle (which was outside the viewing angles for the monitor) and from a large distance the pulsing marker was still reliably detectable within 5 frames (0.2 seconds) with 90 % probability.

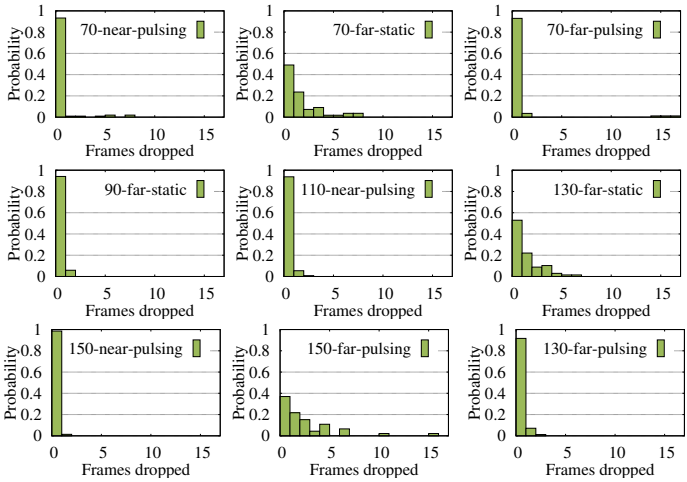

Figure 3.4: Selected results of the reliability test. The histograms show the probability distribution of the time (in frame count) between consecutive successful position identifications. The name of the histogram corresponds to the pitch angle, distance from the content provider plane and whether the marker was pulsing (the pulsing period calculated in frames was 37.5 frames).

The results indicate that the pulsing marker gave consistently better results then the static marker, especially at the extreme pitch angles (150°, 70°). This is mainly due to the fact that it is tricky to calibrate the correct transparency level for the static marker that is unobtrusive but still detectable by the smartphone camera for all the possible display pitch angles. Notice that histograms for the pulsing markers show two main peaks (Fig. 3.4 – 70-near-pulsing, 150-far-pulsing). The first one is near

zero, since while the marker transparency is lower, the marker is detected in each frame. The second peak marks the interval, where the marker is too transparent during the pulsing period for the algorithm to be able to detect it (in the 70-near-pulsing histogram it is at 15 dropped frames which is about the third of the pulsing period).

3.2.3 Speed Performance

In order to be able to compare our solution to other frameworks (primarily DeepShot [5]), we have tested four target applications: maps from Google Maps, photos from Picasa, long articles with images from CNN.com and short textual information from Twitter. For each application, ten information requests were sent and processed. The setup for this experiment was identical to the previous test.

The average time for all forty requests was 336 milliseconds (Standard Deviation 67 milliseconds). Table 4.2 summarizes an approximate breakdown of the time consumed in different phases of the processing. The marker field decoding (the "computational part" of the process) takes around one third of the time, whereas the rest is spent in network communication and related activities. From the marker field decoding part the edgel extraction and edge classification required on average $\sim$60 % of the time on an older single core smartphone (HTC Desire with 1GHz ARM v7 processor released in 2010) even with the very low "pixel footprint" of the detection algorithm. The overall average time required by our naive implementation for mobile platforms excluding the system overhead to acquire the image was 34 ms ($\sim$30 FPS) for 800×480 resolution extracting on average $\sim$120 edgels. We used standard web-site content as the background during these evaluations with the marker constantly visible in the field-of-view. Results might differ considerably for different smartphones or display content.

Table 3.1: Timing breakdown of the mobile client.

activity	% time spent
connection initialization	28 %
marker field detect & decode	34 %
network transfers	13 %
information retrieval	25 %

The results show a significant speed increase when compared to task migration solutions based on visual features. When compared to the DeepShot [5] task migration framework, our solution is on average 20.3× faster (authors of the DeepShot report 7.7 seconds (Standard Deviation 0.3 seconds) for processing the frame) and allows for real-time information feedback for a selected screen area. At the same time, our solution operates on a video stream with all the benefits: if one camera frame fails for any reason, the mobile client program determines the location from a subsequent valid one, etc.

3.3 Implications

We presented a technical solution for reliable and unobtrusive interaction between a ultramobile device and another (typically desktop) monitor screen. We proposed a system architecture harnessing this communication in order to achieve task migration between the devices. This task migration procedure is designed to impose minimalistic requirements on the system setup and at the same time to be straightforward and intuitive to the user.

We measured the performance of the system and the results show that it substantially outperforms the existing solutions: the detection and recognition of the marker field is done in real

time on a mid-level cellphone on a video stream; the recognition is reliable even for different observation angles and for cluttered screen content. We conducted a user study to select an as unobtrusive as possible variant of the marker field and the mixing parameters.

Our system allows for direct task migration, without any direct interaction with the desktop system – straight from the mobile device. Thanks to the usage of the marker field, the detection is reliable, because the system ensures reliable and easily detectable keypoints in the monitor screen. At the same time, the marker field proves to be unobtrusive and aesthetically tolerable by the user.

4 Video Recording – A Promising Metaphor for Inter-Device Task Migration

Nowadays, people are interacting with an ever increasing number of devices - smartphones, public displays, kiosks or desktop computers. Currently, there is a lack of support for seamless task migration among devices - starting a task on one device and continuing it on another, without the need of manual application state inspection and data transfer.

In this chapter we present a way to make this task migration or content sharing process instant and intuitive. We are solving this problem by employing our framework for application state acquisition coupled with user interface based on an intuitive metaphor: video recording. Our interface allows for *continuous interaction* - mobile device's display is updated in real-time and receives continuous feedback. In every moment, user is given relevant task and content-migration options for selected application. Our approach thus emphasizes spontaneous and unplanned content access with minimal user input, while being very responsive. This a significant enhancement when compared to our previous approach, which allowed only for discreet interaction.

The experimental results show that our solution provides reliable task migration at interactive frame rates. This substantially outperforms the existing solutions. We carried out a user study as well as empirical evaluation tests. The results indicate that the system is perceived as intuitive, easy to learn and ef-

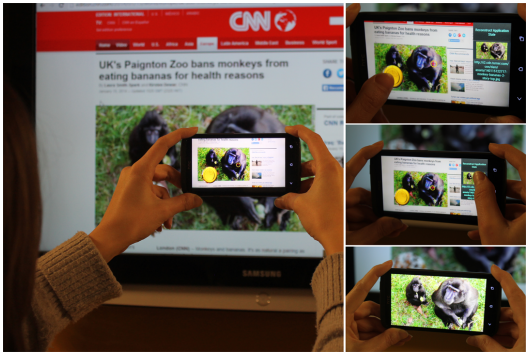

Figure 4.1: In our work, we aim at *continuous* exchange of information between a large screen (a desktop computer, a public kiosk, etc.) and a mobile device. This information exchange should be visual and intuitive: based on the metaphor of "video recording" with the mobile camera.

fective in transferring ongoing tasks between the desktop/kiosk and a mobile device.

We make the following contributions with our system: a) We present a novel interaction technique for *continuous* interaction across mobile and desktop platforms; b) We present a framework for real-time application state acquisition and reconstruction on target platform; c) We report on a user study focused on the usability of our prototype and a system performance evaluation. This chapter is based on our published paper [15].

4.1 System Architecture

With the *continuous interaction* in mind, we have designed a highly responsive system (See section Speed Performance) which allows for intuitive task migration without the need of manual application state inspection or copying of "raw" pixels without any additional semantic information. The task migration process from the system architecture's point of view is a two-way communication between a *content provider* and a *content requester* device.

The *content provider* device is the device with the application state that needs to be transferred to the other device or platform based on the user's current context. A device in this role is able to share the state of its applications with authenticated clients – *content requesters*. The *content provider* device provides the state of its applications by either querying individual applications for their current work state (URL and internal settings for web applications, the document along with current page number for document viewers, streamed multimedia content, and other metadata) or provides general services, e.g. providing high quality screenshots of a selected screen area or text from a selected area via optical character recognition.

Content requesters are responsible for communication initiation with the target provider device, for selection of the screen region or application of interest and selection of requested/offered content based on the user's intent. In a typical scenario, *content requesters* are mobile or ultra-mobile devices (smart phone, tablet, PDA).

4.1.1 System Components and Their Functionality

Prior to the task migration, a network connection between the content requester and the provider must be established. The *network session manager* module, which is present on both devices, is responsible for network connection initiation to a remote *content provider* (e.g. public display, laptop). At the moment, the communication is implemented through a WiFi connection, due to its availability on a broad range of devices.

The target device is located either via network discovery or by scanning a specific code associated with target device (e.g. on-screen or printed visual marker / matrix code). Sudden changes in network connectivity are addressed by usage of MobileIP [1] in the existing infrastructure. MobileIP is able to seamlessly handle connection hand-overs during migration in a way that is transparent to the *content provider*. The communication channel is secured by IPSec - suite of protocols which provide data source authentication, data integrity, confidentiality and protection against attacks [2].

The *localization manager* on the *requester* device is used for fast and accurate client-side within-screen localization. The localization is based either on marker tracking or natural image keypoint detection, features extraction and matching. Natural features based detection is employed during the initial position estimation and when the devices cannot successfully track the marker. This approach minimizes the amount of transferred data between the devices, because only the detected 2D position coordinates are sent back to the *content provider*.

The transferred and processed content is much smaller compared to the purely feature-based solutions where either the feature vectors or the whole camera stream are sent to the target

[1] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=656077
[2] http://tools.ietf.org/html/rfc3776

device or to an intermediate server for processing and camera localization. This allows for fast and reliable real-time interaction with immediate feedback even on low-end devices.

Based on the obtained camera-localization information, the *provider's context manager* queries individual applications and gathers their internal state. In order to obtain the full application state from web applications, we have implemented an extension for Google Chrome browser.

If the selected application is unable to provide its state and metadata, only general intents are available. General intents include high-quality screenshots, and text and phone numbers recognition for the selected part of the screen (the OCR functionality is implemented via Microsoft MODI library[3]).

The acquired application state is sent to the *intent manager* on the *requester device* which translates these JSON-encoded messages to intents directly usable on the requester platform.

Afterwards, the *state manager* provides the user with a visual feedback and updates the GUI, based on the available actions for the selected content. The options include resuming work on a *requester device* – continuing work with a reconstructed web application state on a current device, editing text in an available text editor, manipulation and viewing of images, audio/video playback, etc.

4.1.2 Marker Tracking and Natural Keypoints based Detection

Our solution utilizes a combination of natural features based detection and marker tracking in order to reliably establish the homography between the screen and the observation of the mobile device's camera. This allows us to employ a fast and precise

[3] http://support.microsoft.com/kb/982760
[4] http://www.themaninblue.com/experiment/BunnyHunt

continuous interaction even on low-end mobile devices.

During the initialization phase and in case of fast camera movement, we employ natural features based detection. Detecting keypoints and extracting features on the mobile phone would be too costly on some low-end devices. Instead, the features are computed and matched on the content provider. Similar approach was taken in [5] and [4]. The difference is, that our solution does not stream the video, as it would generate high network traffic (see experiments). Instead, we use natural features detection as a fallback method, and send frames only in large intervals.

A major disadvantage of pure natural features based methods is that they rely on rich features being present on the target display. This assumption is rarely met in the highly manhattanic world of desktop and web applications. As a solution, we utilize a *virtual cursor* using the Vuforia[5] library on the content requester side combined with a small natural image target on the content provider. The natural image target is used to compute the required offset on the content provider caused by the camera movement. The computed relative correction is sent to the content provider. This is our primary method of camera movement tracking.

The image target also serves a secondary objective as a reference position to draw the augmented UI elements of the application. These elements give visual feedback to the users, so they move within acceptable distance from the content provider. The augmented layer also hides the obtrusive marker on the client side.

If multiple users are simultaneously interacting with a single provider, their primary mean of localization is natural features based detection of a target. In the case that multiple targets overlap, clients automatically fall back to natural keypoints

[5]https://www.vuforia.com/

tracking on the target display. This approach ensures that the interaction will not be interrupted even if multiple users are migrating the same elements at the same time.

4.2 Experiments: Empirical Tests and User Study

We conducted an initial user study to observe how people would use our prototype. Our main goal was to find out whether this approach is feasible also for inexperienced users.

The study we conducted involved 25 participants. 72% of participants use multiple devices for content reaccess on a daily basis. Table 4.1 shows the reported usage of most frequently used methods of content reaccess by the participants. Our results support previous findings obtained by Dearman et al. [8].

Method of content reaccess	**Reported usage**
File synchronization service	88%
Search queries	76%
Flash disks or external drives	60%
Shared bookmark systems	56%
Leave browser tabs open as a reminder	40%
Handwritten or printed notes	16%

Table 4.1: Reported usage of most frequently used methods of content reaccess.

After filling in the questionnaire we introduced our system and four basic task migration scenarios – map state transfer, acquisition of textual information from a web page, acquisition of an image from a long online article with a photo gallery and resuming writing of a text on a laptop computer.

We provided the participants with an Android smartphone and a laptop with our system; the laptop also contained an application which allowed evaluation of the movement accuracy. 72 % of the participants were able to finish the tasks under 6 s on average (not counting application startup). The participants needed on average 9.5 s (SD 2.3 s) to reach the goal of each task from application startup and network initialization to successfully migrating content. The DeepShot [5] task migration framework authors report 7.7 s for frame processing only.

In general, our system was perceived very positively, with 86% of participants stating that it would definitely help them with content reaccess. 72% of participants would use it to obtain information from public displays. Among the participants the most valued features of our system were simplicity of usage (81% of participants), high response rate and fluidity of interaction (78% of participants) and the ability to migrate several elements in one quick session (67% of participants).

The biggest concern were privacy issues – fear that a publicly available system could access private information stored in mobile devices due to security flaws or identify individual users and track their actions.

These results support our initial hypothesis that *continuous interaction* outperforms the two–phase interaction (acquisition followed by processing for each object).

4.2.1 Tracking Reliability

We tested both the reliability of our feature detection-based algorithm and the tracking performance of the Vuforia library as a part of the user tests. We created a setup consisting of one device in the role of content provider (17 inch laptop computer with 1920×1080 resolution display and an Intel®Core^{TM}i7 processor running at 2.2 GHz) and one device in the role of content requester (Samsung Galaxy S2 smartphone).

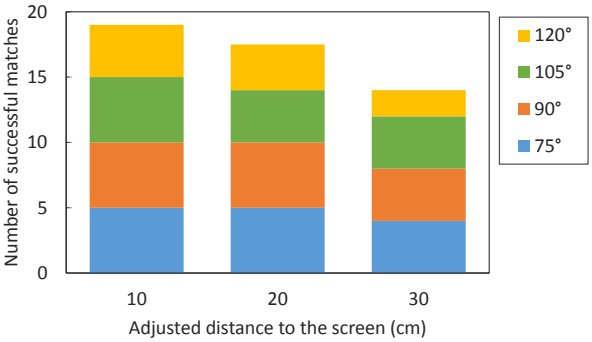

Figure 4.2: The results of natural feature detection reliability for different pitch angles. For each distance settings 20 images were taken - 5 per each angle.

In order to evaluate the reliability of the feature detection based algorithm used during initiation, the requester device was attached to a base perpendicular to the floor, in a fixed height, focused at a chosen part of the screen. The experiment was conducted in a room with artificial (fluorescent) lighting. The devices were connected through a WiFi connection.

The requester device was held at a distance of 10, 20 and 30 cm and a pitch angle of 75°, 90°, 105°, and 120°. On the content provider device a fullscreen web application containing text, several smaller images, and a map was displayed during the experiment. The resolution of the images sent to the content provider to compute the initial homography was 320×240.

Figure 4.2 contains the evaluation of the reliability of our natural feature based detection for different pitch angles for the content provider device. For each distance settings 20 images were taken – 5 per each angle. The results show that the natural features based detection was highly reliable. For angles 120°

and 75° the colors shown on the content provider were highly shifted changing the visible key-point features causing slightly worse results. This issue is caused mostly by the display used in testing and would harm any computer vision based technique.

During user testing, the participants were given several tasks to migrate data. During the tasks we recorded the tracking status of our system. The detection algorithm ran at 20 frames per second. After the tracking was lost, a full frame was transferred to the content provider in order to be used for natural features based detection (hence the step around the 1*s* mark). The delay interval of 1 second for full frame sending was chosen not to overload the network connection. The results show that after 4*s* the cursor tracking algorithm was able to restore tracking with 99 % probability. This causes a short but noticeable delay for the user after the tracking is lost and needs to be restored. Despite this delay, the overall performance of the natural feature detection is good enough to provide the users with *continuous interaction*, and is an area which we are planning to improve.

4.2.2 Speed Performance

In order to be able to compare our solution with alternative frameworks (e.g. Touch Projector or DeepShot), we tested four target applications: maps from Google Maps, photos from Picasa, long articles with images from CNN.com and short textual information from Twitter. For each application, 10 information requests were sent and processed. The setup for this experiment was identical to the previous test. All tests were done using the hardware from the setup for the reliability testing (See section Tracking Reliability above).

Table 4.2 summarizes the breakdown of the time consumed by the initiation phase of the interaction for a single frame. The majority of the time (59.2 %) was consumed by network transfer of the reference image. This gives 1.3 FPS for the natural fea-

tures based position estimation part. In the setup experiment we sent reference images in 1*s* intervals to avoid flooding the network.

Activity	**Time spent**
Client side processing	11 ms
Network transfers (WiFi)	442 ms
Provider-side processing	289 ms
State acquisition via plugin	4 ms
Sum	746 ms

Table 4.2: Timing breakdown of the initialization phase. Client side processing covers camera image retrieval and resizing operation. Provider-side processing includes image reconstruction, acquiring screenshot and homography calculations.

Once the tracker was initialized, it was able to track the cursor with full 20 FPS speed provided by the camera on the tested smartphone. After the user decided to migrate content from the content provider, the required time to transfer information was 19 *ms* on average including network communication (approximately 73 %).

The results show a significant speed increase when compared to task migration solutions based on visual features – the authors of the DeepShot [5] task migration framework report 7.7 seconds (SD 0.3 seconds) for processing the request, and allows for real-time information feedback for a selected screen area. A big advantage of our system is the utilization of video stream, which enables continuous interaction instead of discreet selection.

4.2.3 Content Selection Accuracy

In order to measure accuracy of content selection with our system, we have used targeting tasks based on ISO 9241-9 standard [18]. However, we have used a rectangular target instead of a distinct target point. We asked participants to try to navigate pointer into the rectangular area, while being as fast as possible.

The task started after the connection between requester and provider devices was established and the tracking subsystem was fully initialized. Afterwards users were notified about trial's start and moved the virtual cursor inside the area filled with text or images. The task ended once the cursor was inside the area and user touched the content acquisition button with non-dominant hand. We measured time and virtual cursor's coordinates throughout trials.

When compared to commonly used pointing devices, our system had a lower Throughput (TP), but also lower Error Rate (ER) for primary migration targets - images, text paragraphs, links (in [18] the reported values were: joystick TP 1.8 bps ER 9%, touchpad TP 2.9 bps ER 7%, trackball TP 3.0 bps ER 8.6%, mouse TP 4.9 bps ER 9.4%).

These results show that our system is comparable to commonly used pointing devices and usable even by inexperienced users. In the near future, we will improve both Throughput and Error Rate. We will compensate for natural hand tremble (which is the main source of lower TP and higher ER) by employing a smooth estimate of cursor's position and add the option to (semi-)automatically zoom for a better selection of content from remote providers.

4.3 Implications

We presented a solution for seamless task migration among a broad range of devices. Our approach emphasizes spontaneous and unplanned content access with minimal user input, while being very responsive. This interaction is based on an intuitive metaphor of video recording.

Our interface allows for *continuous interaction* - mobile device's display is updated in real-time and receives continuous feedback based on the content user is currently looking at. In every moment user is given relevant task and content-migration options for selected application and its content.

In order to reliably establish the homography between the screen and the observation of the mobile device's camera our system utilizes a combination of natural features based camera pose detection and virtual cursor tracking. This allows us to employ a fast and precise interaction even on low-end mobile devices, support migration of static and dynamic screen content and allow for simultaneous interaction of multiple users.

We created a prototype implementation of the whole solution which allows for task and content migration from web applications (through Google Chrome extension) and desktop applications (plugins for MS Office suite applications) to a mobile client (implemented for the Android platform).

This prototype was examined within a user study and by a set of performance evaluation experiments. The results indicate that it substantially outperforms the existing solutions: the localization and task migration is done in real time on a mid-level cellphone; the localization is reliable even for different observation angles and for cluttered screen content. Our solution operates on a video stream with all the benefits: if one camera frame fails for a reason, the mobile client program determines the location from a subsequent valid one.

5 Adapting Ex-Post Interaction with Public Display Content Using Eye Tracking

Short intensive interactions with unfamiliar pervasive displays coerce users to perform cognitive operations with uncertainty and risk of not being able to access the information of relevance later. We developed a new way of interaction with pervasive displays by harnessing the eye-tracking technology to extract information that are most likely relevant to the user. These extracted bits of important information are presented to the user and sorted according to their estimated importance – in the PeepList. The users can interact with the PeepList without explicit commands and they can access the customized PeepList ex-post in order to review information previously consumed from the pervasive display. What is more, by employing the gaze tracking we are able to eliminate a major drawback of our previous solutions - visible part of a marker field mixed with screen's content.

We carried out a user study involving 16 participants to evaluate the contribution of PeepList to efficient pervasive display interaction. The tests revealed that the PeepList system is unobtrusive, accurate, and in particular reduces interaction times by 40% when complex tasks were presented to participants. A feasible user model can be built in under 30 seconds in 50% of all interactions, and in one minute a majority of all interactions (70%) lead to a useful user model. Experimental results show that eye-tracking is a valuable real-time implicit source of

information about what the user is searching for on a pervasive display and that it can be used for real-time user interface adaptation. This considerably improves the efficiency of obtaining and retaining required data. This chapter is based on our published paper [14].

5.1 PeepList – Pervasive Interactive Display

PeepList is a novel interaction technique with pervasive displays and similar information devices (Figure 5.1). It provides personalized, implicit feedback based on user's gaze. PeepList's main objective is to help users quickly and efficiently interact with desired content based on multiple criteria under time constraints.

An example of this situation is food selection from a pervasive display board where a sheer number of types and combinations of food are presented to customers. Customers select food from a broad range of options while considering several criteria at the same time (e.g. type of food, price and value, nutritional contents, allergens, personal preference).

To demonstrate PeepList's generality we have developed also an additional scenario from a different domain – the car shop selector. In this scenario, user's task is to select one or more cars from a pervasive display based on their brand, price and available car features (e.g., rear spoiler, trunk release, privacy glass, leather seats). This task is based on actual car selection from a used car dealership/car rentals where these attributes are commonly used to navigate in a large pool of cars.

PeepList's main goal is to provide fast and accurate recommendation of items, while being unobtrusive. Such information can be exploited for further interactions, for example, to

Figure 5.1: The PeepList interaction overview. 1) User approaches pervasive display that is equipped with an eye-tracking technology. 2) While user examines the content of the pervasive display, the PeepList system (a part of the pervasive display) internally reorders relevance for on-screen elements based on user's eye tracking data. Reordered items are automatically sent to user's mobile device. 3) After the interaction, the user has access to personalized results and is able to further filter and reorder the results.

speed up interactions with the teller, improve further query generation, generate recommendations, and to simplify further choices. The user data collection utilized by the PeepList system is based on implicit eye tracking, where the eye tracker is embedded into the pervasive display. This type of eye tracking is generally considered to be unobtrusive [1, 9], especially when compared to mobile eye trackers or marker-based interaction with a smartphone.

5.1.1 PeepList Concept

When a user approaches the PeepList-enabled pervasive display (see Figure 5.1 for the overview of the whole concept), the system registers user's behavior and based on several features from the embedded gaze tracking (see Section 5.1.2 for details) starts ranking individual items (e.g. type of food – burgers, salads, shakes in the food scenario, or car brands in the car shop selector scenario) along with attributes associated with the items (e.g. price, nutritional values, or car features).

For each attribute a vector of gaze metrics is computed and data from all the attributes within one element are also accumulated (for details see Section 5.1.2). By analyzing this individual pattern of user's gaze behavior, the PeepList system creates a personalized and ranked list of items. This list is continuously updated (1 sec. in our implementation) and sent to user's mobile device.

This way the user is able to access personalized information from the pervasive display at any time and be recommended other relevant items. Also, the user can access the collected PeepList after departing from the pervasive display and deal with the list adapted by the inferred preference.

Because the list of options on a content provider is usually long and secondary interaction takes place on a mobile device, we have developed a minimalistic interface for the mobile client. The interface is based on a responsive layout of elements, which efficiently uses the screen space; filters are turned on and off by a single tap. If the system infers that certain attributes are relevant, corresponding filters are automatically turned on, thus filtering (and reducing) the list of results. On mobile platforms, this approach, we believe, is intuitive and unobtrusive, as the whole client-side interaction can be controlled with one hand and the user sees the results and the filters at the same time.

5.1.2 Gaze Data Processing and Relevancy Estimation

A pervasive display typically contains several dozens of items that can be of relevance to the user. We created bounding rectangular boxes around each item for simplicity, however, more complex shapes can be used. Each element has the same structure and it is further divided into areas containing detailed information about the element (hereafter attribute) – see Figure 5.2.

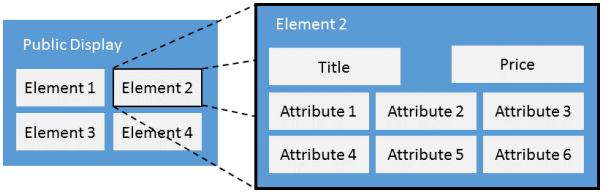

Figure 5.2: Pervasive display elements. Elements are currently rectangular and structured into attributes. Gaze tracking reasons which attributes are important to the user and uses this information for forming the PeepList.

As the user interacts with the display, for each attribute a vector of gaze metrics is computed (Figure 5.3). Data from all the attributes within one element are also accumulated. It is thus possible to determine which attributes in which element were attended by the user. When the participant's eyes could not be tracked, the system would not update the currently maintained rank of relevance. Using a series of piloting experiments, we derived the following list of gaze metrics. They are inspired by standard metrics used in the eye-tracking research [13, 12] and we experimented with all of them in order to determine, which

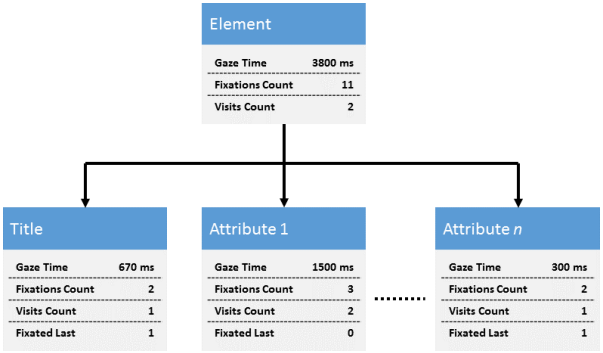

Figure 5.3: Collecting gaze data for Element and its Attributes. A vector of gaze features is created for the element as a whole and for the individual attributes.

ones would contribute most to our purpose.

Let us set the terminology for the following explanation. Each eye-tracking session is treated as a sequence of eye fixations $f_i, \forall i \in (1, \ldots, F)$, where F is the total number of fixations in the session. The duration (length) of one fixation will be denoted as $|f|$. Observable items $g \in G$ are both the elements $e \in E$ and attributes within elements $[e, a] \in E \times A$ (A is the set of attributes). According to the screen layout and based on the sufficient size of the observable items, the relation of fixations with observable items is defined: $\phi(f, g)$. This relation indicates that fixation f is related to observable item g (i.e. the user is attending to the item). It should be noted that one fixation f can be related to multiple items g at the same time. This typically means that when the subject observes an attribute, she also attends to its parent element (see Figure 5.2 and 5.3).

Gaze Time τ: Sum of all fixation durations on the item in mil-

liseconds. In eye-tracking research, gaze duration is considered as a measure of relevance [13].

$$\tau(g) = \sum_{\forall f_i : \phi(f_i, g)} |f_i| \tag{5.1}$$

Fixations Count σ: Total number of fixations on the item. Previous research shows that the number of fixations is negatively correlated with search efficiency [13] and semantic importance [12]. We define this value by the means of the cardinality of the proper set of fixations $\|\{\cdots\}\|$.

$$\sigma(g) = \|\{f_i | \phi(f_i, g)\}\| \tag{5.2}$$

Visits Counts ν: Total number of *visits* of the item. A *visit* is either a single fixation or multiple fixations connected by saccades belonging to a single observable element. A *visit* is ended by a fixation on a different observable element than the previous fixation. The ending of a visit is similar to *OnMouseExit* type of event commonly used in traditional interfaces.

$$\nu(g) = \|\{f_i | \phi(f_i, g) \wedge \neg\phi(f_{i+1}, g)\}\| \tag{5.3}$$

Fixated Last φ: Total number of cases, when the item was fixated as the last item within its parent (sudden saccade out of the element after the visual search of its attribute). We have chosen "*Fixated Last*" instead of a more commonly used "*Fixated First*" because we have frequently observed that in cases when users are looking for a certain attribute value in a randomly ordered list of attributes within an element, the information that this attribute was visited last very often indicates that user found what was relevant.

$$\varphi(a) = \|\{f_i | \exists e \in E : \phi(f_i, [e, a]) \wedge \neg\phi(f_{i+1}, e)\}\|, \tag{5.4}$$

$$\varphi(e) = 0, \quad \forall e \in E, a \in A \tag{5.5}$$

The score of each item (element or attribute) is then:

$$s(g) = \alpha_1 \tau(g) + \alpha_2 \varphi(g) + \alpha_3 \sigma(g) + \alpha_4 \nu(g) \tag{5.6}$$

We estimated the parameters from the piloting data. We first obtained coarse settings of the coefficients to find feasible ranges by trial and error. Later, we fine tuned the settings by linear regression to estimate the influence of individual metrics on the final raking of relevance by inferring the values for α_i constants in Equation (5.6). The most contributing metrics for a stable ranking of relevance are *gaze time* τ and *fixated last* φ. The other two metrics (*visits counts* ν and *fixations count* σ) have negligible influence on the overall score $s(g)$ and in the experiments reported below they are omitted for simplicity. An interesting finding is that the rarely used metric *fixated last* is very important in this measurement. However, it is understandable, because when user after visiting an attribute moves on to another element, it strongly indicates the importance of the given attribute. All gaze-based metrics were recorded to an experimental database that is made public with this paper to allow further data mining from a research community.

Based on the score, the list of elements and attributes is continuously reordered, prioritizing items with highest score. After each computation, the list is sent to the client device. It is the client's responsibility to render the list in the most appropriate way. In our implementation, which is targeting mobile devices, instead of showing the whole ordered list, the list is filtered. Each filter represents one element or attribute present in the scenario.

The filters were ordered in the same way as attributes. Their order was based on the value of Normalized discounted cumulative gain (NDCG) [23]. The NDCG is a metric commonly used to measure the performance of a recommendation system. The filters were ordered from the highest NDCG value to lowest.

In our client implementation three filters with the highest score were turned on for the client interaction part of the session.

5.2 Experimental Evaluation

To evaluate the efficacy of PeepList for interaction with pervasive displays, we conducted a user study. We hypothesized that when the system is capable to estimate items of relevance of users on a primary task and transfer it into the mobile device, performance and user experience with the secondary display improve. In particular, we measured the accuracy of relevance prediction across queries with varying complexity, and the time spent on the mobile user interface as a function of accuracy.

5.2.1 Participants and Apparatus

The study we conducted involved 16 participants (mean age = 30.25, SD=5.91), of which 6 were female. All participants had normal or corrected-to-normal vision. All had a technical background with at least a first degree in natural sciences or engineering. They used computers frequently and daily and were skilled in interacting with computers and touch-enabled devices. Participants took part in the experiment individually, and assisting personnel was present at the experiment throughout its whole duration.

The hardware setup consisted of a Tobii TX300 eye tracker with 23 inch LCD display with a resolution of 1920×1080 pixels. The sampling rate was set to 300 Hz. Participants were seated at the distance of 60 cm from the screen. For the secondary interaction, an iPad Air 2 tablet was used, running a Chrome browser.

5.2.2 Experiment Design, Task and Procedure

Participants were first introduced to the experiment and in about ten minutes completed a warm-up session. During the warm-up, they could interact with the interfaces and ask questions to make sure they fully understood the task at hand. They were presented an example task consisting of finding a car with two given attributes. They also practised working with the mobile user interface. At the end of the warm-up session, participants performed a calibration routine for the eye-tracker. Participants remained naive about the purpose of the study.

Each session task was broken down to three parts. First, a query was presented to the participant, as a dialog box on the screen and on a sheet of paper that was available throughout the whole study. Next, the participant interacted with the experimental pervasive display interface to gather necessary information for answering the query. The interaction was limited to two minutes and an audio reminder informed the participants when 60, 30 and 10 seconds remained till the end of each trial. If the participant wished, the current query could have been displayed by pressing a defined key. Data were not collected during the query display and the 2 minutes countdown was paused.

The display interface was preloaded on a large computer screen equipped with the eye tracker. Computer mouse cursor was hidden, but participants could use the scroll-wheel to browse through the options.

The study was designed as within-subject experiment. A database of queries was created before the experiments, such that each query had at least 1 and at most 4 parameters. We balanced the number of queries to represent the varying complexity. The queries given to the participants were randomly sampled without repetition from a database of queries. In the experiment, first three filters on the mobile user interface were

pre-set depending on the ranked list of relevance, while other filters were deselected.

Because answering each query required processing of a high number of options, and for each new query the content of the screen was regenerated, it was virtually impossible to memorise the list of options. Participants wrote down the answers onto a sheet of paper immediately, and the response was recorded. Then, participants were further asked an additional question concerning the resulting options, which motivated them to use the second interactive list. The second list was presented on a mobile device. After discerning the answer, participants recorded it to the same sheet of paper.

The primary independent variable and a within-subject factor was the complexity of a query. This was measured as the number of attributes it contained. By modifying the complexity of a query we were able to evaluate the benefits of gaze-based relevance estimation for simple queries (1 or 2 attributes), and complex queries (3 or 4 attributes).

Dependent variables included the times on the primary display, the time on the mobile display, and the normalized discounted cumulative gain (NDCG) for each trial. The NDCG was employed also as the post-hoc factor: we were aware that not all participants' gaze can be reliably tracked (typically about 10 % of data) and thus the estimation of the relevance does not yield acceptable (i.e. above chance) levels.

When tracking did not work, the system would not update the currently maintained rank of relevance, and the resulting list was randomly ordered. For each interaction in this experiment, the outcome ranking can turn out to be relevant (high NDCG) or not (below chance levels derived empirically [23]). In a majority of these cases, these data points resulted from poor tracking quality. It is near to impossible that a poor-quality tracking would lead to a well-ordered list, because 1) the data and screens were always randomized and in a situation that

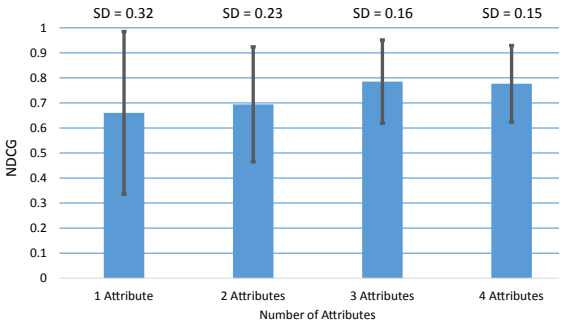

Figure 5.4: Performance of PeepList sorting (NDCG) as it depends on the number of element's attributes. The more complex the task (high number of attributes), the more accurate and thus helpful is PeepList's sorting.

gaze-tracking did not work the resulting ranking would again be random, 2) to confirm this, we carefully analyzed the reasons for low NDCG, and 3) when the participant's eyes could not be tracked, the system would not update the currently maintained rank of relevance. Therefore, the low-NDCG samples resulting from poor eye tracking share the same order of items with the original content provider display.

5.3 Results

In total, the participants solved 144 queries, and 130 queries were accepted into the sample. We discarded about $\sim$ 10% of recorded sessions due to the problems with data logging during the interaction session (web-service related problems mostly in the early stages of recordings); the result was that we did not

have complete data about the length of the mobile interaction session. To avoid any questionable data we decided to discard these sessions. Out of the 130 accepted queries, 31 queries had 1 relevant attribute, 35 queries had 2 attributes, 34 had 3 attributes and 30 queries had 4 relevant attributes. Median NDCG across all 130 queries was 0.765 (SD = 0.233) which is well above the chance level of 0.44.

To evaluate how the relevancy estimation deals with increasing complexity of the task (i.e. the number of attributes per element), we measured the mean value of NDCG for different numbers of attributes. Figure 5.4 summarizes the results. The system performs significantly better with queries containing three or four attributes than for queries with lower number of attributes, according to a one-way ANOVA $F(1, 128) = 4.29, p = .04$. The reason for this is that simpler queries with one attribute are solved by participants in a shorter time, not allowing the system to create a sufficiently accurate user model. Interactions requiring higher number of attributes take longer time, which leads to a better user model. This is also supported by the decreasing variance of NDCG with the increasing number of attributes. In 77 % of all cases at least one attribute was correctly inferred to belong to top 5, and in 27 % of all interactions all attributes were ranked in the very top positions in the inferred list of relevancy.

The average time the participants spent in the main content provider application was 64 seconds. The average time the participants spent in the mobile application was 30 seconds, matching with our expectation that interaction on the mobile device should be shorter, due to the filtered list of options. Table 5.1 shows the mean times of the interactions with the pervasive display and the mobile client, as function of NDCG distribution. With three levels of NDCG, the high NDCG values lead to a shorter interaction time (19 %) on the mobile device when compared to low levels. This improvement differs considerably

Table 5.1: Mean interaction time (minutes, seconds) participants spent with the pervasive display and with the client PeepList application. The trials are split evenly by the NDCG. Better (i.e. higher) NDCG considerably helps the users by speeding up their secondary interaction.

	NDCG Level		
	0.23 – 0.59	**0.60 – 0.84**	**0.85 – 1.0**
Pervasive Display	01:08.8	01:13.5	01:07.4
Mobile Client	00:34.9	00:31.1	00:25.3

with the complexity of the assigned task (up to 38 %); a one-way ANOVA showed a significant effect of the NDCG level on the interaction time with the mobile task ($F(2, 127) = 6.50$, $p = .002$).

We also examined how the duration of the secondary interaction time is influenced by the number of attributes in the query. We observed a reduction of time participants needed to complete complex queries with a larger number of attributes – if the NDCG was low, on average, the participants required 38 seconds to complete the secondary task. With a high NDCG, this time was reduced to 24 seconds ($\sim$ 37 % reduction). With simpler queries, the time was reduced from 31 seconds to 27 seconds ($\sim$ 13 % reduction), therefore, the relevance based filtering is more efficient for more complex queries (See Figure 5.5). When the mobile interface presents an incorrectly selected filter (due to a lower NDCG), the error can be fixed by the user almost instantly by a simple tap on the filter's name. It turns out that PeepList is especially suitable for non-trivial tasks, where it offers considerable speedup. The functionality of automatically keeping the items of relevance is valid for queries of all complexities.

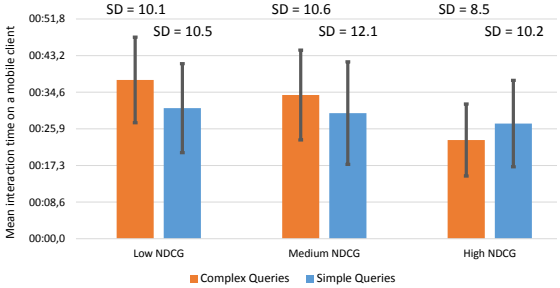

Figure 5.5: Mean interaction time on the mobile client for three levels of NDCG (as in Table 5.1 right) and for two levels of query complexity. When NDCG is higher, the interaction time is trimmed down considerably.

5.4 Discussion and Implications

This chapter presents PeepList, a novel technique for interacting with pervasive display equipped with an eye tracker, and its user evaluation. Eye tracking is used to infer short-time relevance when interacting with the display. These items of relevance are used to pre-select display elements and sort them, so that they form the PeepList – a user-centric collection of task-relevant information for later use. This paper expands the knowledge about modeling of implicit user preferences through gaze data.

Eye tracking technology is developing quickly [12] and will soon be embedded into everyday digital technologies [22]. Although there is not a commercially available eye-tracking technology and associated processing at the moment that would instantly enable massive use of PeepList-like user streams, such class of eye tracking technology can be expected in the near

future. Implementation and evaluation of technologies such as the PeepList demonstrates the potentials of pervasive eye-tracking and fuels its further research.

Our experiments were formed around real-life scenarios which are commonly experienced throughout the day, such as food selection or interaction with advertisement displays. The experiments show that with increasing task complexity, the PeepList becomes increasingly helpful; the PeepList significantly speeds up the users' ex-post interaction by almost 40 % and was generally well appreciated. Although the evaluated scenarios are simple, they reach the complexity and sophistication found elsewhere (e.g. navigation assistance and guidance across complex public spaces [20] or direct personalization of content [7]). An interesting finding is the importance of the *fixated last* φ gaze metric. It contributes by valuable information to the raking of the elements, because this terminal fixation is a powerful indicator of the fact that the wanted information was found.

The interactions presented here were short, and users' preferences changed rapidly. Despite these challenges, the experiments confirmed that gaze tracking can provide valuable cues about the user's items of relevance in this domain. Electronic pervasive interactive technology can assist their users even more efficiently when they have access to this type of information and eye tracking turned out to be an unobtrusive means of obtaining it.

6 Closing Remarks

The coupling of personal devices and situated displays enables to combine personalized interactivity of a private device with presentation space of a large, publicly available display. Only with appropriate design, and by using technologies and interaction metaphors that parallel the way the user thinks about a task as closely as possible, are we able to achieve intuitive content sharing and information reaccess. The traditional desktop metaphor commonly seen in contemporary interactions is more and more flawed and unable to satisfy the new user interaction and user experience requirements. These requirements are originating from the necessity to utilize many heterogeneous devices in distributed computing environment. Enabling this type of interaction, all in the context of Weiser's vision of "Calm Technology", was the goal of our work.

We introduced our interaction technique based on Uniform Marker Fields – two-dimensional planar grid of squares that can be easily and robustly recognized in camera image even when occluded, blurred or observed under poor lighting condition. By utilizing optical localization we have built a system which allows to identify screen region and transfer not only pixels observed by camera, but displayed content to the personal device. We provided an empirical evaluation of this task migration design by evaluating reliability of detection and computational demands. Besides that, a user testing was carried out to determine which way of inserting/blending the marker field into the screen image is best accepted by the users and what level of obtrusiveness they are sensitive to.

This approach to content transfer and information reaccess was later enhanced by introducing the video recording metaphor. Instead of discrete selection our interface allows for continuous interaction – mobile device's display is updated in real-time and receives continuous feedback. In every moment, user is given relevant task and content-migration options for selected application. Our approach thus emphasizes spontaneous and unplanned content access with minimal user input, while being very responsive. The experimental results show that our solution provides reliable task migration at interactive frame rates. This substantially outperforms the existing solutions. We carried out a user study as well as empirical evaluation tests. The results indicate that the system is perceived as intuitive, easy to learn and effective in transferring ongoing tasks between the desktop/kiosk and a mobile device.

Our latest solution – PeepList solves the major drawback of previous approaches – observable marker field mixed with screen's content while being able to build user-model during the interaction. We developed a new way of interaction with pervasive displays by harnessing the eye-tracking technology to extract information that are most likely relevant to the user. The users can interact with the PeepList without explicit commands and they can access the customized PeepList ex-post in order to review information previously consumed from the pervasive display. We carried out a user study involving 16 participants to evaluate the contribution of PeepList to efficient pervasive display interaction. The tests revealed that the PeepList system is unobtrusive, accurate, and in particular reduces interaction times by 40% when complex tasks were presented to participants. A feasible user model can be built in under 30 seconds in 50% of all interactions, and in one minute a majority of all interactions (70%) lead to a useful user model.

Bibliography

[1] Researching mobile learning: Frameworks, tools and research designs, May 2009.

[2] Elizabeth Bales, Timothy Sohn, and Vidya Setlur. Planning, apps, and the high-end smartphone: exploring the landscape of modern cross-device reaccess. In *Proceedings of the 9th international conference on Pervasive computing*, Pervasive'11, pages 1–18, Berlin, Heidelberg, 2011. Springer-Verlag.

[3] E. Bardram. Activity-based computing: support for mobility and collaboration in ubiquitous computing. *Personal Ubiquitous Comput.*, 9(5):312–322, September 2005.

[4] Dominikus Baur, Sebastian Boring, and Steven Feiner. Virtual projection: exploring optical projection as a metaphor for multi-device interaction. In *Annual Conference on Human Factors in Computing Systems*, 2012.

[5] Tsung-Hsiang Chang and Yang Li. Deep shot: a framework for migrating tasks across devices using mobile phone cameras. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2163–2172, New York, NY, USA, 2011. ACM.

[6] Karen Church and Barry Smyth. Understanding mobile information needs. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, MobileHCI '08, pages 493–494, New York, NY, USA, 2008. ACM.

[7] Nigel Davies, Marc Langheinrich, Sarah Clinch, Ivan Elhart, Adrian Friday, Thomas Kubitza, and Bholanathsingh Surajbali. Personalisation and privacy in future pervasive display networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 2357–2366, New York, NY, USA, 2014. ACM.

[8] David Dearman and Jeffery S. Pierce. It's on my other computer!: computing with multiple devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 767–776, New York, NY, USA, 2008. ACM.

[9] M. Farid, F. Murtagh, and J.L. Starck. Computer display control and interaction using eye-gaze. *Journal of the Society for Information Display*, 2002:289–293, 2002.

[10] Mike Harding, Oliver Storz, Nigel Davies, and Adrian Friday. Planning ahead: techniques for simplifying mobile service use. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, HotMobile '09, pages 13:1–13:6, New York, NY, USA, 2009. ACM.

[11] Ken Hinckley, R Jacob, and Colin Ware. Input/output devices and interaction techniques, 2004.

[12] Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost Van de Weijer. *Eye tracking: A comprehensive guide to methods and measures*. Oxford University Press, 2011.

[13] Robert Jacob. Eye tracking in human-computer interaction and usability research: ready to deliver the promises (section commentary). *The mind's eye: cognitive and applied aspects of eye movement research*, pages 573–605, 2003.

[14] Rudolf Kajan, Adam Herout, Roman Bednarik, and Filip Povolny. Peeplist: Adapting ex-post interaction with pervasive display content using eye tracking. *Pervasive and Mobile Computing*, 2015.

[15] Rudolf Kajan, István Szentandrási, Adam Herout, and Alena Pavelková. Video recording - a promising metaphor for inter-device task migration. In *Journal of WSCG*, pages 95–103. University of West Bohemia in Pilsen, 2014.

[16] Rudolf Kajan, István Szentandrási, Adam Herout, and Michal Zachariáš. On-screen marker fields for reliable screen-to-screen task migration. In *Proceedings of the 2013 International Conference on Human Factors in Computing & Informatics*, pages 692–710. Springer Verlag, 2013.

[17] Amy K. Karlson, Shamsi T. Iqbal, Brian Meyers, Gonzalo Ramos, Kathy Lee, and John C. Tang. Mobile taskflow in context: a screenshot study of smartphone usage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2009–2018, New York, NY, USA, 2010. ACM.

[18] I. Scott MacKenzie, Tatu Kauppinen, and Miika Silfverberg. Accuracy measures for evaluating computer pointing devices. In *CHI*, 2001.

[19] Tim Paek, Maneesh Agrawala, Sumit Basu, Steve Drucker, Trausti Kristjansson, Ron Logan, Kentaro Toyama, and Andy Wilson. Toward universal mobile interaction for shared displays. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 266–269, New York, NY, USA, 2004. ACM.

[20] Luigi Palopoli, Antonis Argyros, Josef Birchbauer, Alessio Colombo, Daniele Fontanelli, Axel Legay, Andrea Garulli, Antonello Giannitrapani, David Macii, Federico Moro, Payam Nazemzadeh, Pashalis Padeleris, Roberto Passerone, Georg Poier, Domenico Prattichizzo, Tizar Rizano, Luca Rizzon, Stefano Scheggi, and Sean Sedwards. Navigation assistance and guidance of older adults across complex public spaces: the dali approach. *Intelligent Service Robotics*, 8(2):77–92, 2015.

[21] Umar Rashid. *Cross-display attention switching in mobile interaction with large displays*. PhD thesis, University of St Andrews, 2012.

[22] Hana Vrzakova and Roman Bednarik. Eyecloud: Cloud computing for pervasive eye-tracking. *Proc. PETMEI 2013*, 2013.

[23] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. A theoretical analysis of NDCG ranking measures. *Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013)*, 2013.

[24] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, July 1999.