A Scalable Framework For Markerless Camera-based Smartphone Interaction with Large Public Displays

Matthias Baldauf, Peter Fröhlich, and Katrin Lasinger
FTW Telecommunications Research Center Vienna
Donau-City-Strasse 1, Vienna, Austria
+43 1 5052830-47
{baldauf, froehlich, lasinger}@ftw.at

ABSTRACT
The camera-based control of large markerless displays through smartphones in real-time is an appealing novel interaction technique in urban environments and opens a wide range of new use cases. While recent research has focused on investigating respective user aspects using simplified lab prototypes, this paper addresses the question how such services can be technically enabled for a large network of screens. We present our efforts towards a suitable distributed framework for enabling such smartphone-based screen services at a large scale and introduce a prototypical implementation. To illustrate its features we present two concrete interactive services we realized on top of the framework.

Categories and Subject Descriptors
D.2.11 [**Software Engineering**]: Software Architectures – *Patterns;* H.5.2 [**Information interfaces and presentation**]: User Interfaces – *Input devices and strategies; Interaction styles*

General Terms
Design, Experimentation, Human Factors

Keywords
Distributed system, mobile interaction, large screens, remote control, mobile augmented reality

1. INTRODUCTION
Large public screens have become ubiquitous in urban surroundings over the last few years. At stations they show real-time departure times, in malls they act as destination board and entertainment guide and in shopping windows they inform us about latest bargain offers. While traditional flat screens provide no means of interaction for interested passers-by, modern feature-rich smartphones have been identified as promising input devices for advanced interactive services for remote public screens [2].

Recently, especially vision-based approaches utilizing the mobile camera and respective methods of computer vision are increasingly gaining attention in academia and industry. Advances in mobile computing technology and computer vision enabled the intuitive touch-based interaction with unmodified screen content over the viewfinder of the mobile device in real-time (Figure 1).

PerDis 2012, June 04 - 05 2012, Porto, Portugal

While researchers in the domain of Human-Computer Interaction (HCI) started to investigate and proof the attractiveness of this novel interaction technique using lab prototypes, this work is concerned with the design of a suitable framework for enabling such markerless camera-based services at a large scale (i.e. for a huge number of screens and involved mobile devices). Such a platform has to avoid central performance bottlenecks to ensure scalability and must provide the following basic functionality:

Registration. As a first step for enabling the interaction with a specific screen, the platform needs to offer a mechanism to identify the screen a smartphone is currently targeted at. Thus it must allow the smartphone to register itself at the desired screen, i.e. to establish a wireless connection to it for communicating input commands.

Synchronization. After a successful registration, the smartphone and the display need to periodically exchange image information in order to determine the pose of the mobile device in relation to the screen. Aware of this transformation, touches on the mobile display can be mapped to corresponding screen positions despite the dynamic screen content.

In the remainder of this paper, we present out efforts towards a suitable interaction framework focusing on the large-scale deployment of such interactive screen services. First, we summarize some related work in the field of camera-based interaction in Section 2. Section 3 presents an overview of the proposed system architecture and Section 4 gives implementation details of our prototypical realization. In Section 5 we introduce two showcase services that we implemented based on the presented platform. Finally, we draw some concluding remarks and outline future work in Section 6.

2. CAMERA-BASED INTERACTION
Early work in the field of camera-based handheld interaction with screens relies on obtrusive visual markers superimposed onto the actual application. E.g., the system by Madhavapeddy et al. [9] makes use of so-called 'spot codes' augmenting the current screen content. Photographing these static markers allows for example to select specific screen areas (e.g., to zoom in an exemplary map application). Herbert et al. [7] demonstrated how smartphones can be used to visually interact with screens in real-time using distinctively colored reference targets which are dynamically overlaid onto the actual screen content and can be easily localized on the mobile's camera video stream.

In the meanwhile, markerless vision-based interaction approaches became technically feasible even on mobile devices. The cornerstone for markerless object recognition and visual tracking is the extraction and comparison of so-called natural image features, specific image structures which can be used to identify a

Figure 1: Interacting with a distant public screen via the viewfinder of the mobile device.

known template within an observed scene. Popular algorithms to detect and describe such features include SIFT by Lowe et al. [8] and SURF by Bay et al. [3]. Both algorithms are robust to scaling and rotating, yet their execution is computationally expensive. However, researchers in the field of mobile Augmented Reality (AR) made tremendous advances in optimizing and adapting such algorithms and achieved interactive frame rates on mobile phones (cf. [14]).

The markerless interaction with a screen through the live video of a mobile device in real-time was first investigated by Boring et al. [4]. The researchers introduced a system called 'TouchProjector' which forwards touch events occurred on the mobile display to a distant screen targeted through the viewfinder and compared different design alternatives for improving the efficiency of this novel interaction technique. Baldauf et al. [1] presented a related system for touch-enabling remote screens by utilizing natural feature descriptors. Another related system was introduced by Herbert et al. [7]. However, their low-fi prototype was only built for a usability evaluation and was based on a webcam instead of a real mobile device.

In preliminary systems with dedicated visual markers the solving of the registration task was straightforward: the marker usually not only encoded an action or position identifier but also the actual address of the computer (cf. [13]). However, in recently introduced systems enabling markerless live video interaction, an efficient registration is harder to achieve. Most of the recent live video prototypes presented so far acted as proof-of-concept demonstrators or were developed for evaluating the user acceptance of such novel camera-based services, thus the registration step and the technical requirements for a large-scale deployment have been neglected. For example, 'TouchProjector' [4] makes use of a centralized 'environment manager' which continuously receives updates of the content shown on the involved screens. Further, it is responsible for the execution the computer vision tasks since the mobile devices only stream their video over WiFi to this server. This setup works well for prototyping purposes in a lab environment. However, with an increasing number of connected mobile devices and screens showing dynamic content and thus constantly updating the centralized manager, such a centralized approach must fail. The system by Baldauf et al. [1] introduces a first decentralized approach where mobile devices directly connect to the computers hosting the screens. However, their registration method based on a Bluetooth scan is slow and cumbersome needing purpose-given Bluetooth device names and user input to work. Other scan-based approaches for interacting with screens (e.g. [5]) assume that 'the connection is established when entering the room'. Obviously, such approaches based on proximity are not precise enough and thus not suitable for multi-screen environments.

3. SYSTEM ARCHITECTURE

Our goal is the creation of a scalable framework which enables the markerless vision-based registration and interaction with public screens at a large scale. As mentioned above, existing lab prototypes assuming that one central instance is always aware of the current content of each screen as well as executing image matching processes for the connected mobile clients are obviously not feasible for large-scale deployment with many mobile users and lots of screens continuously forwarding image information to the central server.

While investigating a suitable architecture avoiding central performance bottlenecks we were inspired by successful Peer-to-Peer (P2P) applications, highly distributed systems smartly sharing workload among peers. One of the most popular P2P applications, the file sharing platform Napster [6], was based on a so-called centralized P2P architecture: While the actual content was directly transferred between the peers, a central server was responsible for search queries. These were based on file lists provided by each peer and thus did not require much bandwidth for search.

Adopting some aspects of a centralized P2P system, we designed the architecture depicted in Figure 2. The overall framework consists of three different components: *Screen Clients*, a *Mediator*, and the mobile devices.

So-called *Screen Clients* are executed on the computers hosting the public screens. Figure 2 shows four exemplary deployed Screen Clients, three running at traditional flat screens, one at public projection.

The *Mediator* keeps track of the available service-enabled screens, i.e. the running Screen Clients. To register a new Screen Client at the Mediator, its data may either be manually entered using a Web form or stored in a corresponding local configuration file allowing the Screen Client to register itself at the Mediator during startup. The required dataset for each Screen Client includes its IP address, a textual name as identifier, the location of the hosted screen in WGS84 coordinates as well as its approximate orientation in degrees (i.e. the direction the screen surface is facing). Once, a Screen Client is listed at the Mediator, the application starts its work and periodically checks the connected screen for updates and stores the most current video frame for later processing and provisioning.

Finally, smartphones act as end user devices for interacting with the involved public screens. While these mobile clients make use of the Mediator for the registration task, the actual image and control information is exchanged directly between the mobile devices and the Screen Clients.

3.1 Registration

The goal of a successful registration is to provide the (IP) address of the screen targeted through the viewfinder of the smartphone. In order to avoid a performance bottleneck and allow for large scale deployment, the presented framework follows a distributed registration approach.

A mobile device initiates a registration by submitting a suitable video frame captured by the camera to the mediator (Figure 2, step 1) whose address is well-known. Instead of steadily collecting image information from all available Screen Clients,

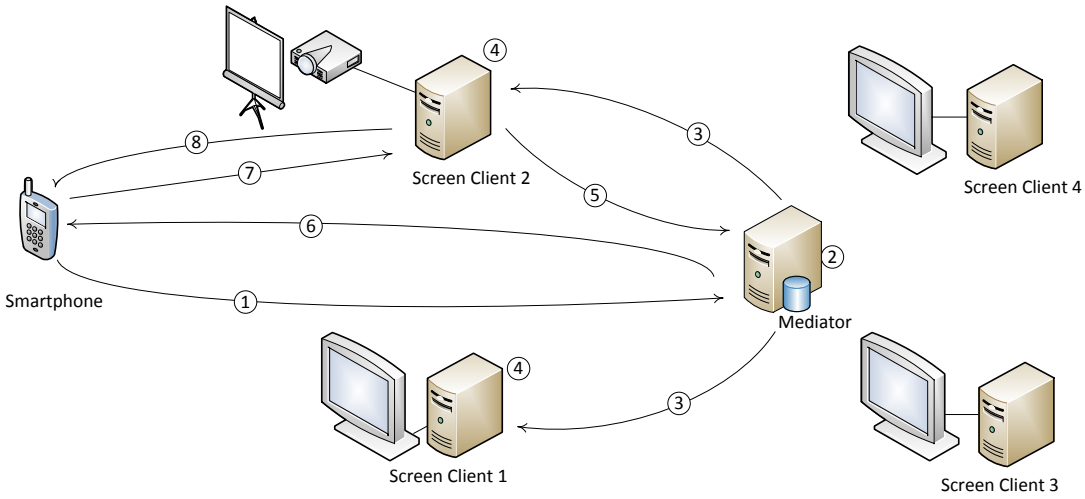

Figure 2: Simplified overview of the system architecture. The framework consists of so-called Screen Clients installed at the computers hosting the screens, a Mediator, and the smartphones as end user devices.

the Mediator is able to forward registration requests to Screen Clients. However, it does not contact all screen clients registered in its database but makes use of the additionally submitted context information to limit the set of potential candidates (step 2). With each registration request the mobile device includes its current location with an inaccuracy range (dependent on the available localization method such as GPS-based or network-based) as well as its orientation determined by the built-in compass. Based on this information, the mediator selects only screens in the respective area and with a suitable orientation for forwarding the video frame (step 3).

When a Screen Client receives a registration request in form of a mobile video frame, it tries to match the image with its screen content (step 4). Only if a mapping can be achieved, the Screen Client replies with a corresponding response message (step 5). As soon as the mediator is informed about a successful match it forwards the IP address of the respective Screen Client to the requesting mobile device (step 6). If no Screen Client replies within a specified timeout interval, the Mediator returns an error message.

3.2 Synchronization

Once, the registration is successfully completed and the mobile device is aware of the address of the targeted screen, it may establish a direct network connection to the screen computer (step 7) to exchange image information and input commands. In general, the synchronization of the camera video stream and the content of the screen are possible in two ways:

- Informing the Screen Client about the camera video stream of the mobile device and performing the image matching tasks at the Screen Client
- Informing the mobile device about the video stream of the Screen Client and performing the image matching tasks at the mobile device.

To keep bandwidth low and increase scalability, it is reasonable to transfer the video which is more stable and thus requires less updates. We decided to send the screen content to the mobile device since the mobile camera obviously suffers from more movement. Since the necessary computer vision tasks are nowadays also feasible on mobile devices, this approach has another advantage: image matching can be 'outsourced' to the mobiles and performed in a distributed manner which does not burden the Screen Clients. Further, the approach of notifying the mobile device about the currently visible screen content is more flexible and does not limit the set of potential use cases for the framework.

To enable this distributed image matching approach, the respective Screen Client manages a list of connected mobile devices and periodically updates them with scaled screenshots to allow for device-initiated screen tracking. Being aware of the currently shown content of the screen, the mobile device is able to detect its appearance in the camera video stream and thus to calculate its current pose and the transformation for mapping touches on the mobile display to corresponding screen position as seen through viewfinder.

4. PROTOTYPE IMPLEMENTATION

In order to experiment with this architecture in practice and realize some concrete services on top of it, we implemented a fully functional prototype of the system. The involved desktop applications, the Screen Client and the Mediator, are written in C# using Microsoft's .net framework, our mobile application is implemented for Android devices. Further, we make use of OpenCV [10], a popular open source library for computer vision tasks, for executing various image related operations.

4.1 Screen Client

Since the main task of the Screen Client is the capturing and provisioning of the current screen content, we use a background thread to continuously create screenshots (in one second intervals in the current configuration) utilizing Windows GDI (Graphics Device Interface) calls. Each screenshot is scaled down to a size of approximately 400x225 pixels (dependent on the original screen size) to speed up the following operations. On this frame and its predecessor we then perform an image subtraction, a common process to detect changes between two images: for each pixel the color values are subtracted resulting in black image in case of two identical images. To be robust against minor updates such as a blinking cursor or a small animated web banner we apply a certain threshold for the number of non-black pixels. In case an update is detected, a 50% JPG compression is applied to

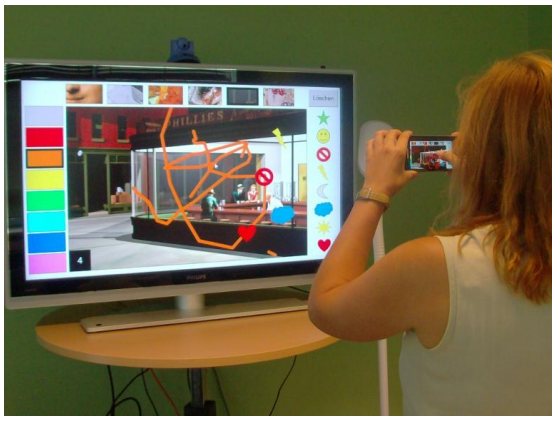

Figure 3: Remotely controlling a painting application through the viewfinder of the mobile device.

Figure 4: Sharing media files between the smartphone and the screen using an extended drag'n'drop operation.

the scaled screen image to significantly reduce the image size. Then it is sent to the connected mobile devices.

Via plain TCP sockets a Screen Client provides the following services:

Synchronization. A mobile device may ask to be notified about screen updates and receive screen frames in form of JPG encoded images.

Matching Request. The Mediator may use this service to forward a mobile camera frame to this Screen Client and trigger a matching attempt. The Screen Client tries to match the received image with the current screenshot and a certain number of previous frames using SURF features and the FLANN algorithm [10]. If a transformation in form of a homography matrix can be recovered from the matching image, we assume to have a match and asynchronously return a respective result code to the Mediator.

Remote Control. In order to enable interactive services, a mobile device may submit a screen position in pixel coordinates and a mouse button status (down, up, click) to trigger the respective mouse action at the remote screen. To allow for multi-user applications, receiving remote control commands from several connected mobile devices is supported.

Content Transfer. A mobile device may start the transfer of a local file to the screen client as well as fetch information about a file hosted at the screen client content and initiate its transfer to the mobile device.

4.2 Mediator

The centralized Mediator is responsible for distributing registration requests. We make use of MSSQL database to store the information about available screens including e.g. the IP address of the respective Screen Client and the location and orientation of the screen. When a new screen is added to the database, we calculate the corresponding metric Cartesian coordinates (Gauss-Krüger coordinates in our current setup) from the passed WGS84 coordinates to simplify range-based queries.

The Mediator offers two core services, again over plain TCP sockets:

Registration Request. When a mobile device submits a video frame and its location and orientation via this service, the Mediator forwards the image to the respective Screen Clients. Since server sockets are often blocked by mobile network operators due to security constraints, this original TCP connection initiated by the mobile device is kept open and used for returning the result: The address of the first replying Screen Client is replied to the mobile requester. In case of no reply within a time frame of currently five seconds, the Mediator returns an error message.

Matching Response. At this service, the Mediator listens for incoming asynchronous replies by Screen Clients reporting about a successful match with their current screen content.

Further, the Mediator offers two basic HTTP services for adding and removing screens from its database.

4.3 Mobile toolkit

To facilitate the communication with the services provided by the Mediator and the Screen Client and thus to ease the creation of new mobile applications, we created a corresponding mobile toolkit for Android. The features provided by this library include methods for passing a camera frame to the mediator, to transparently connect to a known Screen Client, and to conveniently trigger mouse actions on the remote screen. Further it features convenient mechanisms for receiving screen images from a Screen Client using an observer pattern.

We make use of the Android NDK (Native Development Kit) to integrate OpenCV on the Android platform. For matching the camera images with the received screenshots and thus visually recognize the screen, we combine SURF feature detection with an optical flow tracker (cf. [14]). Similar to the image matching at the Screen Client during the registration, we apply a FLANN-based matching to determine the homography matrix for later position transformations.

5. DEMO APPLICATIONS

To showcase and validate the features of the presented framework, we implemented two exemplary demo applications. Both are currently single-user applications, i.e. a connection or interaction attempt by a second user is prevented when the application is already in use. The first demo application realizes a remote control over the camera viewfinder, the second one demonstrates a novel approach for sharing media files between a mobile device and a remote screen.

5.1 Remote control

The basic use case for the presented framework is an ‘AR remote control’. While targeting the remote screen through the camera viewfinder, touches on the mobile display can be mapped to

corresponding screen positions and forwarded to the screen where the respective mouse action is triggered. To show the possibilities we implemented a demo application that combines typical mouse operations such as clicking, dragging and tracing: the drawing program depicted in Figure 3 allows creating an art collage by selecting a pen color, choosing a background painting, dragging stencils and drawing onto the painting.

5.2 Content transfer

A more sophisticated use case example realized on top of the framework is this novel approach for transferring media files. Using the platform services we extended the well-known drag'n'drop action to share content between a mobile device and a remote screen. Figure 4 shows our 'black board' demo application in action. We augmented the mobile application with the full-screen viewfinder by a simple file browser (slid in from the left side) listing locally stored documents and photos. In this way, file icons can be dragged between two different reference systems: the traditional 2D mobile display and the real world in form of a public screen viewed through viewfinder.

When an icon is dragged from the file browser onto the viewfinder showing the remote screen, a placeholder is shown at the corresponding position on the remote screen that also moves according to the dragging on the mobile display. As soon as the user drops the icon (i.e. raises the finger), the file is actually transferred and a suitable preview image is displayed instead of the placeholder. Vice versa, a file symbolized by such a preview image on the screen can also be dragged back to the smartphone. In Figure 4, the user drags a photo of a sunflower (which was copied to the black board before by another user) from the screen to the local file gallery. Once, an item visible on the remote screen is long-pressed, a suitable overlay icon appears on the mobile to indicate the transfer possibility. When this icon is dropped over the file gallery the actual file transfer is started.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented a framework for enabling camera-based smartphone interaction with public screens at a large scale. The introduced architecture eliminates central performance bottlenecks by applying distributed approaches for the relevant tasks such as registration and synchronization. In contrast to previous work, computationally intensive operations are not executed at one central instance but are shared between a decentralized network of Screen Clients and the involved mobile devices utilizing their increasing processing capacities. Based on the prototypical implementation, we developed two showcase applications for demonstrating the features of the framework.

While this paper presented the architecture design and prototypical implementation of the framework, an important part of future work will be a detailed performance evaluation and load testing in order to derive precise guidelines for the large-scale deployment of the presented framework.

Since the presented interaction framework tries to completely spare obtrusive visual markers for both registration and synchronization, this implies an inherent limitation: two adjacent screens (i.e. they cannot be separated by the proposed location/orientation-based query) showing the same content cannot be distinguished. Future work needs to investigate mechanisms to prevent this scenario, e.g. by inserting visual marker only in this case. Further, we plan to address the simplifications we applied in the presented proof-of-concept implementation. For example, we neglected security aspects for prototyping purposes such as preventing the execution of harmful actions via the remote control service, etc.

7. ACKNOWLEDGMENTS

This work has been carried out within the project PRIAMUS financed by FFG and A1. The Competence Center FTW Forschungszentrum Telekommunikation Wien GmbH is funded within the program COMET - Competence Centers for Excellent Technologies by BMVIT, BMWA, and the City of Vienna. The COMET program is managed by the FFG.

8. REFERENCES

[1] Baldauf, M., Fröhlich, P. and Reichl, P. Touching the untouchables – vision-based real-time interaction with public displays through mobile touchscreen devices. *Adj. Proc. Pervasive'10*, ACM.

[2] Ballagas, R., Rohs, M., Sheridan, J. and Borchers, J. The smart phone: a ubiquitous input device. *IEEE Pervasive Computing*, 5(1):70-77, Jan-Mar 2006.

[3] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. SURF: Speeded Up Robust Features, *Computer Vision and Image Understanding (CVIU)*, Vol. 110, No. 3, pp. 346--359, 2008

[4] Boring, S., Baur, D., Butz, A., Gustafson, S. and Baudisch, P. Touch projector: mobile interaction through video. *Proc. CHI'10*, ACM (2010), 2287-2296.

[5] Dachselt, R. and Buchholz, R. Natural throw and tilt interaction between mobile phones and distant displays. *Ext. Abs. CHI'09*, ACM (2009), 3253–3258.

[6] Eng Keong, L., Crowcroft, J., Pias, M., Sharma, R. and Lim, S. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials*, IEEE , vol.7, no.2, pp. 72- 93, Second Quarter 2005

[7] Herbert, L., Pears, N., Jackson, D. and Olivier, P. Mobile device and intelligent display interaction via scale-invariant image feature matching. *Proc. PECCS'11*, 2011, 207–214.

[8] Lowe, D.G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 2 (2004), pp. 91-110.

[9] Madhavapeddy, A., Scott, D., Sharp, R. and Upton, E. Using camera-phones to enhance human-computer interaction. *Adj. Proc. Ubicomp'04*, pp. 1-2, 2004.

[10] Muja, M. and Lowe, D.G. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *Proc. of International Conference on Computer Vision Theory and Applications (VISAPP'09)*, 2009, pp 331-340

[11] OpenCV. Open Source Computer Vision library, http://opencv.willowgarage.com, accessed 23rd January 2012

[12] Pears, N., Jackson, D.G., Olivier, P. Smart Phone Interaction with Registered Displays. *Pervasive Computing*, IEEE, vol.8, no.2, pp.14-21, April-June 2009

[13] Toye, E., Sharp, R., Madhavapeddy, A., Scott, D., Upton, E. and Blackwell, A. 2007. Interacting with mobile services: an evaluation of camera-phones and visual tags. *Personal Ubiquitous Computing*. 11, 2 (January 2007), 97-106.

[14] Wagner, D. Reitmayr, G. Mulloni, A. Drummond, T. and Schmalstieg, D. Real-Time Detection and Tracking for Augmented Reality on Mobile Phones. *IEEE Transactions on Visualization and Computer Graphics*, vol.16, no.3, pp.355-368, May-June 2010