# Mobile Information Systems

Final Project

Jana Puschmann
**Bauhaus-Universität Weimar**
Faculty Media
Media Informatics, B.Sc.
jana.puschmann@uni-weimar.de

Jonas Dorsch
**Bauhaus-Universität Weimar**
Faculty Media
Media Informatics, B.Sc.
jonas.dorsch@uni-weimar.de

## ABSTRACT

The Project *Touch Projector: Interaction through Video* [1] presents an implementation to interact with projectors, that are currently out of reach. Since there are circumstances in exhibitions and presentations where the presenter is not able to control the content of the screen directly, the authors of the paper came up with a solution to control the projections from a distance with a mobile device. Because we believe the chosen methods are kind of unrealistic and inefficient for every day use, we designed an alternative way to remotely control a screen or projection using a mobile device.

## 1 INTRODUCTION

The paper of the German research group [1] introduces a method, to interact and even manipulate objects on an unreachable display or a set of displays using a mobile device. To do so, a live video image stream is captured by the devices camera and displayed. In the following they added a set of interaction and manipulations techniques to increase usability and performance while controlling the correspondent system. The main features include a zoom function, a screen freeze and the ability to transfer files between different screens.
While the ideas of the paper sound quite promising to us, we found a set of problems when it comes to actual real world applications and on that account we decided to build up a slightly different application, which fits a little better in some usage contexts.
We believe the main goal of their paper is the ability to interact with distant, non interactive or unreachable remote screens. Since the approach is based on a live video image feed, the user has to meet two requirements. First of all, the user has to point his device to the screen for the whole presentation. Imagining a lecture or exhibition, demonstrations can take quite a while which makes this task very exhausting over time. Additionally, the presenter has to guarantee a clear line of sight at every point of the presentation. Imagining a science exposition using a multi-screen setup similar to the in Figure 1 of the related paper described scenario, the presenter would have to make sure non of the guests would ever cross his line of sight to any of the controlled displays. Assuring this precondition can be very hard in some cases and for these reasons we came up with an alternative approach we will describe closer in the following.

[1] Boring S., Baur D., Butz A., Gustafson S., Baudisch P. (2010) Touch Projector: Mobile Interaction through Video. CHI 2010: Public Display, 2287 - 2296

## 2 OUR APPROACH

Compared to the initial idea we found it more suiting to build a remote desktop application, which allows to mirror multiple screens in a local network onto your mobile device. We also implemented basic interaction techniques which seem useful for presentations.
We basically developed an application which consists of two communicating tools. On the one hand you got the client application on the android device. On the other hand there are several servers each representing one of the projectors and displays. To connect the client and the servers, we set up two small applications which act as a java socket server and a corresponding client. This setup allows us to connect to an individual amount of servers and therefore control all wanted target screens, as long as we got the pertinent information of these. To keep it as simple as possible, we wanted to require a minimum set of information to connect to a server. All one needs is the local network IP address of the server and in some cases also a PORT if a server can not use the default set port. To simplify this process even further, we included a simple bar code scanner. As long as the presenter generated some bar codes storing the important information in json-format in advance, they can interact with them, connecting to different servers is easily possible.
Since we are imagining a scenario in which the presenter did somehow prepare a slide presentation or something similar, we tried to implement simple navigation gestures. Therefore we wanted to map the swipe functions on the mobile device to the arrow keys on the servers keyboard. Due to the fact, that every common slide or image viewer is able to be navigated through with the arrow keys, we think this mapping fits the goal.
In addition to that we give the user the opportunity to navigate the mouse using the touch display of his mobile device. Using this feature, the lecturer can kind of fully control the remote computer, missing only the ability to control the rest of the keyboard. Even though adding a keyboard navigation should be an easy expansion for future work.

### 2.1 server - side

The server[2] side of our application is held as simple as possible and only contains the most important features. It basically consists of a Server- and a ClientHandler-file. As soon as a connection to the client is established the server starts to continuously capture the current screen using the java.awt.robot extension. After compressing the image to pngformat, it is sent via the sockets outputstream

[2] https://github.com/Jojooo0o/mis-2018-final-project-server

| client input (swipe) | server output (arrow key) |
|---|---|
| left to right | left arrow |
| right to left | right arrow |
| up to down | up arrow |
| down to up | down arrow |

**Tabelle 1: Mapping of swipe direction to arrow keys**

over to the client.
Simultaneously the ClientHandler reads the incoming stream from the client. The incoming data is formatted as a simple string, which will be searched for keywords by the server and according to some predefined cases some action is triggered. At the current state, we implemented the key features needed for basic interaction and navigation through lectures, slide- or picture-shows. Therefore we can currently interpret the arrow keys, a mouse click and a disconnect request. On client side we linked the arrow keys to a swipe gesture, which seems naturally to us, while the click is similar to itself presented by a simple touch on the mobile device. The disconnect on the other hand is send to end the continuous image stream send by the server so the connection will not be closed in the middle of a sending process.

## 2.2 client - side

Our client[3] device connects to our server applications through our android application.
The applications layout is viewed in landscape mode and shows an ImageView with two buttons in the top left corner on top of it.
We implemented immersive full screen mode which allows the user to view the navigation bars by swiping inside from the border of the screen. The navigations will then be displayed on top of the ImageView to avoid a distortion of the screens resolution, which would interfere with the mapping of the users click gesture.
If the user wants to view the display of a server, they need to use the **SCAN** button to scan a QR-code which stores the IP-address and port of the server. The information is stored inside a json-object which could look like the following

$$\{''server\_ip'' :'' 192.168.0.6'',''server\_port'' :'' 8090''\}.$$

To scan the QR-code we are using the ZXING 3.5.0 library. [4]
After the applications has the information to connect to our server, the user can use the **CONNECT** button to create a connection by using a ConnectPhoneTask, so the network activity will not run on the main thread. The client application takes the information from the json-object and opens a socket. Afterwards it creates a MessageThread to send information back to the server.
The client receives image data from the server and displays it continuously.
An onTouchEvent monitors touch gestures on the display. If a user taps on a certain position of the screen, the server will simulate a mouse click at the corresponding position. Else; if a user performs a swipe, the server will simulate an arrow key event (s. table 1).This can be very useful if a user wants to remote control a power point

[3]https://github.com/JaniTomati/mis-2018-final-project
[4]https://github.com/zxing/zxing


**Abbildung 1: Scanning a QR-Code to get the IP-address and port of the server**

presentation from their phone.

Given that the display of the mobile device has a way smaller display than the represented screen, we also had to find a way to make the interaction, especially the mouse control, more reliable. Therefore we implemented a zoom function, which allows the user to perform a pinch zoom and pan in the ImageView. This could allow users to interact with the actual screen with higher precision. Unfortunately it is still in a very experimental state. The pinch zoom uses the display center as focus for the zoom. Panning shifts the image, when the user zooms back, the image is not necessarily centered like it is in the default state.
Also; when zoomed in, the user cannot use click or swipe gestures. The swipe gesture would interfere with the panning of the image. The click gesture can be implemented in the future so it maps the pixel positions of the zoomed in image to the pixel positions of the original resolution. This will also need to be done for server displays that do not have the same resolution as the phones display. Therefore, swiping and clicking is only possible when the scale factor is 1.0.
Since zooming and panning does not work flawlessly, we tried to find an alternative library to use in our application. The PhotoView library by chrisbanes[5] extends the ImageView class and allows zooming and panning and many more features. We tried implementing the PhotoView instead of an ImageView. Unfortunately we cannot use an onTouchEvent listener on this object. Detecting a swipe or click therefore needs to be done differently. We decided to use an onViewTapListner and onViewDragListener. Performing a tap on the display also returns the pixel position of the tap. Sadly, the tap also interfered with the zooming feature of the PhotoView. It would have also been necessary to map the new resolution to the old one to calculate the pixel of the zoomed click position on the server. The onDrag function can detect swipe gestures. Unfortunately that function is called more than once for each swipe. Since the onTouch

[5]https://github.com/chrisbanes/PhotoView

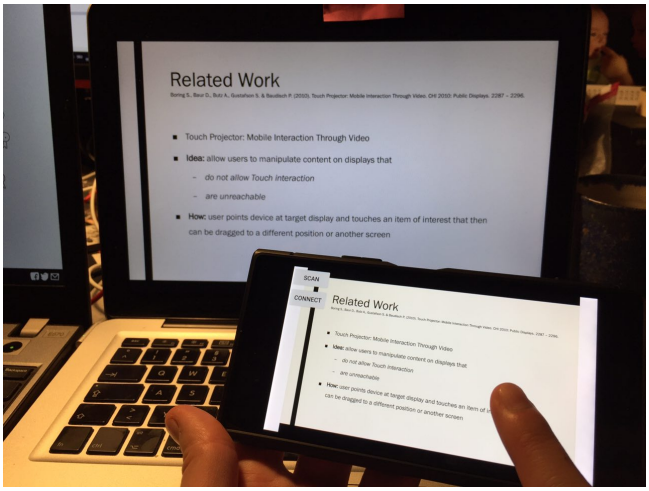


**Abbildung 2: Power point presentation navigation with our application**

function does not work, we cannot detect at which point the finger is really lifted. In this state the onDrag method sends more swipes to the server than actually performed.
This would destroy the purpose of the application in practice, which is why we decided to go with our first approach. The implementation of the second approach can still be found in comments in our repository.

## 3 CHALLENGES AND DIFFICULTIES

During our approach we encountered a number of problems and limitations to our application. First of all, sending a continuous image stream from the server to a client over a local network requires a steady and fast connection. Slower networks will highly decrease the frame rate and therefore usability in any kind of scenario. Since we are using the png standard, further compressing of the images can only be achieved by special and complex algorithms. On the other hand png already makes use of the deflate algorithm also used by the zlib library and therefore does some compression itself. Also further compression will add additional time exposure, depending to the algorithm the required time will increase way to much to guarantee a live image feed.
During the project we noticed the use of java socket servers require a lot of caution when it comes to handling interruptions and disconnects between client and server. Due to the lack of experience we did not manage to achieve a flawless implementation which can handle any kind of disconnect or connection interrupt. We tried to build our way around most of these problems by sending the previous described disconnect notification to the server. Nevertheless; in the current version you should be able to connect to any number of devices and reconnect to the previous again without having any problems.
Since we are mirroring a full size desktop screen on a mobile device, the displayed image is very small and for some operations not well suited. We try to counteract against that with the zoom function but there are for sure some options to optimize the interaction and the zoom function to increase usability. At the current state we got two versions of our client application. The currently active version makes use of pinch zooming and also is able to translate the zoomed image. We did not manage to correctly map the key input in the zoomed image, therefore we currently disabled clicking in the zoomed view. Also, the translation leads to misalignment which causes the image to not be centered correctly at every time. The second approach is currently also available in the code but commented out. Here we use the PhotoView [6] library instead of the basic ImageView. The PhotoView library extends the ImageView class with additional functions like pinch zoom and fling zoom. The zoom and translation is handled way better by the library, but the native touch detection can not be used anymore when using this approach. If we do not zoom at all, the touch works just fine disregarded the fact that it is very hard to hit the right spots then.

## 4 FUTURE WORK

As already mentioned above, there is a lot of improvement and additional feature integration possible. Improving the performance of the image stream should be number one priority for future optimization to increase usability in low speed networks as well. A compression in means of a loss in quality will lead to some disadvantages for sure but further studies could show a way how decrease quality and increase performance at the same time.
Following to that the correct handling of disconnects and disrupts in client server socket connection has to be handled to make the system real world scenarios suitable. Once neither the server nor the client causes each other to totally collapse, one can use the application without worrying about any kind of connection issues any more.
A highly increase in performance will happen if the zoom function will be optimized to a part, where clicking is able. This will make it more easy to hit specific buttons and navigate the remote computer. Studies and use tests may show how an optimized zoom function could be further improved.
Afterwards adding any kind of additional interaction techniques might be a good idea at that point, even though one has to think about the needs and complexity of more features. Since there is feedback for the keyboard partially anyway, giving access to the whole keyboard might be a first step, which will very likely does not affect the performance at all or imperceptibly.

## 5 CONCLUSION

The current implementation of our project allows to mirror the screen of a distant or unreachable display as long as they are connected to the same network and the user got the possibility to read in the IP address of the correspondent display. To simplify this task scanning a bar code is possible in our application. Since you can easily find QR-code generators for free with eligible input, one can pre-generate a set of bar codes encoding the important IP addresses and make them accessible for the presenter. Switching between multiple displays and going back to the previous screens works just fine. Therefore navigating and controlling multiple screens in any

[6] https://github.com/chrisbanes/PhotoView

given scenario is made possible. As shown in the end of the vi
With some basic interaction techniques the user is able to navigate through slides and even got the possibility to perform tasks that make use of the primary mouse button. Supporting the user interaction, the zoom function magnifies the screen and would make interaction more easy due to bigger representation on the small mobile screen if a correct mapping is added. At the current state the magnification can still be used to help the user orient or get a more detailed and readable view on the mobile device.
After applying the mentioned optimization in the future work part, the application should be very well suited for the described use cases, but we think the over all usability of our system is already quite good and has some actual real world usage compared to the method described in the paper. Except of the limitations resulting of a slow network connection, we think the app can be used for example in a teaching context, at exhibitions or in every other case where the lecturer might not want to be bound to a location but instead be able to control one or multiple screens from all over the room. It could even be used as a remote control for use at home.

## 6 EXTERNAL SOURCES

[1] Khan B. (2015). Android QR Code Scanner Tutorial using Zxing Library. Simplified Coding.
https://www.simplifiedcoding.net/android-qr-code-scanner-tutorial/

[2] (2015). Immersive Mode Android Studio. Stackoverflow.
https://stackoverflow.com/questions/31482522/immersive-mode-android-studio

[3] Rishabh (2014). Using onTouchEvent() and View.OnTouchListener Interface with MotionEvent to Detect Common Gestures Like Tap and Swipes on Android. Codetheory.
http://codetheory.in/android-ontouchevent-ontouchlistener-motionevent-to-detect-common-gestures/

[4] (2017). How to save state of image 505. GitHub.
https://github.com/chrisbanes/PhotoView/issues/505

[5] Trust Onyekwere (2018). Pinch-to-zoom with multi-touch gestures In Android. Medium.
https://medium.com/quick-code/pinch-to-zoom-with-multi-touch-gestures-in-android-d6392e4bf52d

[6] Nick Casey (2017). The Evolution of Trust.
https://ncase.me/trust/