INF

Studiengang
Medien- und
Kommunikationsinformatik

Hochschule Reutlingen
Reutlingen University

Uwe Kloos, Natividad Martínez, Gabriela Tullius (Hrsg.)

Informatics Inside:

Human-Centered Computing

Informatik-Konferenz an der Hochschule Reutlingen
30. April 2014

ISBN 978-3-00-045427-1

9 783000 454271 >

Impressum

Anschrift:
Hochschule Reutlingen
Reutlingen University
Fakultät Informatik
Medien- und Kommunikationsinformatik
Alteburgstraße 150
D-72762 Reutlingen

Telefon: +49 7121 / 271-4002
Telefax: +49 7121 / 271-4042

E-Mail: infoinside@reutlingen-university.de
Internet: http://www.infoinside.reutlingen-university.de

Hochschule Reutlingen
Reutlingen University

Copyright: © Hochschule Reutlingen, Reutlingen 2014
Herstellung und Verlag: Hochschule Reutlingen
ISBN 978-3-00-045427-1

Inhaltsverzeichnis

Analysis of Finger- and Palm-based interaction paradigms for Touch-Free Gesture-Based Control of Medical Devices with the Leap Motion Controller

Stanislas Mauser
Reutlingen University
stanislas_christophe_yves.mauser@ student.reutlingen-university.de

Abstract

There are several intra-operative use cases which require the surgeon to interact with medical devices. I used the Leap Motion Controller as input device for three use-cases: 2D-Interaction (e.g. advancing EPR data), selection of a value (e.g. room illumination brightness) and an application point and click scenario. I evaluated the Palm Mouse as the most suitable gesture solution to coordinate the mouse and advise to use the implementation using all fingers to perform a click. This small case study introduces the implementations and methods that result those recommendations.

Keywords

Gesture recognition, touch-less interaction, human machine interface, medical device control

CR-Categories

H.5.2 [**Information Systems**]: User Interfaces

Betreuer Hochschule: Prof. Dr. Oliver Burgert
Hochschule Reutlingen
oliver.burgert@reutlingen-university.de

Informatics Inside 2014
Wissenschaftliche Vertiefungskonferenz
30. April 2014, Hochschule Reutlingen

1 Introduction

1.1 Goals

During a surgical intervention, a surgeon depends on the availability of patient information such as risk areas of tumor margins, further he has to control surgical devices like endoscope, x-ray, drill etc. Nowadays, this control is either performed by using dedicated sterile equipment like foot pedals or by directing a surgical assistant or nurse. This impedes the execution of the surgery, increases the risk of infection for the patient and is generally associated with a higher expenditure of time (see [1], [2]).

For a subset of the use cases described above, a contact free, gesture based controller, which can be used by the surgeon himself might be helpful. Such a touch-less approach should be reachable for the surgeon while operating and without needing special gloves or additional tools.

In [1], several surgeons were interviewed whether they prefer to instruct other persons to acquire information or access requested information themselves by using e.g. new gesture recognition based system technologies. Over 80% of the surveyed surgeons preferred to search for information themselves and not by instructing another person. Using this information and the fact that the availability of touch-based systems (like smartphones and tablets) and pose tracking systems like Kinect pave the way towards additional intuitive interaction devices, such a solution is desperately needed.

1.2 *State of the art*

There exist multiple examples for contact free input devices: In [3], a sterile solution to navigate the planning system MeVis with a Wii Remote Controller is described. This allows gesture recognition, but the controller needs to be touched. Gesture recognition can be performed based on tracked surgical instruments. In [4], such an approach is used to detect surgical gestures performed by a Polaris® navigation system pointer. A Microsoft Kinect is used for contact free interaction with the InVesalius program. Here the surgeon may contact-free interact with the program with one hand, to visualize 2D and 3D images [5].

A commercially available solution is the Mi-Report developed by Fraunhofer HHI for Karl Storz which allows the presentation of patient data with contact free gestures control, specifically for use in the sterile medical field [6]. The camera system is called HHI Handtracker and is based on an infrared camera system, which allows a fast and robust detection and tracking of fingers and gestures. The cameras are mounted to the hand orthogonal to the ceiling. By using infrared-based cameras, the system is also less insensitive to unwanted light of other devices or the sun. Several fingers are recognized and processed in real time (50 Hz). The 3D coordinates are then passed to the application.

The available systems have several drawbacks: Either they require a dedicated input device for gesture recognition (navigation system, Wii controller), or the gestures recognition is relatively coarse. Furthermore, large gestures result in physical fatigue.

2 Methods & Materials

For this solution I used the new Leap Motion Controller: A small device, not bigger than two fingers, but with the capability to track "ten fingers up to a precision of 1/100th of a millimeter" [7]. The tracking volume of the Leap Motion Controller is about 20 to 600 mm semicircular around the device. I used a development kit including a Leap Motion Controller revision 0.6.5., C# as programming language, and the Visual Studio 2013 IDE.

Development hardware included a Apple Mac Pro running two Intel Xeon Westmere CPUs, 24GB RAM, two ATI 5700 HD graphic cards and Windows 8 in a virtual machine. For evaluation and presentation, a Lenovo W520 running a Intel i7 CPU, 8GB RAM and a nVidia Quadro 2000M graphic card was used.

2.1 *Contact-free, gesture-solution prototype for medical devices*

The high precision of the Leap Motion offers much more possibilities then similar devices like the Microsoft Kinect. But basically, the problem of finding a gesture which is easy to learn and intuitively to use, is the same as with other devices. There are no commonly accepted gestures for 3D contact-free input devices, yet. Best practices gestures from touch-based systems, like Apple iPhone or similar Point-Gestures can be implemented with the Leap Motion, but must be seen as starting point for the development of truly intuitive 3D input gestures.

In the medical context, I identified different classes of interaction: binary interaction (e.g. a simulated mouse click), simple 2D interaction, like advancing to the next picture or patient record entry, entering a value (e.g. adjusting OR table height or control a light source), point and click for selection of objects, and complex 3D-interaction, e.g. rotating a volumetric patient reconstruction. For the first prototype, I implemented the "simple 2D interaction" and "entering a value" use cases, in a second step, a point and click scenario was realized.

2.2 Contact-free approaches to control a mouse pointer

There are no studies which evaluate best practices in contact-free mouse control, yet. Therefore, I developed a prototype which offers the possibility to switch between different implementations to control a mouse pointer using the Leap Motion Controller and to test multiple concepts to emit a left or right click event.

The current version of the prototype has two different implementations: The first uses one finger as mouse with two different methods to perform a left click event. The second concept controls the mouse by palm orientation. For the Palm-Mouse I implemented two different click methods.

2.2.1 Mouse navigation using one finger

This implementation uses one finger to map the mouse movement to the position the user points to. If the Leap Motion Controller API recognizes more than one finger, the foremost one is chosen as reference. The Leap Motion API represents a finger as directional vector. The position in xyz-space of the Leap Motion Coordinate System is mapped linearly to the screens xy-coordinates.

One of the implemented click approaches uses the so called Touch Zone provided by the Leap Motion SDK. This is a defined plane in front of the controller. The SDK can "report whether that pointable object is close-to or touching this imaginary surface" [8]. Whenever a touch event is detected, a mouse click is triggered.

The second click approach uses a simple tap-gesture, which is one of the four default gestures of the Leap SDK: While navigating the mouse with the frontmost finger, the thumb is used to perform a tap-gesture creating a click event.

For the study described in 1.3, I implemented a rule allowing to apply a click even though it is slightly next to a button, if this button is slipped over within the next 600ms.

One problem with this implementation is the precision of the Leap Motion controller. Because the precision of the controller is very high, every trembling of the finger is also detected, which may result in an unstable behavior of the mouse pointer.

2.2.2 Mouse navigation with a single finger and optional precision mode

Reason for implementing this mouse solution was the problem described in the section above. In order to work more precisely, I implemented a so called precision mode.

The general mouse movement using the frontmost finger is identical to the implementation described above. In addition, the mouse movement velocity is reduced to a hundredth of the actual movement by lifting the thumb additionally to the frontmost finger.

The options to perform a click event remain the same.

2.2.3 Palm-Mouse implementation

The navigation with the palm has been used in the Light Control application (see [9]) and is known from gaming applications using the Leap Motion. The tilt of the palm is used to navigate the mouse: The steeper the tilt, the faster the mouse movements. The mouse position does not change if the palm is fully orthogonal to the controller. With e.g. 5° variation from the orthogonal position, the mouse position changes are nearly zero.

This implementation eases the stabilization of the mouse position at a desired position, compared to the finger mouse implementation where the user needs more cognitive concentration to ensure the finger doesn't vary its position heavily. Moreover, all fingers can be used for a finger tap-gesture to perform a left click.

To perform mouse click events I implemented two approaches. One approach uses the simulated touch area from the SDK. If this click method is activated, it is important to interact with the hand in front of the controller. When the mouse pointer is over a desired object, a user fires a click event by moving the palm to the direction of the controller and staying meanwhile orthogonal to the controller.

The second approach uses the built-in tap-gesture of the SDK. With this method it is not important where the palm is, as long as it is in the range of the Leap Motion tracking area. With a tap gesture performed by one arbitrarily selected finger, a left click event is fired.

2.3 Application for a click-case study of the Leap-Mouse implementations

To evaluate the usability, performance and acceptance of each of the implementations, I performed a first case study.

For this study, I developed an application which can use all mouse implementations. I defined five different button sizes, each size is shown six times on the evaluations software GUI. To detect the possible minimal button size the study includes buttons of 10x10 pixel, 15x15 pixel, 30x30 pixel, 50x50 pixel and 80x80 pixel size. While testing the mouse interaction, successively one of the possible 30 buttons is shown to the user. After clicking on a button, another button in a different size appears on another spot of the GUI. The buttons are arranged randomly on the screen, but the same order and position is guaranteed for each implementation and test person. If the user fails to click a button for five times, the current button is skipped.

The application logs the following information:

- Test person ID
- Distance between the button and the mouse pointer at the time the button appears;
- Sum of time the mouse was over the button;
- Sum of failed clicks;
- Sum of time until a button was clicked or skipped;
- And the information whether this button was skipped.

This data is automatically exported at the end of a test run into an Excel file.

Every implementation test followed a same procedure: Before the study begins, I introduced all mouse implementations to the test persons and allowed unlimited time to try each implementation, until they felt ready to start the study. During this warm-up phase, all 30 buttons are visible and clickable. All five implementations described above are part of the study; their order is hard-coded but can be tested also in reversed order. I distinguished the sequence of the implementations for each test by giving each test person having an even personal ID the reversed order of the five implementations. The personal ID is equal to the number of test persons, in their temporal order.

2.4 2D Interaction

2D interaction is realized by performing a swipe gesture to the right or to the left, orthogonal over the Leap Motion Controller. To detect a swipe I implemented a two-staged process: At the beginning, I used the build in Leap SDK swipe detection. On top of that, I defined three threshold states. The first threshold starts the swipe process and defines whether it is a left or right swipe. The next two thresholds and the speed of the finger or hand ensures that the gesture was really intended. The gesture detected can be mapped to a key-press event (e.g. cursor left, cursor right). This allows control of applications which were not developed with contact-free interaction in mind. I implemented a mock-up of an electronic patient record viewer to show the feasibility of the approach.

Evaluation of the detection capabilities and error rate was performed by controlling the PowerPoint presentation of a 90 minutes lecture on computer science (Informatik II) at Reutlingen University. The contact-free 2D interaction was the only input device for slide advancement. The long duration of 90 minutes and the real lecture situation ensured that the teacher was not concentrating on using the interaction device correctly but on the teaching situation. The professor had no training on the device and started after a 20 seconds test. Afterwards, a video recording of 60 minutes of the lecture was analyzed to verify how many unwanted gestures were performed and how many gestures failed.

2.5 Entering a value

Since there is no commonly accepted interaction scenario for entering a value, yet, I implemented a control interface which allows for mapping different types of gestures to a method which finally passes the calculated value to the target application.

The gestures for entering a value take the best practice gestures from touch devices into account. I used the built-in gestures of the Leap Motion and additional values like pitch, yawl or the y-position of the hand, which can be read out from the Leap Motion. Gestures like “drag and swipe” seem to be intuitive, but detection of start and stop position is difficult. Along with this, the problem of how the given minimum, maximum and current value shall be represented occurs.

2.5.1 Counting fingers

From a technical and logical viewpoint, this is the simplest gesture. Analog to the sign language, the surgeon can modify the value by showing up to five fingers orthogonal over the leap motion Controller. This can be mapped to the values 0-5 or to a percentage of the value range. E.g. for setting of 20% of the maximum value, a surgeon shows only one finger, etc. The problem of this solution is the limitation on only five levels. The termination of the gesture must be defined properly to prevent misinterpretations: After the desired value has been set and the hand is removed from the detection area, the fingers successively disappear and the value would decrease to one or zero.

2.5.2 Mapping the palm orientation

A rolling gesture of the palm, analogue to an airplane‘s wing that rolls down to the right or to the left for navigation, was implemented as an alternative. The vertical palm position, relates to 50% of the maximum value. By rolling the palm to the right, the value increases and when rolling to the left, the value decreases.

Like in the counting fingers scenario, unwanted values sometimes occurred when removing the hand out of the tracking area. This effect happened much less frequently than in the number of fingers gesture, but often enough that it hinders productivity. Therefore, the number of detected fingers on the hand that performs the roll gesture was added as an additional condition. For example, if two fingers are set as the restriction modification, the values will only change when at least two fingers are recognized by the controller. So if the surgeon has set the desired value, he can simply make a fist and remove the hand without concerns out of the detection area.

2.5.3 Mapping the vertical hand position

The third gesture approach is based on the vertical movement of the palm position. In this scenario, the distance between the location of the controller and the measured center of the palm will modify the value. In the prototype, 250 sampling stages were implemented, starting from about 15 cm above the controller and ending at about 40 cm. In the graphical prototype this gesture is mapped to a vertical slider.

In most cases the values rest untouched, when the hand is removed slowly to the side out of the detection area. But in few cases, errors still occurred and moreover, for the use case during the OR course, it can´t be a requirement to remove the hand slowly. To compensate this, the technique of restriction by a certain number of fingers, as described above, was introduced.

2.5.4 Lock and unlock gesture

In analogy to the Mi-Report [6], a lock-unlock gesture was implemented for this gesture concept. To reduce the possibility of unintended gesture input during the intervention, a defined gesture for locking and unlocking the detection of gestures was implemented. For unlocking the gesture detection, the surgeon has to hold his palm orthogonal over the controller with all fingers extended and as straight as possible for a defined duration. In the prototype these conditions can be adjusted at run time, since it's not entirely clear how precise the check of the gesture must be. Adjustable variables can be for example the length of time, the gesture has to be executed without interruption, the minimum number of fingers it has to recognize, and the maximum pitch and roll valued of the palm.

This gesture toggles the locking and unlocking condition, so that the same gesture can be used for both. But additionally the gesture detection is automatically locked after a defined amount of time, when no changes to the light value occur. The current default value in the prototype is three seconds.

3 Results

3.1 2D Interaction study results

During 60 minutes, 99 gestures were performed: 78 intentional gestures and 21 unwanted gestures. Unwanted gestures resulted from arm movement, putting down chalk, etc. 37 of the intentional gestures were performed to correct unwanted gestures. 58 of the 78 intentional gestures were recognized correctly and triggered a slide change. Thus, ~25% of the intended gestures were not fully recognized because they were not performed in the working space or the gesture was not performed properly.

In an interview, the performing teacher stated that the interaction was simple and fun, but the false positive rate was too high for routine use.

3.2 Click-Prototype study results

Following charts describe the results of the case study for the mouse click prototype. Those results are representing the findings of the study, which has been fulfilled and analyzed with 12 participants (6 female, 6 male, average age is 23.5).

Figure 1 shows the relation between the time until a button was pressed compared to the different button sizes. Recognizable is that with an increasing button size the click appeared earlier. The slowest implementation is the line (C), the fastest is the line (B).

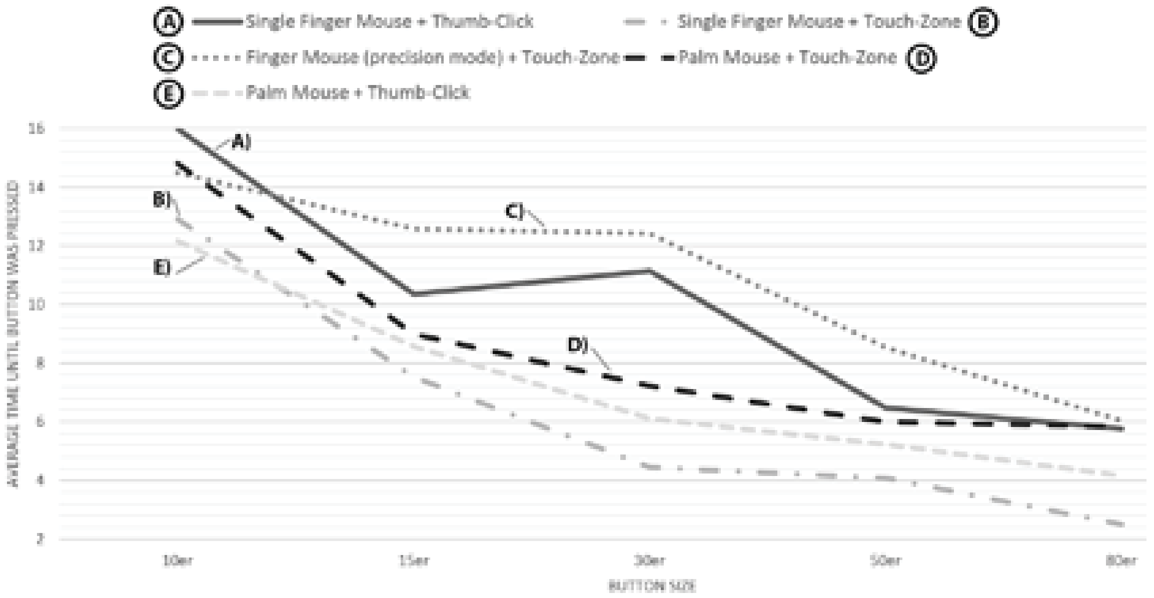

Figure 1: Time until button was pressed

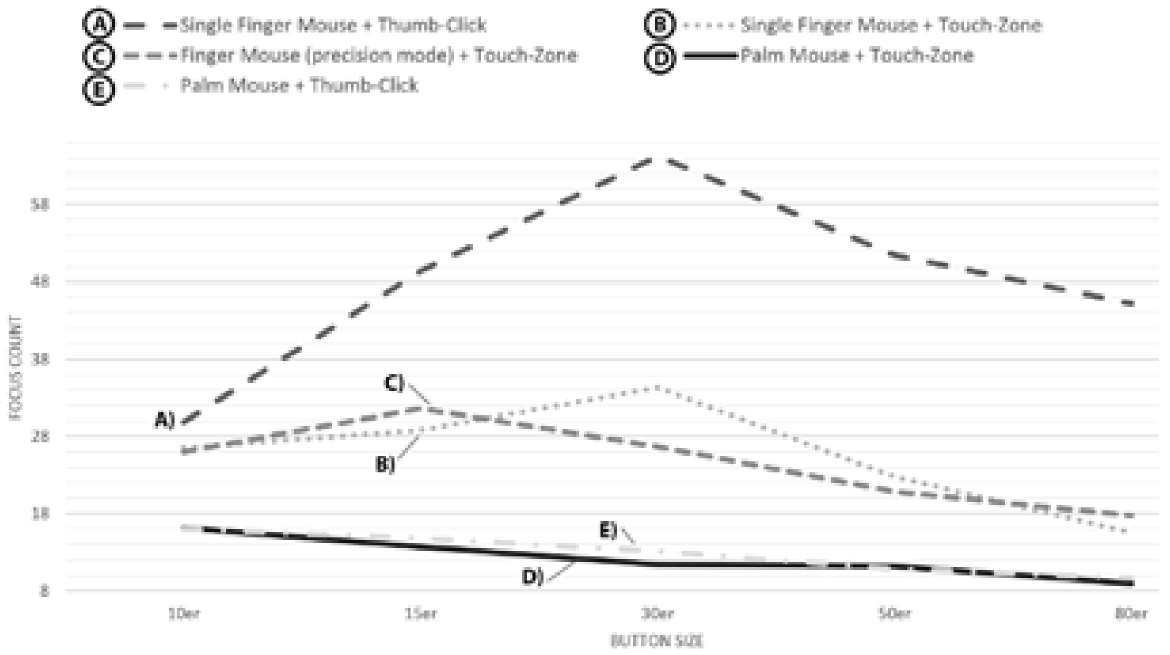

Figure 2: Relation of lost focus on a button to button size

This figure 2 above shows the relation between the sum of focus lost and the different button sizes. Focus lost represents the amount how often the mouse was over a button but lost its focus again before the button was finally clicked. The figure also displays, that there are no significant differences comparing both Palm Mouse Implementations. Generally it is recognizable that the two Palm Mouse implementations achieve the best overall rate of minimum lost focuses compared to all other implementations.

Figure 3 illustrates the relation between the amount of failed clicks and the button size. Failed clicks are mouse clicks beside a button and not direct on a button.

The implementation (E) shows a significant better result to all the other four implementations. As well it shows that the average number of failed clicks of the Palm Mouse Implementation received already with the 15x15 pixel button size a constantly low value of less than five fail clicks per person.

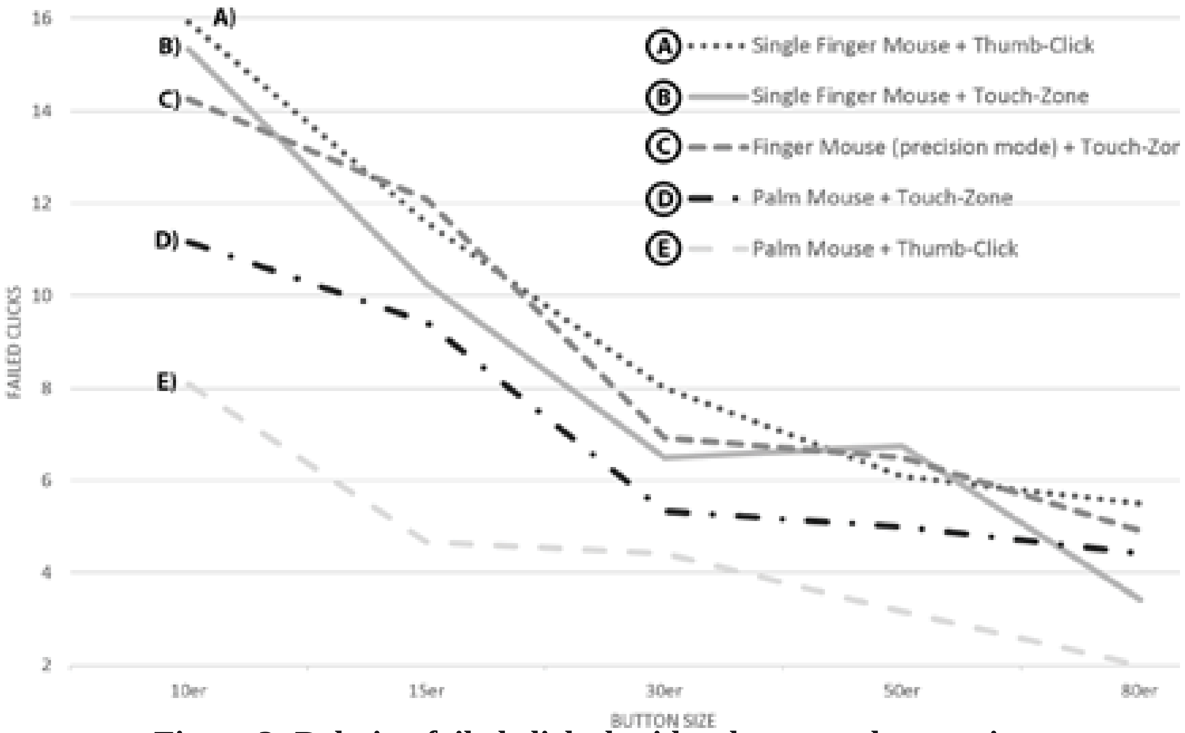

Figure 3: Relation failed clicks beside a button to button size

This graph shows that the three finger mouse implementations had similar results.

Figure 4 visualizes the amount how many buttons per person were skipped. It is considerable to mention the extreme high value of skipped buttons using the Single Finger Mouse with Thumb Click Implementation for the smallest button size. This is due to a slight finger movement when the thumb is moved to trigger a click.

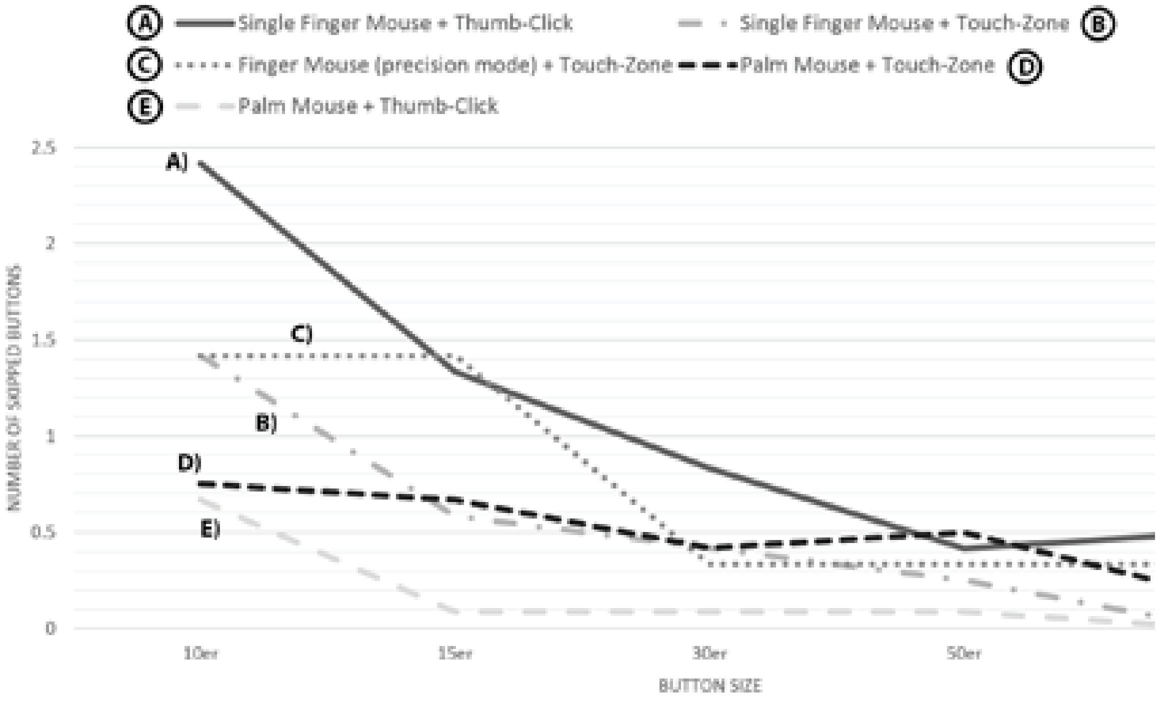

Figure 4: Relation Skipped Buttons to button size

The line (A) shows the best results over all button sizes.

Figure 5 outlines the usability of the button size comparing it for all implementations. The usability is rated through the relation of the following data: the amount of skipped buttons, the amount of failed clicks and the average click time per person with an additional priority value to scale the importance of an argument. The faster a person clicked an icon, the better are the results. Therefore this graph shows the best results within the lowest bar.

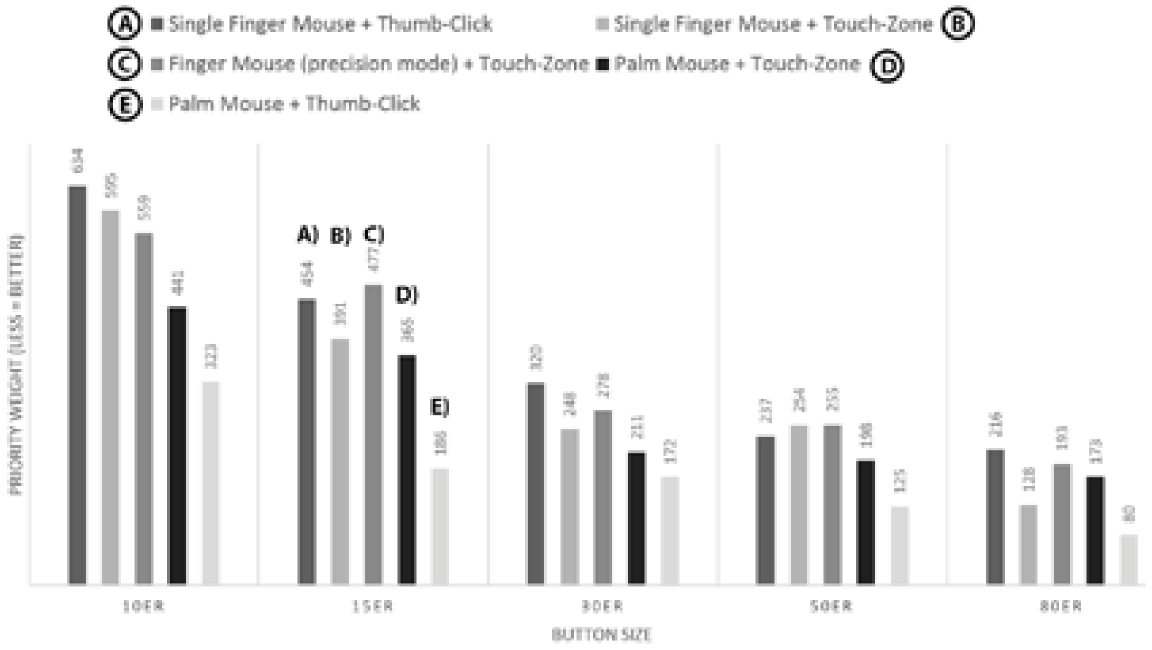

Figure 5: Comparison of usability of all implementations per button size

Goal is to evaluate the test person's favorite button size and implementation. Figure 5 shows that the 30x30 up to 80x80 button sizes have are rated equally. Secondly, I can identify that the line (E) archived better results to all other implementations. Mostly it is twice as good compared to the line (A).

3.3 *MDeviceControl gesture prototype*

In the prototype, the gestures approaches were all mapped to horizontal or vertical sliders to show potential testers the effect of their gesture movements. Both gestures, the palm roll gesture and the vertical movement gestures are implemented and visualized in the prototype with and without the additional condition of **n** fingers, for restriction of unwanted value modification during the gesture process.

In the prototype, as can be seen in figure 6, the lock- and unlock gesture conditions can be adjusted in the 3rd box. In the second box the detected gesture is visualized in values. These both boxes are just for the prototype and not for the productive working solution implemented. On the bottom left, the pitch and roll state of the hand is visualized by arrows, which is more intuitive as plain numbers.

First tests with untrained subjects showed that with four or more fingers for three seconds and a maximum pitch/roll-deviation of 15° it is quite sure that this gesture is not unwanted performed during the OR, but can still be carried out quickly and easily.

4 Conclusion & Discussion

This section points out the main consolidated findings I achieved based on this work. The point-and-click study clearly favors the palm-based control method with thumb click over the finger based methods. Further studies which are having a setting closer to the intended clinical use cases can be designed based on this finding and may lead to even better interaction methods. Those studies must be performed with the intended end users of such systems, such as surgeons, interventional radiologists or other clinical staff.

Figure 2 shows that both Palm Mouse Implementations reached the best results in not losing the focus regardless of the button size.

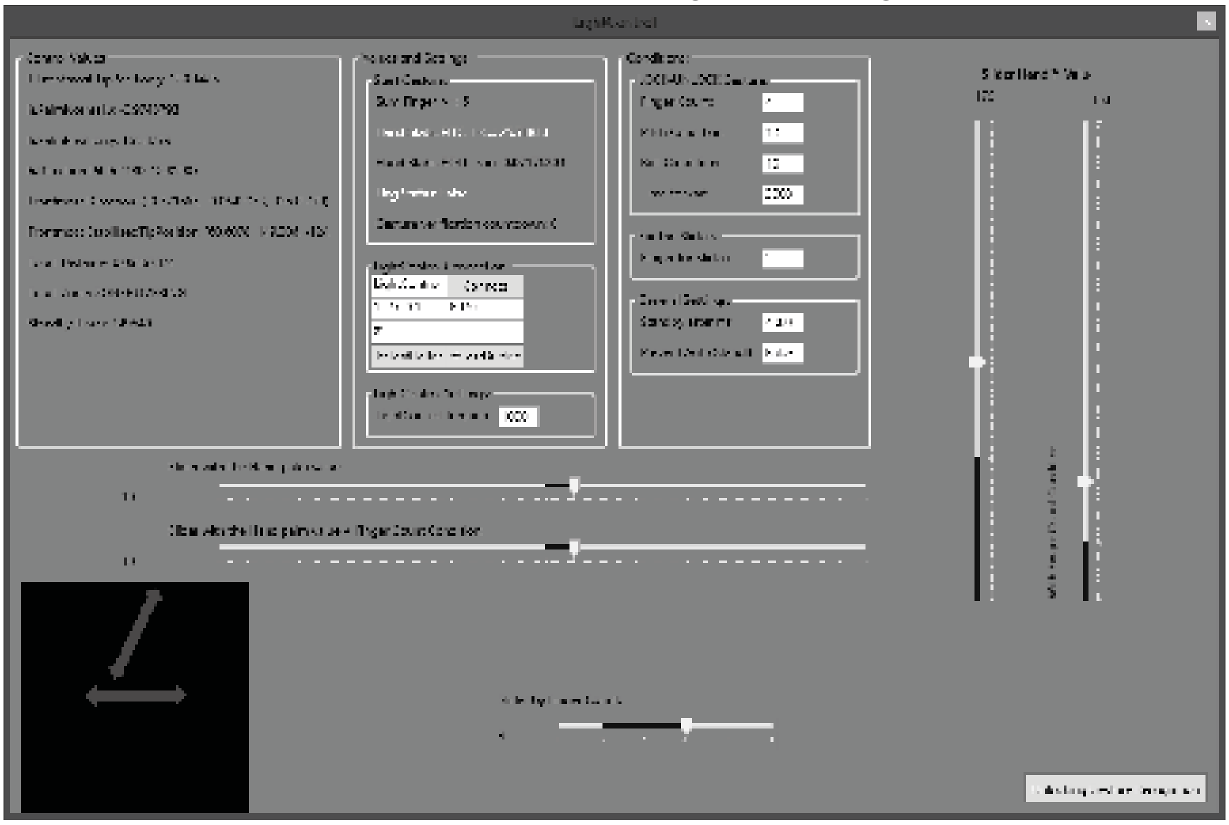

Figure 6: Screenshot of MDeviceControl

Moreover, the Palm Mouse with Thumb-Click got the best results. The Single Finger Mouse with Touch-Zone was the implementation, that allowed to press a button the quickest, the Palm Mouse Implementation reached the second place (see figure 1). Both figures show that with increasingly button size the results are getting proportionally better.

An important measure to evaluate the usability of a contact-free mouse control is the comparison of failed clicks and the button size. The Palm Mouse Implementation received the best score, and for the largest button size of 80x80 all test persons had no failed clicks at all (see figure 3). This is an important aspect for surgeons, who can't spend much cognitive effort for finding the right button.

This interaction paradigm is also the one with the fewest number of skipped buttons (see figure 4). Furthermore, it is constantly low from a button size of 15x15 on whilst other options reach a constant level at a size of 30x30 or even at 50x50.

Figure 5 shows that it gives the best results for all button sizes. The possible minimal button size can be 15x15, for the other implementations it generally starts from a size of 30x30.

Therefore, I expect the Palm Mouse with Thumb-Click interaction to be the most precise and least stressful interaction method for realizing a point-and-click scenario.

Additionally to this technical analysis the results of the questionnaires showed same results: 75% of all test persons prefer the Palm Mouse Implementation. 50% of all test persons favour a 30x30 button and the other 50% decided that a 50x50 button was finally the preferred one size. Besides the analysis results it is important to assess the Leap Motion Controller. This is a stable 3D gesture input device which has potential applications in intra-operative situations. The use-cases have to be carefully selected and the best gestures for each application have to be detected and implemented. The developed MDeviceControl prototype is a valuable tool for user testing and rapid development.

Based on the studies and tests performed, it is clear, that a lock- and unlock gesture is mandatory, especially if precise and intuitive tool control is crucial.

Still not evaluated are the optimal light conditions for the Leap Motion Controller, since unwanted light and the impairment of its emitted IR light could result in reduced recognition rate. I did not detect interferences with surgical navigation systems using IR light, yet. The next step will be to simulate lighting conditions found in operating theaters with or without daylight access and OR illumination.

5 References

[1] C. Dressler, T. Neumuth, M. Fischer, O. Abri, G. Strauss: Intraoperative Bedienung einer elektronischen Patientenakte durch den Operateur, In: HNO, vol. 59 issue 9, Universität Leipzig, September 2011, DOI: 10.1007/s00106-011-2331-y, 900-907.

[2] P. Chojecki, U. Leiner: Berührungslose Gestik-Interaktion im Operationssaal, i-com, Vol. 8, Issue 1, Mensch-Computer-Interaktion im Operationssaal, DOI 10.1524/icom.2009.0003, 13-18.

[3] F. Ritter, C. Hansen, K. Wilkens, A. Köhn, H. Peitgen: Benutzungsschnittstellen für den direkten Zugriff auf 3D-Planungsdaten im OP, i-com, vol. 8 issue 1, Mensch-Computer-Interaktion im Operationssaal, DOI: 10.1524/icom.2009.0005, 24-31.

[4] S. Franke: Integration intraoperativer Patientendaten in ein TIMMS Patientenmodell am Beispiel der Lokalisation des Sulcus Centralis, Diploma Thesis, Universität Leipzig, 2010.

[5] G. Ruppert, L. Reis, P. Amorim, T. Moraes, J. Silva: Touchless gesture user interface for interactive image visualiza-

tion in urological surgery, World Journal of Urology, Ausgabe 5, 2012, DOI: 10.1007/s00345-012-0879-0, 687–691.

[6] S. Schrader, U. Leiner: Neuartiges System zur Visualisierung von Patientendaten mit berührungsloser Handgestik-Steuerung, In: Workshop-Proceedings der Tagung Mensch & Computer, Berlin, 2009, ISBN: 978-3-8325-2181-3, 302-304.

[7] Leap Motion official website at https://www.leapmotion.com/ (last access: 14.06.2013).

[8] Leap Motion developer website: Touch Emulation; https://developer.leapmotion.com/documentation/java/devguide/Leap_Touch_Emulation.html (last access: 14.06.2013).

[9] S. Mauser, O. Burgert: Touch-Free, Gesture-Based Control of Medical Devices and Software Based on the Leap Motion Controller, In: Medicine Meets Virtual Reality 21; IOS Press, 2014, DOI: 10.3233/978-61499-375-9-265, 265-270.