Evaluation of Textual Feedback for Incorrect Usage of an Augmented Reality Application

Julian Lüken, Mehmed Mustafa, Jan Schneider,
Steffen Tunkel, Chris Warin

March 28, 2020

Georg-August University, Göttingen

Abstract

Mobile Augmented Reality (AR) applications have gained in importance over the years. Although they provide a variety of uses, they remain mostly unknown by the public, which results in poor understanding of the controls and features. Our research goal is to find out how to react to incorrect AR usage in order to enhance the usability of mobile AR applications. For this purpose, we created three generic textual feedback implementations for AR applications with different content. In order to assess the potential gain in usability provided by these feedback messages, we conducted a case study with two interactive prototypes. Each combination of prototype and feedback implementation was considered and each of the 40 participants had to complete the same set of tasks. We found that, given appropriate feedback, most people still use incorrect gestures, but quickly recover from mistakes and are much less likely to do them more often.

Introduction

In recent years, the number of available mobile AR applications for smartphones has increased [1]. Although mobile AR technology is no novelty, it was not widely available for public until most smartphones became capable of running AR applications. Since most users are newly discovering AR technology, they do not know how to use its features yet, thus using unsupported gestures as inputs, which leads to frustration. Our main motivation for conducting this research is the lack of good feedback in case of incorrect usage, which could help a lot of users to avoid repeating common mistakes while using AR applications for smartphones. Providing good feedback could also help and teach users to control such AR applications without the need of a separate introduction.

Our research goal is to provide appropriate feedback to unsupported gestures and decide what type of textual feedback is the best. For validation of our approach we conducted a usability case study by using AR prototypes of technical devices. Test subjects were divided into groups and they had to complete several tasks with the prototypes. Different kind of feedbacks were provided to them according to their group.

The organization of the paper is as follows. The section Foundations gives information about usability in general and the AR environment we provide the feedback for. In the section Related

Work we discuss similar work on AR and feedback done in the past. The Approach section gives a detailed illustration of the feedback we present to users. In the section Case Study, we talk about the setup case study we conducted in order to validate our approach, followed by our results and the discussion thereof. Lastly, we summarize and conclude in the Summary and Outlook.

Foundations

Since the main goal of our work presented in this paper is to find suiTable user feedback for augmented reality environments, we will further clarify what AR means and which technologies we used. AR itself describes a variety of technologies that place 3D virtual objects into a 3D real environment in real time, for example via a handheld device like a mobile phone (mobile AR) or using wearable goggles like the Microsoft HoloLens. In this paper we focus on AR for mobile phones, where recorded imagery of the phone is being augmented and shown to the user to display a mix of virtual objects and the real world at runtime.

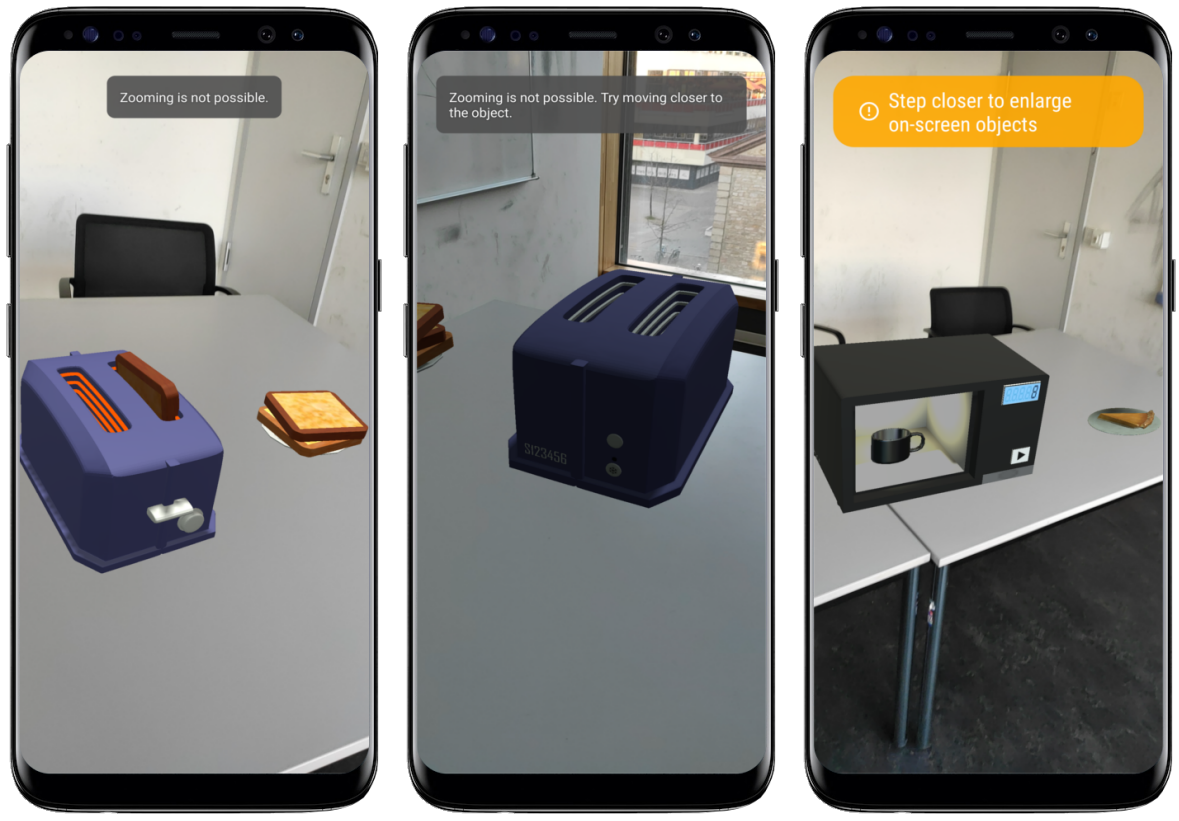


Figure 1: Three screenshots of an app made with the Vivian framework

A virtual prototype is a *"[...] computer simulation of a physical product that can be presented, analyzed, and tested from concerned product life-cycle aspects such as design/engineering, manufacturing, service, and recycling as if on a real physical model."* [2]. The Vivian frame-

work is an extension for Unity3D, that can be used to create mobile AR implementations of prototypes using a 3D model and one or more finite-state machines. (sauce?)

In a mobile app made with the Vivian framework, all user input relies on one finger gestures on the touch screen. In the example in Figure 1 (A and B) we can observe a 3D model of a toaster projected onto a live video of a real world scenario using Vivian. The user can interact with the different 3D models on the screen. Buttons can be pressed by simply tapping the screen where the respective button is displayed. Knobs and levers can be moved by putting a finger on the screen where the knob or lever is displayed, moving the finger on the screen until finding the desired position, and removing the finger from the screen afterwards. We refer to this gesture as *drag-and-drop*. The Vivian framework makes a clear distinction between movable and non-movable objects in a scene. The slice of bread in the aforementioned example is movable, whereas the toaster itself cannot be moved. Movable objects can be moved in the same fashion as a lever. To cover greater distances, one can move the device itself while dragging the object. Rotation similarly requires the user to hold their finger on the object and rotate the device until they find the desired orientation. The limit of one-finger-interactions exists due to the peculiarity of AR environments where two-finger-interactions (like a pinch-zoom or a two-finger-rotation) would be contradicting with the intended design of such an environment. If one wants to take a closer look at the previously mentioned toaster for example, they would need to step closer to the toaster.

To evaluate different types of feedback, we decided to use usability testing as a tool to quantify the corresponding results. According to Nielsen, *"usability is a quality attribute that assesses how easy user interfaces are to use. The word 'usability' also refers to methods for improving ease-of-use during the design process."* [3]. While usability testing refers to a *"observational methodology to uncover problems and opportunities in designs"* with the goal to identify problems in the design of the product or service, uncover opportunities to improve and learn about the target user's behavior and preferences [4]. Therefore we chose to have the users fill out a System Usability Scale (SUS) questionnaire which we tailored to fit our needs. The basic SUS consists of ten questions regarding a systems usability. We added a few questions which target the feedback more as well as an open feedback item, where the participants were able to share their thoughts on the system with us.

Related Work

The topic of error handling in AR-applications has not been broadly studied yet. There are several guidelines on how to handle wrong user input. Microsoft states in its design guidelines, that error messages should alert users of an already occurred problem. Another point they make is that the message should be suppressed if it does not make the users change their behavior or perform an action [5]. In addition to that, Nielsen's *Error Message Guidelines* demand error messages to be phrased politely. This is to avoid the implication that the user did something stupid or inherently wrong. It is also important that the messages are precise descriptions of exact problems, offering constructive advice on how to fix the problem [6]. A 2002 IEEE article

discusses two approaches of how user help might be implemented in AR environments. What they did was to offer an icon which the users might activate themselves when they seek advice for a given task. The other feature was an automated help message that pops up when the user moves an object associated to a certain task closer to his face and tilts it. They state, that *"*[t]*his approach is more suiTable for AR interfaces than traditional desktop help systems, which either distract users with a constant barrage of help messages or interrupt their work by making them search explicitly for help."* [7].

Approach

To make users quickly and easily understand the controls of mobile AR apps, we extended the Vivian framework to provide three feedback message implementations. Each implementation's messages differ in content, colors and size (see Figure 1). In this section, we will first describe the most common non-functional user input gestures made on the touch screen (as opposed to the gestures that were described in Foundations). Next, we present the exact content of each feedback message and the conditions under which they appear for each set of messages.

Common Misconceptions in Mobile AR User Input

In order to find feedback in response to incorrect usage of a mobile AR application, we have to define what kinds of gestures we consider incorrect in the first place. The user interface provided by the Vivian framework is based on one finger inputs (see Foundations), therefore we should take into consideration each input that is not done with a single finger. Prominent examples for multiple finger gestures are the *pinch-zoom* and the *two-finger rotation*. The pinch-zoom is a gesture in which two fingers are placed on the touch screen and moved apart in opposite directions. The two-finger rotation is a gesture in which two fingers are placed on the touch screen and are together moved clockwise or counterclockwise around the center of the positions of the fingers. A two-finger rotation can also be executed by only moving one of the fingers clockwise or counterclockwise around the other finger. Common misconceptions by inexperienced mobile AR users might include using said pinch-zoom to either zoom in on objects or to bring them closer, or using two-finger rotation to rotate objects.

Feedback Message Implementations

The content of the first feedback message implementation is a *critique*. If the user inputs a gesture that we consider incorrect, the app outputs a message that said input is not possible. For example, if a user tried pinch-zooming, they would receive a message saying: "Zooming is not possible" (see Figure 1, A). The content of the second feedback message implementation is *critique* and *support*. We therefore refer to this feedback type as *combined* in the course of this paper. The user is provided with a message stating that above mentioned input is not possible and given a hint on what they should try instead. In the scope of the previous example, such a message would say: "Zooming is not possible. Try moving around the object" (see Figure 1, B).

In the first two implementations, the messages have the same size and color scheme. The third implementation is more concise *support*. For our previous example the exact message is saying: "Step closer to enlarge on-screen objects". It also differs in color scheme and size (see Figure 1, C). Each feedback message in every implementation is displayed for 5 seconds.

	Description	**Gesture (input)**	**Response (output)**
#1	critique	two-finger rotation	Rotating the object is not possible.
		pinch-zoom	Zooming is not possible.
#2	combined	two-finger rotation	Rotating the object is not possible. Try moving around the object.
		pinch-zoom	Zooming is not possible. Try moving closer to the object.
#3	support	two-finger rotation (movable object)	Hold the object and move the phone to rotate
		two-finger rotation (elsewhere)	Move around the objects
		pinch-zoom	Step closer to enlarge on-screen objects

Table 1: The responses for each input in each implementation.

The contents of each message can be found in Table 1. The description column holds the names we assigned to the three different feedback implementations. In the gesture and response columns you can find the corresponding inputs and outputs respectively. If for example a user tried a two-finger rotation in the critique implementation, the app would display a message saying "Rotating the object is not possible". In the support implementation we made the distinction between two-finger rotations on movable objects and elsewhere. If the aforementioned center point of the two-finger rotation is placed on a movable object (see Foundations), the app displays the message that corresponds to "two-finger rotation (movable object)". If the two-finger rotation is executed elsewhere, the "two-finger rotation (elsewhere)" message is displayed instead. We recognize user input using a software library called TouchKit. TouchKit provides an easy way to detect gestures such as our two-finger rotations and pinch-zooms. For two-finger rotations, we set a tiny minimum angle threshold. If the two-finger rotation's angle is smaller than this threshold, it is ignored and no message is provided. Likewise for pinch-zoom, we set a minimum distance threshold. To further ensure clear distinction of the gestures, we disable the pinch-zoom recognition while two-finger rotation is currently recognized and vice versa.

Case Study

To evaluate the quality of our feedback implementations we conducted a case study in the domain of usability engineering, which is presented in this section. In the section "Setup of the Case Study", we describe in detail what each participant in the case study had to do and what data we collected. The results of the study are found in Results of the Case Study and afterwards discussed in Discussion of the Case Study Results.

Setup of the Case Study

In order to evaluate, which variant in our approach works best, we created two different prototypes with the Vivian framework (see Foundations). Both of them are quite simple kitchen devices: a toaster prototype and a microwave prototype (as seen in Figure 1).

The microwave's functionality is limited to heating an object inside it with constant power. Figure 1 C shows this prototype in an application. It has one button to add 10 seconds to the heating duration and another one below to open the door. The door can be closed by moving it. The status of the microwave is visually indicated by a small display showing the remaining heating time or "ready" in idle mode and a light inside the device, which turns on when it is heating or when the door is opened. A serial number is printed on the back of the microwave. The prototype scene is completed by two additional objects. A cup that is already in the microwave when the application launches and a piece of cake on a plate next to it. These objects are movable by the user.

The toaster can toast one or two pieces of bread at a time. Figure 1 A shows the front of the prototype in an application and Figure 1 B its back. The toasting can be started by pulling down a lever at the front and stopped by pushing it up again or by pressing the stop button, which is on the back. Otherwise, the toasting stops automatically after a time which is defined by the position of a rotaTable knob at the front. The time mode is divided into low, medium and high duration. Further functionality is provided by the unfreezing mode, which can be activated using the snowflake button on the back. The activation of the unfreezing mode results in a longer toasting duration and is indicated by a light above the button. The toasting process itself is displayed by the glowing of the heating elements, which is visible in Figure 1-A. The toaster prototype also has a serial number on its back. The prototype scene is completed by a stack of 3 pieces of bread lying next to it.

Based on these functionalities we designed three tasks per prototype, which are shown in Table 2. The tasks are increasing in complexity. An example of this increase is that in task 2 of the microwave we ask to heat the cup, which is already in the microwave, for any amount of time. Therefore, only the start button has to be pressed once to fulfill this task. Afterwards, in task 3, we ask to replace the cup with the pie and heat for a specific amount of time. This task contains more necessary steps: opening the door, moving the cup out, moving the pie in, closing the door, pressing the start button multiple times and finally pressing the stop button after the desired amount of time.

#	**Microwave**	**Toaster**
1	Read serial number of the microwave	Read serial number of the toaster
2	Heat up the cup	Toast the bread
3	Remove the cup, put the pie in, set the timer to 20 seconds and remove the plate at 5 seconds	Toast the bread on high heat and put the toaster in unfreezing mode

Table 2: The tasks for the different prototypes.

For the case study, we divided the participants into four groups. Each of these groups tested one version of the app on a mobile phone. The versions differed in the feedback given (see Approach). We gave no feedback at all to the participants in group 0, which is our control group. Participants in group 1 got our first implementation with *critique* feedback. In group 2 they got our second implementation with a *combined* feedback version and in group 3 they got our third implementation with *supportive* feedback (see Table 1). Each participant was asked to fulfill the tasks for each prototype in the order provided in Table 2. One half of the participants were asked to do the tasks of the toaster prototype first, while the other half did the tasks for the microwave prototype first. This divide was equally for all groups. We were able to evaluate subgroups with a certain implementation level and a certain order of prototypes they tested. In the following, the subgroups are named *TM*, which translates to "first toaster, then microwave" and *MT*, which translates to "first microwave then toaster", followed by the number of their feedback group. For example, a participant who was assigned the toaster first and then the microwave without any feedback was in the subgroup TM0.

Before the start of the test, we gave every participant a short introduction. This introduction contains information about the mobile AR technology and reasons behind usability testing to prepare the participants and create a similar level of expectation. It deliberately does not contain information about how to use the application or the scope of the case study. This is important so the participants are not biased about wrong usage and the feedback they would get. During the test, we used screen recordings to measure each participants' use of the application. These enabled us to acquire completion times for the different tasks and count the number of incorrect usages. We used a notepad to register further observations.

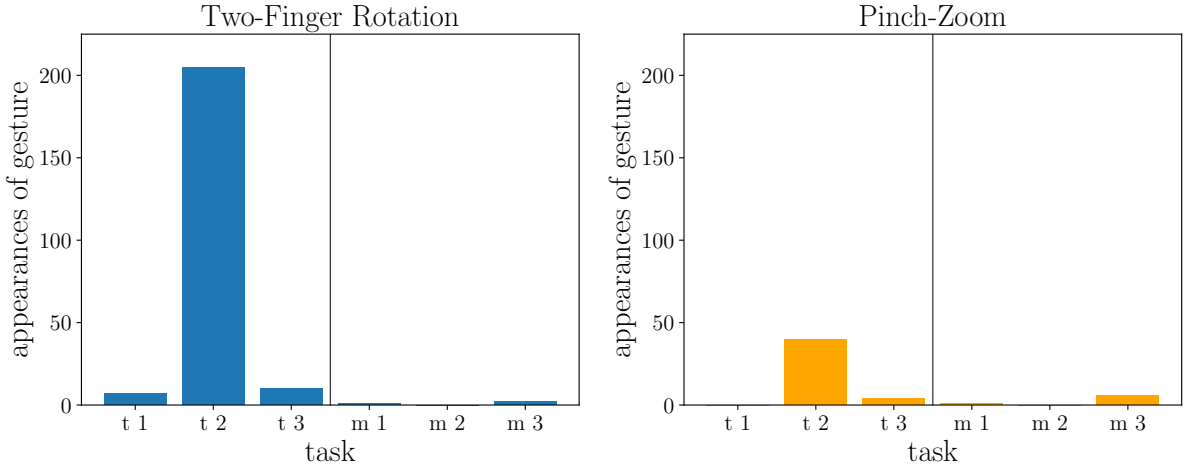
Following the test of the application, the participants were asked to fill in a questionnaire. This questionnaire consists of 3 parts. First we asked participants if they had used AR applications before. The second part is an SUS (see Foundations) about the usability of the AR application, that we tailored for our needs. The third part is an open question directly aimed to the feedback provided by the application, namely: *"If you have any suggestions on what kind of feedback from the app you would find better, please let us know"*.

Results of the Case Study

We conducted the case study with 10 participants per feedback group. This leads to 40 people participating in the case study in total. The smallest subgroups, defined by implementation and task order, were still filled with 5 participants. 21 of the 40 participants said they had prior experience with AR applications. The group with the highest prior knowledge was the control group with 7 of 10 participants. The groups for the *critique* feedback and the *combined* feedback both had 6 participants with prior knowledge, while the group with the *support* feedback had just 2 participants, who said they had prior knowledge.

In our approach, we consider especially two gestures as incorrect usage: the pinch-zoom and the two-finger rotation (see Approach). With the setup of our case study, the two-finger rotation appeared more often than the pinch-zoom gesture. Figure 2 shows the appearances of

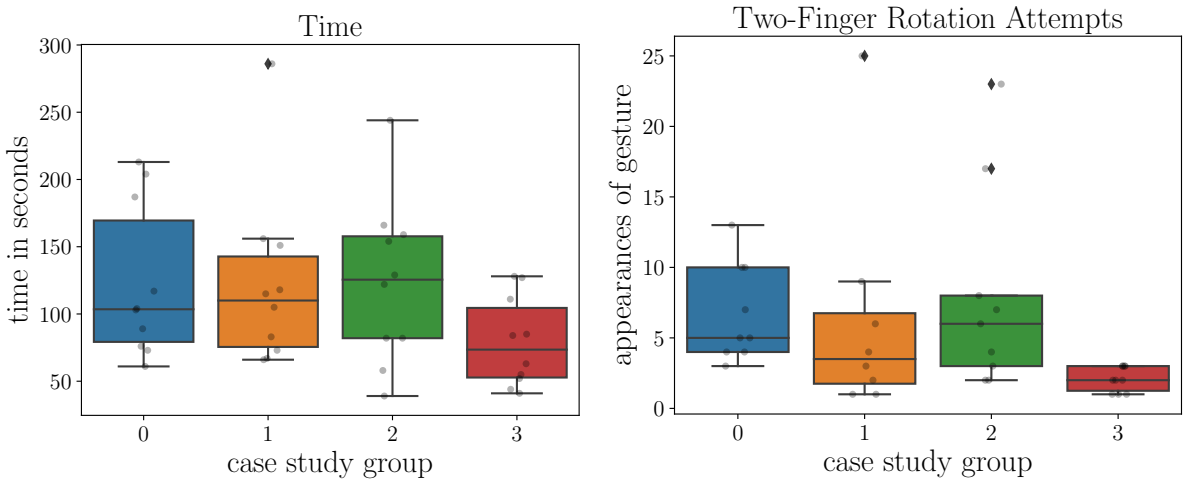
two-finger rotations and pinch-zooms for all tasks given to the participants in the case study. In total over all participants and all tasks, two-finger rotations appeared 225 times, while pinch-zoom appeared 51 times. Since 82% of the incorrect usages are two-finger rotations, the results of our case study focus on those. We especially evaluate the second task for the toaster, since it was most demanding for the users. As shown in Figure 2 A, 91% of the overall two-finger rotations occurred in that task. Also the majority of pinch-zoom gestures occurred there.



(A) **(B)**

Figure 2: Appearances of the two-finger rotation gesture and the pinch-zoom gesture per task of the case study.

To evaluate the effectiveness of the different implementations, we compare the users' behavior for the second task of the toaster, with different measures. Figure 3 shows box plots for the two most important measures. The diagram in Figure 3 A gives the time participants needed to fulfill the task by their case study group. The median is given by the line inside the box. The box itself gives the 25 percentile on its lower bound and the 75 percentile on its upper bound. The whiskers are defined by the last data points which are within the range of 1.5 times the IQR. For clarification the data points are printed semi-transparent over the box plot. Figure 3 B is similar: it shows the two-finger rotation attempts per user for the different case study groups. Important for this graph is that users, who had zero two-finger rotation attempts, are excluded from the statistic. The reason behind it is that these users didn't receive any feedback message independently of their given implementation. To trigger any feedback, an incorrect usage must occur at least once. In total, 4 of the 40 participants didn't try any two-finger rotations on this task. The maximum number for one specific implementation is 2. That means that for each implementation at least 8 participants' data was used to take the statistical values.



(A) **(B)**

Figure 3: Evaluation of the two-finger rotation attempts and time needed for the 2nd task for the toaster prototype.

To check whether the result is statistically significant, we performed statistical tests on both the time needed to complete the task and the number of two-finger rotation attempts. The main test is the *Welch t-test*, that is used to compare the different implementations with the control group, which didn't receive any kind of feedback messages. The *Welch t-test* has the assumption that both populations are normally distributed. We used the *Shapiro-Wilk* test on the residuals between the given implementation group and the zero feedback group to check on the assumption. Generally we consider p-values below 0.05 as statistically significant. Table 3 shows the p-values for the different tests for all three implementations. The *Shapiro-Wilk* tests does not indicate normality for the two-finger rotation metric with the *critique* feedback implementation and the time metric with the *combined* feedback implementation. Since the assumption for the *Welch t-test* is not given, the test cannot be run for these two cases. In all other cases the assumption that the data is normally distributed holds. For the *critique* and the *combined* feedback implementations the test with the other metric did not reveal any significant difference to the control group. For the *support* feedback implementation the result of the *Welch t-test* for the number of two-finger rotation attempts in task 2 of the toaster prototype implies a significant difference to the control group, which however cannot be fully confirmed by our result for the completion times for the task.

statistical test	**implementation**	**p-values: time**	**p-values: two-finger rotation**
Shapiro-Wilk	critique	0.972	0.001
	combined	0.029	0.420
	support	0.507	0.971
Welch t-test	critique	0.980	-
	combined	-	0.681
	support	0.054	0.010

Table 3: p-values of statistical tests for the time needed to fulfill the 2nd task for the toaster prototype and the two-finger rotation attempts in that task.

In addition, we evaluated the relationship between our two metrics for the task. The correlation coefficient between the time and the attempts of two-finger rotation is 0.43, which indicates a moderate positive relationship between the metrics. Figure 4 visualizes the relationship between the time needed for the task and the number of two-finger rotation tries. The color of a data point is set by the implementation used by this participant. One can observe that the participants with implementation 3, which are shown by the red data points, are all clustered with a low time and a low amount of two-finger rotations, while all other groups are more randomly spread.

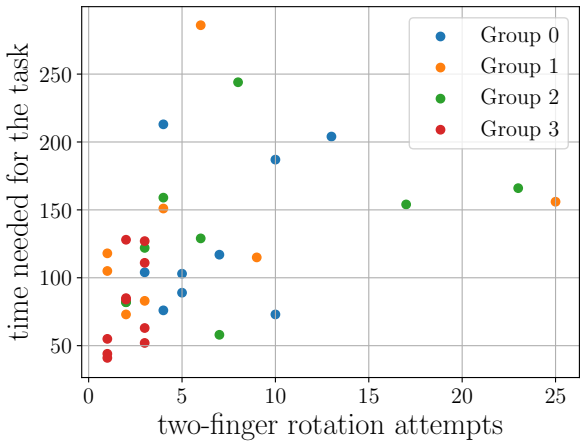


Figure 4: Time needed to fulfill the 2nd task for the toaster prototype depending on the count of two-finger rotation attempts.

For the evaluation of the questionnaire, we calculated the SUS scores for every participant. Figure 5 shows the mean score for each case study group. The horizontal red line is at a score of 68, which in terms of SUS stands for average usability. The Figure shows that the mean SUS result of participants in case study group 3 with the support implementation is slightly higher than the mean of case study groups 2 and 3. The mean SUS score of the control group is the lowest.

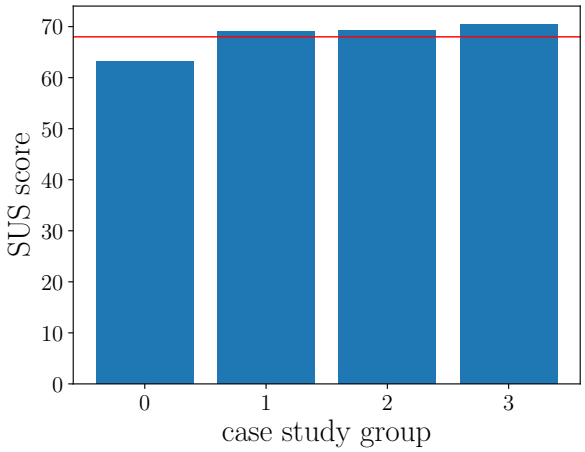


Figure 5: SUS score by the case study groups.

24 of the 40 participants answered the open question (*"If you have any suggestions on what kind of feedback from the app you would find better, please let us know"*). We categorize the answers with four different labels, divided by the topic the answer is about. These are *app feedback*, *app controls*, *prototype design* and *other*. Answers in the *app feedback* category are directed to the provided feedback messages or other desired feedback. Answers in the *app controls* category are about struggles with or improvement ideas for the applications controls. The *prototype* category is related to the prototype design and its usability, for example button sizes. The last category, *other*, is summing up single appearing answers which scope is not relevant for this research. The appearances of the different types of answers are listed in Figure 6. In case an answer fit to multiple labels, we count it for all the fitting categories.

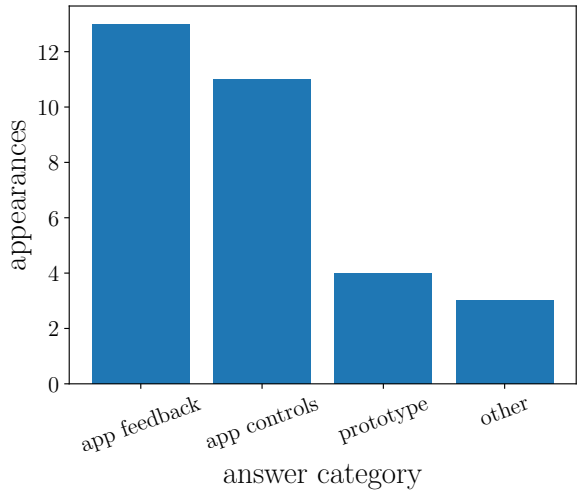


Figure 6: Categorized open question answers.

For a deeper evaluation of the answers we split them by the case study groups (see Figure 7). Most of the answers related to *app feedback* came from participants in the control group. All of this answers were asking for any kind of help. Two of them directly asked for text messages

providing help. Also the ideas of a help button or an introduction, which explains the controls were mentioned. Participants in the other groups also mentioned additional help implementation methods, like vibration feedback or help for the specific task from the application. Answers directed to our feedback message implementations were that they could be hidden by the fingers, when the user turns the phone and one participant with the *combined* feedback mentioned that the message should be visible for a longer duration. The other big category of the participants' answers is *app control* related. They have in common that the rotation of the piece of bread in the second task for the toaster prototype was not easy to archive for them. Also some were mentioning that the two-finger rotation could be the more intuitive option.

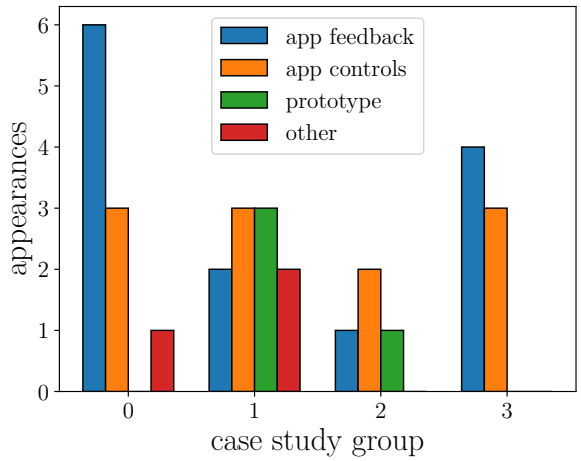


Figure 7: Appearances of answer categories by the participants case study groups.

Discussion of the Case Study Results

With the given setup of our case study, the two-finger rotation gesture appeared more often than the pinch-zoom gesture (see Figure 2). We assume users attempt to use the pinch-zoom gesture when an object on the screen is too small to see or use. Given this, there was no temptation for our participants to pinch-zoom, because they started right in front of the objects. Also, for simplicity's sake, the prototypes were not created with unnecessarily small controls, which means all of them were well visible in general. It is possible that our results for two-finger rotations are applicable in a similar way for pinch-zooms. This would however require further investigation.

We divided the evaluation of the rotation into two groups, as we did for our *supportive* feedback implementation (see Approach). While most of the users intuitively walked around objects to rotate their view on the scene, they struggled when trying to rotate movable objects. There were two occurences of tasks in which an object had to be rotated in order to complete them, namely the second and third toaster tasks (see Table 2). However, our results suggest that once the participants learned how to rotate the pieces of bread in the second toaster task, most of them knew how to accomplish said rotation in the third toaster task (see Figure 2). Therefore we focus on the second toaster task. The difficulty of this task is also represented by

the fact that only 10% of the participants were able to complete the task intuitively without trying a two-finger rotation before. Hence most users could profit from helpful feedback.

The feedback versions we provided performed differently in the case study. While the implementation with *critique* feedback and the implementation with *combined* feedback did not have a significant effect, the users with the *support* feedback were able to fulfill the task in less time and with significantly less attempts of the incorrect two-finger rotation gestures. According to our results, prior knowledge about AR cannot account for this observation, because the *support* feedback group had the lowest percentage of participants with prior AR knowledge. Instead we assume multiple reasons for this difference. One is that the *support* feedback implementation has better visibility than the other two (see Figure 1). While the other two implementations provide typical Android pop-up messages [8], the *support* feedback messages are larger and have an orange background color instead of gray. These changes make the messages more noticeable for the users. This matches our observation during the test: Some of the participants with the *critique* or *combined* ignored the messages, which happened far less for participants with the *support* feedback. The other difference is the helpfulness of the messages' content: The *critique* feedback just tells the users what they have done incorrectly, but not how to do it correctly instead. Since the correct way for rotating objects was not intuitive for most participants, they could not easily figure it out without help. Our *combined* feedback implementation has a similar issue. When a two-finger rotation is detected, a message is provided. In this implemenentation, the content of the message does not depend on whether the rotation was executed on an object that can be moved or not (see Table 1). This indifference causes the message to provide inappropriate support for some situations. This fact concerns the second toaster task. Since the *support* implementation clearly discriminates between two-finger rotations executed on movable and unmovable objects (see Table 1), the feedback messages are specific enough to help in any given situation.

Our data states that participants perform significantly better with the *support* feedback implementation, but we also asked the participants to assess their experience with our AR applications themselves. However, although there was a significant difference in the performance between groups, the results of our SUS questionnaire do not clearly show a difference between the case study groups for the different implementations (see Figure 5). Thus, the performance in the tasks did not influence the SUS score significantly. Furthermore, in the assessment of the usability, other aspects also affected the general experience. Examples include the need to restart the application in case the prototype disappeared when the users accidentally covered the camera with their hand, the simplistic graphics of the prototypes which made interactive elements hard to discern for some users, or the differences in lighting between the test locations. Moreover, the users often referred to the overall application in the open question, although they were asked to evaluate the feedback the applications provided (or did not provide in the case of the control group). This suggests that the participants might have often rated the quality of the prototype itself rather than the overall application in the course of the SUS questionnaire, just like they did in the open question. Given this and provided a larger sample size it might however be possible to find a significant, but probably small effect our feedback could have on

the SUS score.

Summary and Outlook

This paper aims at contributing towards improving the overall mobile AR experience of users by providing them the appropriate feedback in case of usage of unsupported gestures. We developed three different sets of feedback messages and conducted a case study to test different tasks with users which were divided into four different groups. Users in different groups received different types of feedback messages. The feedback messages were implemented as an extension to a mobile AR framework called Vivian. The Approach and Case Study sections describe in detail the feedback types and how users were divided into different groups.

In our work, we focused on the pinch-zoom and the two-finger rotation, which are incorrect input gestures described in Common Misconceptions in Mobile AR User Input and therefore not supported by Vivian. According to our findings, users are inclined to use two-finger rotations when they are confronted with a task in which they have to rotate an object in order to complete it. Our results support that appropriate feedback does indeed help users complete such tasks in less time. In our case study, users with appropriate, supportive feedback were completing the first rotation task twice as fast as users without any feedback. Moreover, during the completion of tasks, the number of incorrect gestures for each user with appropriate feedback was three to four times smaller in comparison to the number of incorrect gestures for each user without feedback. Pinch-zoom gestures however were not widely used by our participants, which is most likely due to our test setup (see Discussion of the Case Study Results). The feedback type that we found most effective among the ones we implemented in the course of our case study is the supportive one. The exact content of the messages can be reviewed in Feedback Message Implementations. The messages in that implementation are not only more accurate, they are also provided with the brightest colors and biggest font.

Pinch-zoom gestures are not yet taken into account by our case study setup, which makes it impossible to evaluate the feedback implementation we developed in the scope of this gesture. Further we noticed 17 of our 40 participants were trying to interact with on-screen objects by moving their hand in front of the camera lens of the mobile device instead of using the built-in touch screen. Since our results look promising, we will conduct another case study in the future, where we consider these observations in the case study setup. We will also consider other feedback types with different visibility and content, to discover whether the effect we found is due to one or another.

References

[1] Tractica. *Mobile Augmented Reality Market to Reach 1.9 Billion Unique Monthly Active Users by 2022*. Apr. 2017. URL: https://tractica.omdia.com/newsroom/press-releases/mobile-augmented-reality-market-to-reach-1-9-billion-unique-monthly-active-users-by-2022-according-to-tractica/.

[2] G. Gary Wang. "Definition and Review of Virtual Prototyping". In: *Journal of Computing and Information Science in Engineering* 2.3 (Sept. 2002), pp. 232–236. DOI: 10.1115/1.1526508.

[3] Jakob Nielsen. *Usability 101: Introduction to Usability*. Jan. 2012. URL: https://www.nngroup.com/articles/usability-101-introduction-to-usability/.

[4] Kate Moran. *Usability Testing 101*. Dec. 2019. URL: https://www.nngroup.com/articles/usability-testing-101/.

[5] Microsoft. *Error Messages*. May 2018. URL: https://docs.microsoft.com/de-de/windows/win32/uxguide/mess-error.

[6] Jakob Nielsen. *Error Message Guidelines*. June 2001. URL: https://www.nngroup.com/articles/error-message-guidelines/.

[7] I. Poupyrev et al. "Developing a generic augmented-reality interface". In: *Computer* 35.3 (Mar. 2002), pp. 44–50. DOI: 10.1109/2.989929.

[8] *Android Toast Documentation*. https://developer.android.com/reference/android/widget/Toast. Accessed: March 2020.