ShapelineGuide – Teaching Mid-Air Gesture for Large Interactive Displays

Florian Alt¹ 2, Sabrina Geiger¹ and Wolfgang Höhl¹
¹LMU Munich, ²University of Applied Sciences Munich
{florian.alt, sabrina.geiger, wolfgang.hoehl}@ifi.lmu.de

Figure 1: ShapelineGuide supports users of large interactive displays in performing mid-air gestures: Upon activating the tool (left) users get feedback on the hand(s) to be used (middle) and we provide feedback and feedforward during gesture execution (right).

ABSTRACT
We present ShapelineGuide, a dynamic visual guide that supports users of large interactive displays while performing mid-air gestures. Today, we find many examples of large displays supporting interaction through gestures performed in mid-air. Yet, approaches that support users in learning and executing these gestures are still scarce. Prior approaches require complex setups, are targeted towards the use of 2D gestures, or focus on the initial gestures only. Our work extends state-of-the-art by presenting a feedforward system that provides users constant updates on their gestures. We report on the design and implementation of the approach and present findings from an evaluation of the system in a lab study (N=44), focusing on learning performance, accuracy, and errors. We found that ShapelineGuide helps users with regard to learning the gestures as well as decreases execution times and cognitive load.

INTRODUCTION
Large displays have become ubiquitous in our everyday life. We use them in our workplaces to interact with different visualizations but also in public spaces to browse through information or to interact with playful applications [9].

Interaction with such displays can be enabled using a large variety of input modalities, with a notably predominant role for gestures [20, 23, 25], smartphones [1, 2], as well as gaze [14, 15, 28]. Interaction based on mid-air gestures in particular suffers from the fact that today no commonly agreed-upon gesture set exists. As a result, people need to be taught which gestures to use as they interact with such a display.

This challenge has been recognized within the display community. As a result, researchers looked into how the use of gestures can be supported, primarily focusing on point and dwell [19, 25, 26]. To support more complex gestures, Sodhi et al. presented LightGuide [22], where hints on how to perform the gesture are projected on the users’ hands. Yet, this approach is challenging, since it requires a complex hardware setup and users need to constantly shift their attention between the content on the screen as well as the feedback on the gesture. Octopocus provides feedback on the current status of a gesture as well as options as to how a gesture could be further executed (feedforward) [3]. However, the approach was designed for desktop applications using mouse interaction and does not cover the depth dimension.

Drawing from and extending previous approaches, we built ShapelineGuide, a system specifically targeted at supporting the use of mid-air gestures in front of large displays. It combines feedback on the executed gesture as well as feedforward on the possible gestures based on the current status, both integrated with the current display in the form of an overlay. For the gesture recognition we use Kinect, hence allowing gestures to be performed in 3D space.

Results from an evaluation in the lab with 44 participants show that our approach outperforms dynamic help menus. In particular, we better support users in learning gestures, enable significantly faster input, and reduce the cognitive load.

CONTRIBUTION STATEMENT
The contribution of this work is twofold. Firstly, we report on the design and development of ShapelineGuide. Secondly, we present an evaluation of the tool, focusing on how well it supports the process of learning gestures, as well as on how the
tool impacts on the accuracy and speed of executing gestures as well as on the induced cognitive load.

BACKGROUND & RELATED WORK

Our work draws from several strands of prior research, most notably mid-air gestures – in particular, when used in front of large displays – and approaches to support learning them.

Mid-Air Gestures

A number of classifications exist for gestures [4, 5, 13, 27]. A gesture can either be symbolic (i.e. has a unique meaning), deictic (i.e. pointing gestures), iconic (i.e. gestures that provide information on size, form, and orientation of an object) or pantomimic (i.e. a simulating a tool or object) [21].

We primarily focus on deictic gestures in this work, in particular mid-air gestures. We define mid-air gestures as gestures that are executed freely in 3D space without necessarily touching the display. Mid-air gestures can be performed with one or multiple parts of the human body – we refer to the latter case as multiple gestures in the remainder of this paper.

Furthermore, discrete and continuous gestures can be distinguished. Whereas discrete gestures trigger an action only as they have been fully executed (for example, printing out a document as the user draws a circle in mid-air), continuous gestures trigger a constant action (for example, controlling the volume of a video showing on the display).

Mid-Air Gestures and Large (Public) Displays

Researchers looked at how mid-air gestures can be employed for large displays. Chen et al. investigated users’ ability to perform scaling, rotation, and translation tasks using mid-air gestures to modify a 3D virtual car [6]. Furthermore, ergonomic issues of using mid-air gestures have been investigated [7].

Relatively little is known on how to teach users gestures in front of large displays. Walter et al. showed how to communicate a single gesture [25]. Furthermore it was shown, that user representations can be used to first attract attention of users and then switch to the use of gestures [26]. Maekelae et al. investigated the concurrent use by multiple users by means of a cursor [19]. However, the aforementioned projects all employ point and dwell or teach a single gestures, rather than focusing on how a more complex gestures set can be taught.

Approaches to Teaching Gestures

Due to the lack of standardization, the use of mid-air gestures leads to challenges in the real world [8]. After being informed that gestures are available the three major challenges are [12]: How to start with the gesture input? What path of movement to follow? What action is associated with this gesture command?

To facilitate gesture-based interaction, it is important to address these questions during the conception of a system. In general, there are two basic mechanisms to support users with the gesture interaction: feedforward and feedback mechanisms.

Feedforward Mechanisms

Feedforward mechanisms give user an indication of the gestures to be executed. Before a gesture is executed, the feedforward mechanism provides relevant information, such as the next possible executable gesture, visualized subsegments of a gesture, or the complete visualized movement path. In general, the feedforward mechanism is characterized by two major characteristics: the level of detail presented to the user and the update rate of these details [3].

Kurtenbach et al. [16] applied this feedforward mechanism to their marking menu in the form of text labels. Another possibility to teach users available gestures are contextual animations [17]. Through the use of animations and accompanying texts, the gestures are demonstrated to the user. Grossmann et al. [10] presented the hover widgets application. While users moves their input pen over a graphic tablet, they get a continuous overview of all possible gestures.

Feedback Mechanisms

Feedback mechanisms provide information about the current status during the execution of a gesture, e.g., status of the executed movement path, error messages on incorrect movements, or current position on a movement path [3].

Igarashi et al. [11] developed pegasus, a system which is based on the interactive beautification technique. In general, a user can draw geometric shapes with an input pen. Subsequently, the drawn shape is examined for predefined conditions and dependencies. Ultimately, the system draws the embellished geometric shape on the basis of newly calculated coordinates. In sum, it gives feedback about the systems interpretation of the gesture and the user is able to choose among alternatives.

The GDT class window from Long et al. [18] gives the user feedback through a numeric value describing how well the gesture was recognized.

Dynamic Guides

An approach to combine feedback and feedforward mechanisms is dynamic guides. In this way, continuously updated information on the gesture execution is provided.

An example is OctoPocus [3], which helps users learn, execute, and remember gestures. Starting from the mouse position, colored lines, representing the partial segments of the gestures, are displayed. To execute a gesture, the user should follow the line, representing the movement path. The color of the line is adapted based on the mouse movement.

Sodhi et al. [22] proposed LightGuide, a dynamic guide for mid-air hand gestures. Through visual hints the user is supported in learning and executing mid-air gestures. With the help of an overhead projector these visual hints are projected onto the user’s body in the form of light.

Due to the input modality, OctoPocus only covers 2D gestures. In case of mid-air gestures, a third dimension is needed. LightGuide achieves this, but focuses on simple gestures, which can only be performed by one body part. One of our goals is to take advantage of OctoPocus and LightGuide and extend it for use with large displays.

SHAPELINEGUIDE

In the following we report on the design of ShapelineGuide, focusing on the requirements, concept and implementation.
With ShapeLineGuide, we combine the advantages of OctoPocus and LightGuide (see Figure 2) and apply them to public displays. Like both systems, we use visual hints, implemented as graphical components, such as shape and color. Like OctoPocus, we show these hints on the display to avoid attention switches for the user. In addition, we allow gestures to be supported in 3D space, which is not possible for OctoPocus that was designed for use on desktops using the mouse.

In addition, three functional requirements are necessary to interact with the ShapelineGuide. The user should be able to choose a gesture from the existing gesture set. To trigger the action of the chosen gesture it must be possible to execute the gesture. If users have begun the gesture execution, but want to cancel it, they should be able to abort the action.

**Concept**

To optimally support users while executing mid-air gestures, the concept integrates continuous feedback on the gesture execution progresses and feedforward. Though in the current implementation we focus on using the hands, in general the developed concept can be extended to further body parts.

**Basic visualization of gestures.** The gestures are represented by geometrical shapes arranged next to each other. The starting point of the gesture is communicated to the user through a fully saturated shape. Finally, a text label is used to make clear to the user which action the respective gesture triggers.

**Structuredness.** Visualizing many gestures can lead to visual clutter. To avoid this, we visualize gestures as geometric shape, including a text label, that can be selected like a button. By hovering with the right body part over the button, the whole gesture visualization is displayed. The positions of the gesture buttons executed with the left hand, are placed on the left display border and the gestures executed with the right hand on the right display border.

**Affiliation.** In the beginning it is important to know with which body part a gesture is executed. Our concept exploits the phenomenon of preattentive perception by assigning each body part a particular color, used in the gesture visualization (e.g., all gestures executed with the left hand are blue).
We conducted a study to find out whether the dynamic feedforward and feedback provided by ShapelinGuide helps users in learning, executing, and remembering a particular gesture set. To recognize gestures we use Kinect and computer vision techniques.

**Gesture Builder**
To create a gesture set the Kinect Studio was used in our work. Kinect Studio is an application with which sensor data can be recorded and monitored. All used gestures are recorded and saved as a file with the help of Kinect Studio. Accordingly, it is possible to create individually gesture sets, suitable to the corresponding system.

**Gesture Recognition**
The recorded gesture files are used by the Visual Gesture Builder for the gesture recognition as well. Its main task is to generate data, that can be used by the ShapelineGuide at runtime, to detect mid-air gestures. This gesture recognition is based on the approach of machine learning. In particular all recorded files are added to the Visual Gesture Builder. Afterwards each file is tagged with different values, which describe if the gesture is executed at the moment, or not. This tagging can be done with discrete values or continuous values, depending on the gesture. After finishing the tagging, a database, based on the files, is build. Finally, this database should be integrated in the implementation, to recognize gestures.

**EVALUATION**
We conducted a study to find out whether the dynamic feedforward and feedback provided by ShapelinGuide helps users in learning, executing, and remembering a particular gesture set. In particular, we compare the ShapelineGuide to a standard help menu, placed on the left display border. As users activate the help menu, all gestures are displayed. The movement path is communicated through an animation.

**Hypotheses**
During the evaluation, we tested the following hypotheses:

- **Hypothesis 1**: The error rate is lower for users of the ShapelineGuide.
- **Hypothesis 2**: The execution time to successfully complete a task is shorter for users of the ShapelineGuide.
- **Hypothesis 3**: Users of the ShapelineGuide learn the mid-air gestures faster than users of the standard help menu.
- **Hypothesis 4**: The usability is rated better by users of the ShapelineGuide than by users of the standard help menu.
- **Hypothesis 5**: Users of the ShapelineGuide have a lower cognitive load than users of the standard help menu.
- **Hypothesis 6**: The user experience is rated better by ShapelineGuide users than by standard help menu users.

**Study Design**
To evaluate the mentioned hypotheses a *true experiment* was designed. To minimize external influences, the experiment was carried out in the lab. To avoid learning effects, we opted for a between subject design.

**Procedure**
Participants were randomly assigned to one of two groups. The first group was completing the experiment with the support of ShapelineGuide, the second one with the support of the standard help menu. The procedure is based on the pre-test/post-test design, which is a standard for evaluating learnability. In general, the experiment is structured in five phases: pre-test, training, post-test, questionnaires and debriefing.

**Pre-test.** Before the training phase, all participants were introduced to the existing commands. Afterwards they should guess a mid-air gesture for each of the introduced commands. The phase was used to determine whether the participants intuitively guessed the gestures and thus had advantages in the execution of the mid-air gestures in the experiment.

**Training.** Depending on the different groups, the participant should successfully execute ten given tasks with the support of the ShapelineGuide or the standard help menu. The ten tasks consist of five commands. Each of the commands was be repeated. To start the training, a displayed grey circle had to be activated by the participant. After activating the training, the first task is displayed. Participants could now decide for themselves whether he or she was using the help or not. One task ended when the user had successfully executed the task. After completing the task, the confirmation and a new task were displayed to the user.

**Questionnaires.** Participants filled in three questionnaires on usability, user experience, and the cognitive load.

**Post-test.** In this phase, the data for evaluating the learnability was collected. Each participant was asked to perform the given commands and the corresponding mid-air gesture. The correct gesture executions were counted and documented.

**Debriefing.** After the participants finished the post-test they were asked to provide feedback in general, including remarks, advantages or disadvantages of the used system.

**Sample Application**
For the study, we gave the participants the task to manipulate the 3D model of an iPhone [24]. Via a defined gesture set the user can interact with the model: *rotation to the left, rotation to the right, zooming/scaling, activating the details of the iPhone and a color change*.

**Tasks**
Within the training phase, every participant had to solve ten tasks. Overall, these ten tasks consists of five individual tasks, where each of these five tasks was repeated once. Repetitions were later used to assess learnability.

1. **Rotate** the iPhone to the left.
2. **Rotate** the iPhone to the right.
3. **Scale** the iPhone to the maximum.

![Figure 4: Icons used by the ShapelineGuide to provide feedback to the users.](image)
4. **Activate** the **details** of the iPhone.
5. **Change** the **color** of the iPhone.
6. **Activate** the **details** of the iPhone.
7. **Rotate** the iPhone to the **left**.
8. **Scale** the iPhone to the **maximum**.
9. **Rotate** the iPhone to the **right**.
10. **Change** the **color** of the iPhone.

**Gesture Set**

The gesture set used in this experiment is shown in Figure 5. We chose arbitrary gestures to avoid any bias from users being familiar with particular gestures. The first gesture is used for rotating the iPhone to the left. The second gesture is for activating the iPhone details. Both of these gestures are executable with the left hand. The third and fourth are executed with the right hand. The rotation to the right is symmetrical to the first one. To change the color of the iPhone, gesture four has to be executed. Finally, gesture number five, triggers scaling and needs to be executed with both hands.

![Figure 5: Gesture set](image)

**Data Collection**

During the experiment, we logged execution times. In particular, we measured the **overall input time**, defined as the time period from the start of the task to the successful completion of the task. The **pure input time** measures the time, as soon as a participant is accessing the help, until the task is completed.

To assess learnability, we counted the **number of correctly executed gestures**.

Furthermore, we counted the **number of errors**, whereas each task is either rated as correct or incorrect. Correct means that the participant has executed the requested gesture and the system has successfully recognized it. An incorrect gesture execution can be explained by the following errors:

- **Starting point**: The participant starts at the wrong position to execute the gesture.
- **Body part**: The participant uses the wrong body part to execute the gesture.
- **Number of body parts**: The participant uses the wrong number of body parts to perform the gesture.
- **Gesture**: The participant executes a gesture that is present in the gesture set, but was not required, or the participant executes a gesture, which is not present in the gesture set.
- **Execution**: The participant performs the required gesture incorrectly, which is why it is not recognized.

We count for each task, how many errors the participant makes, until the task is completed. For each incorrect trial, the specific error was recorded. Participants had three minutes to execute each task. If they were not successful, the experimenter would step in and the attempt be counted as incorrect.

After the training, participants completed three questionnaires: the UEQ to assess **user experience**, the SUS to measure **usability**, and the short NASA TLX to evaluate **cognitive load**.

**Apparatus**

The experiment was conducted on a 42” Samsung LED Display with an aspect ratio of 16:9. The screen resolution was 1920 × 1080 pixel. As sensor to recognize and edit the mid-air gesture interaction we used Microsoft’s Kinect One. The systems ran on a Lenovo IdeaPad U310 running Windows 10. The software was implemented in C# and Unity.

**RESULTS**

In the following section, we report on the results of our experiment, ordered by the different phases.

**Participants**

We recruited 44 participants (18 males, 26 females) aged between 20 and 70 years. None of the participants had previous experience with using mid-air gestures before. Participants were randomly distributed to the different conditions.

**Pre-test**

At the beginning, all participants were asked which gesture they would intuitively associate with the available actions. Since the selected gestures are intentionally more complex, no participant could come up with the correct answer. Thus, all participants were eligible for the following phases.

**Training**

Regarding the **number of errors** we found a significant difference between the ShapelineGuide ($M = 0.51, SD = 1.057$) and the standard help menu ($M = 1.72, SD = 2.270$), with $T(279) = -5.834, p < 0.001$. Fewer errors were made when using the ShapelineGuide (see Figure 6). We believe reasons for this result are that users were able to quickly understand if they executed gestures with the wrong body part. Furthermore, users of the ShapelineGuide rarely made a mistake in the category **gesture** and **execution**, probably as a result of them being aware of the movement path of the gestures.
Regarding the overall input time, users of the ShapelineGuide ($M = 20.76 \, s$, $SD = 14.30 \, s$) were significantly faster than the users of the standard help menu ($M = 31.26 \, s$, $SD = 23.11 \, s$) with $T(151.5) = -3.887, p < 0.001$. Figure 7 shows the arithmetic mean values of the input time. From this diagram it can be seen that the median difference is 10.50s. We explain the shorter input time with the fact that users of the help menu tried more gestures before accessing the help menu.

Similarly, the users of the ShapelineGuide ($M = 14.32 \, s$, $SD = 11.83 \, s$) were significantly faster with regard to pure input time than with the standard help menu ($M = 19.00 \, s$, $SD = 17.37 \, s$), with $T(212) = -2.340, p < 0.05$. The mean difference between the input times is 4.68s. The lower pure input time with ShapelineGuide can be explained through the lower error rate.

Questionnaires

The user experience was assessed through the UEQ questionnaire. In the following, the results, structured by the dimensions, efficiency, transparency, controllability, stimulation, originality and attractiveness, are presented. For all these dimensions, the ShapelineGuide was rated significantly better.

Also with regard to usability, the ShapelineGuide was rated better, $T(33.431) = 8.022, p < 0.001$. On average, users rated the ShapelineGuide with 84.89 points, which indicates a very high usability. The average score for the help menu is 68.

Cognitive load was evaluated by the short version of the NASA TLX questionnaire. The evaluation showed that there was a significant difference between the ShapelineGuide ($M = 25.57$, $SD = 12.55$) and the help menu ($M = 47.88$, $SD = 15.55$), with $T(42) = -5.235, p < 0.001$. In particular, frustration was high among the users of the help menu users compared to the ShapelineGuide users. Note, that on average, cognitive load was not too high for both approaches (all values below 50 on a scale from 0–100).

Post-test

The users of the ShapelineGuide were able to reproduce 87 of 110 gestures. With the support of the standard help menu, the participants were able to remember 64 gestures in total. In addition, a significant difference between the ShapelineGuide ($M = 3.95$, $SD = 0.899$) and the standard help menu ($M = 2.91$, $SD = 1.342$) was found ($T(42) = 3.036, p < 0.01$).

In most cases, participants of the standard help menu could remember the movement path, but the wrong hand was used, or the gesture was executed from a wrong starting point. Note, that the gestures for the rotation of the iPhone was learned fastest in both systems. Although the gesture for the color change was executed last, it was the least remembered gesture.

Debriefing & Observations

Participants reported that with ShapelineGuide, they felt ‘safe’ during the execution of the task, that it was ‘routine’ and ‘fun’. An advantage of the ShapelineGuide was the real-time feedback. This was particularly useful with regard to the size of the gestures. Whereas for the help menu participants sometimes performed gestures in a too subtle manner for the system to recognize, this never happened for the ShapelineGuide.

SUMMARY

Users learn the execution and the gestures itself better with the ShapelineGuide, compared to a standard help menu. The ShapelineGuide also requires less input time and less cognitive load for executing gestures. Especially the on-display learning and the easy-to-understand learning was preferred by the users, compared to the standard help menu.

How do I start with the gesture input? The user knows through the saturation of the visualized gesture path, where to start the input. Furthermore, the correct body part for the execution is communicated with a color code.

What path of movement do I have to follow? The ShapelineGuide shows the full movement path of the gesture, which helps users to learn the gesture execution. With the help of the feedback and feedforward mechanism the visualization of the movement path is updated continuously, regarding the body part movement. An advantage over the standard help menu is, that the size and proportion of a gesture is displayed in a realistic manner.

What action is associated with this gesture command? The use of text labels and icons shows the user, which associated action is triggered, when executing the gesture. By showing the movement path of each possible gesture and associated action, it is easier to learn the gesture command sets.

CONCLUSION & FUTURE WORK

In this work we introduced the ShapelineGuide, a concept to support learning mid-air gestures on large displays. We compared it to standard help menus, finding that ShapelineGuide better supports the learning of gestures and decreases execution time as well as cognitive load.

In the future we plan to apply the concept beyond the lab. In particular, we see large potential of the approach in public space – in particular for situations with many first-time users, for example, in airports or museums. Furthermore, we plan to test the approach with different types of content as well as with further different gesture sets and body parts.

REFERENCES

1. Florian Alt, Thomas Kubitza, Dominik Bial, Firas Zaidan, Markus Ortel, Björn Zurmaar, Tim Lewen,


LightGuide: Projected Visualizations for Hand 
Movement Guidance. In Proceedings of the SIGCHI 
Conference on Human Factors in Computing Systems 
(CHI ’12). ACM, New York, NY, USA, 179–188.

23. Maurice Ten Koppel, Gilles Bailly, Jörg Müller, and 
of Public Displays Can Be Used to Influence Actor-, 
Audience-, and Passer-by Behavior. In Proceedings of the 
SIGCHI Conference on Human Factors in Computing 


StrikeAPose: Revealing Mid-air Gestures on Public 
Displays. In Proceedings of the SIGCHI Conference on 
Human Factors in Computing Systems (CHI ’13). ACM, 
New York, NY, USA, 841–850.

26. Robert Walter, Gilles Bailly, Nina Valkanova, and Jörg 
Müller. 2014. Cunesics: Using Mid-air Gestures to 
Select Items on Interactive Public Displays. In 
Proceedings of the 16th International Conference on 
Human-computer Interaction with Mobile Devices &
#38; Services (MobileHCI ’14). ACM, New York, NY, USA, 
299–308.

27. Jacob O. Wobbrock, Meredith Ringel Morris, and 
Andrew D. Wilson. 2009. User-defined Gestures for 
Surface Computing. In Proceedings of the SIGCHI 
Conference on Human Factors in Computing Systems 

28. Yanxia Zhang, Hans Jörg Müller, Ming Ki Chong, 
Andreas Bulling, and Hans Gellersen. 2014. 
GazeHorizon: Enabling Passers-by to Interact with Public 
Displays by Gaze. In Proc. of the 2014 ACM 
International Joint Conference on Pervasive and 
559–563.