http://dx.doi.org/10.14236/ewic/BCSHCI2023.21

Evaluating Graphical User Interfaces for Handling Latency in Remote Crane Operation

Theodor Fahlén
Mälardalen University
Västerås, Sweden
theodorfahlen@gmail.com

Taufik Akbar Sitompul
Norwegian University of Science & Technology
Trondheim, Norway
taufik.a.sitompul@ntnu.no

Rikard Lindell
Mälardalen University
Västerås, Sweden
rikard.lindell@mdu.se

Due to safety and productivity concerns, there are newer cranes that can be operated remotely from a control room. As operators and their cranes are located separately, the communication between operators and their cranes is not free from time delay or latency. The presence of high latency could not only affect operators' capability to work productively and safely, but could also affect their wellbeing due to the feeling of discomfort. In this paper, we propose two types of graphical user interfaces (GUIs) that could support operators to perform their work in the presence of latency. The first GUI visualizes how the crane will move based on the user's input as if the latency is not present, while the second GUI visualizes what kind of inputs that are being executed. We involved 20 participants in an experiment, where they had to move containers under conditions with three latency rates: 0 ms, 500 ms, and 800 ms, while using either of the proposed GUIs and without having any additional support. However, the results suggest that none of the proposed GUIs were significantly better in terms of performance and user experience than the condition without having any additional support.

graphical user interface, time delay, latency, crane, remote operation

1. INTRODUCTION

Rubber-tired gantry (RTG) cranes or also known as yard cranes (see the left image in Figure 1 for an example) play an important role in the shipping industry, as they are needed for efficient container handling. Although the majority of RTG cranes is still controlled by operators who work on-site from inside the crane's cabin, there are newer RTG cranes that can be operated remotely by operators working from a control room (Koskinen et al. 2013) (see the right image in Figure 1 for an example of the remote control station). The shift from on-site to remote operation has been driven by the call for higher safety and productivity, as working from the control room protects operators from accidents that may happen and they can also control any cranes within the port from the same control room (Sitompul 2022b).

Despite its apparent benefits, remote crane operation also produces new challenges that do not exist when operators work from inside the crane's cabin (Sitompul 2022a). One of the new challenges is the presence of time delay or latency due to data transmission between operators and their remote cranes (Vaughan and Singhose 2014). Having an acceptable latency is crucial in any kind of remote operation, since high latency could affect operators' capability to perform their work productively (Chen et al. 2007). The presence of high latency could increase the time needed to complete a task due to the "move-and-wait" situation, in which the operator makes one input, and then wait for the feedback to be visible before making another input (Bidwell et al. 2014). The presence of high latency could also affect operators' capability to work safely. For example, Neumeier et al. (2019) reported that people's capability to drive a car remotely started to deteriorate when the latency reached 300 ms, as the deviation from the planned path also increased. Moreover, the presence of high latency could also harm operators' wellbeing. In the study of remote operation of forest machinery, Brunnström et al. (2019) reported that the feeling of discomfort started appearing when the latency exceeded 400 ms.

This paper proposes two graphical user interfaces (GUIs) that could be used for operating remote RTG cranes in the presence of latency. Since operators are separated from their cranes, they have to rely on the video stream and the GUIs shown on their monitors (see the right image of Figure 1 for an example) to perform their work (Karvonen et al.

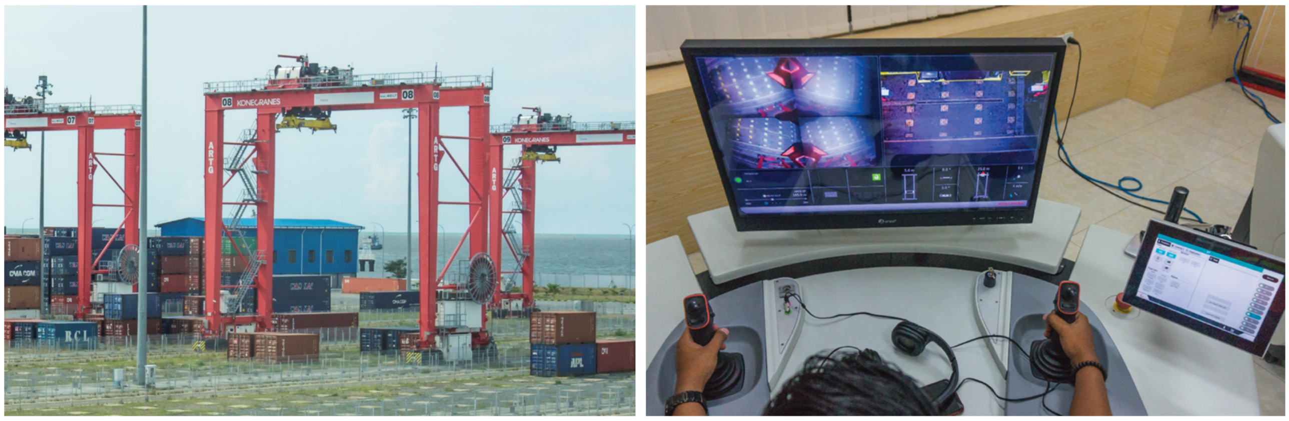

Figure 1: *The left image shows the example of RTG cranes that can be operated remotely. The right image shows the remote control station used by the operator to operate any RTG cranes within one port.*

2012). Note that none of the GUI elements shown in the right image of Figure 1 were designed for handling latency. Considering this situation, we assumed that it would be beneficial to have GUIs that could help operators do their work in the presence of latency.

2. RELATED WORK

As mentioned in Section 1, the data transmission between operators and their remote cranes produces latency (Vaughan and Singhose 2014). There are few prior studies that attempt to mitigate the presence of latency in remote crane operations (see Sitompul (2022a) for the complete review). Since video transmission usually consumes the highest network bandwidth, Villaverde et al. (2012) and Major et al. (2021) proposed to use virtual replicas (or digital twins) of the remote cranes reconstructed based on real-time data from on-site sensors, instead of transmitting the videos. However, both studies did not evaluate how the interaction with the digital twins would support operators' capability to perform their work in the presence of high latency. He et al. (2021) also proposed using a digital twin of a tower crane reconstructed based on real-time data from on-site sensors as an alternative to transmitting the videos. However, the proposed digital twin was designed for monitoring rather than active remote operations. The results of their study suggest that the digital twin enabled their participants to have a shorter response time when detecting hazardous situations, but it is unclear how the digital twin was applicable to active remote operations.

Looking at the studies from other domains, GUIs are typically used for handling the presence of latency by either visualizing the prediction of the near-future state or exposing the latency (Liu et al. 2022). A notable example of predictive GUIs was the "phantom robot" proposed by Bejczy et al. (1990), which visualizes the near-future position of the robotic hand based on the latency. The phantom robot was available in two versions: (1) shaped with solid lines and (2) shaped with dotted lines. In the context of virtual collaborative environments, Chen et al. (2007) proposed the "echo" concept, which visualizes a copy of the object of interest and the distance between the copy and the real object indicates the magnitude of the latency. In the context of lunar-roving vehicles, Matheson et al. (2013) proposed to overlay graphical elements onto the video stream to indicate the position of the vehicle three seconds ahead in the future. In the context of drone operations, Wilde et al. (2020) proposed to overlay information about the drone's flight path and altitude onto the video stream. The overlaid visual information was color-coded, where the green one indicates the current position, while the red one indicates the predicted position.

As mentioned earlier, GUIs can also be used for handling the presence of latency by exposing the latency. In the context of collaborative virtual environments, Gutwin et al. (2004) proposed two approaches to reveal the magnitude of the latency: (1) changing the color gradually and (2) having a "halo effect". In the first approach, the color of the pointer changes gradually from white to black as the latency increases, while the halo effect is visualized with a circle underneath the pointer and the size of the circle gets bigger or smaller depending on the latency. Similar to Gutwin et al. (2004), Shirmohammadi et al. (2005) also proposed two approaches that use different colors to visualize the magnitude of the latency. The first approach was a 2-color scheme of red or black, which is respectively shown whether the latency is present or absent. The second approach was a multi-color approach, which uses different colors ranging from green to red to indicate low to high latency. In the context of virtual

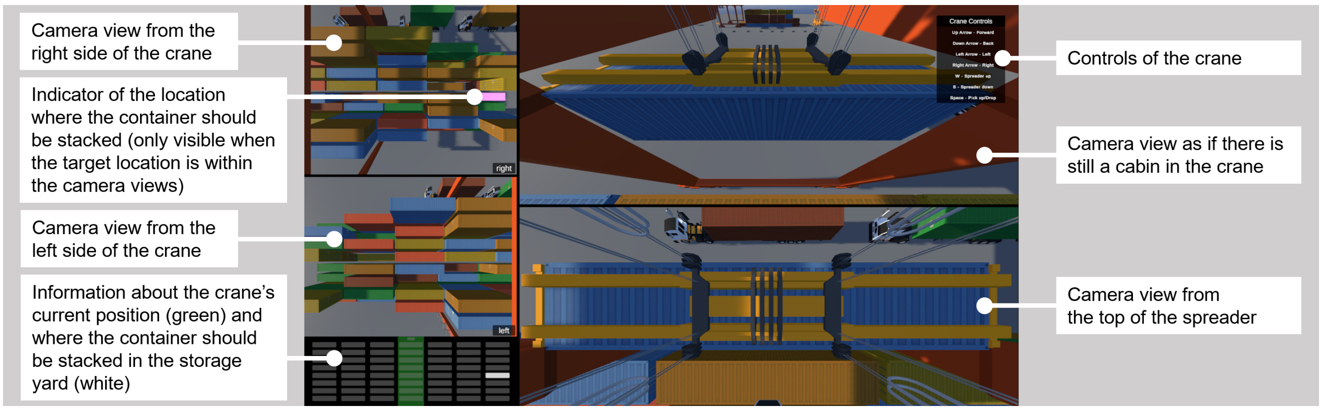
Camera view from the right side of the crane
Indicator of the location where the container should be stacked (only visible when the target location is within the camera views)
Camera view from the left side of the crane
Information about the crane's current position (green) and where the container should be stacked in the storage yard (white)
right
left
Crane Controls
Up Arrow - Forward
Down Arrow - Back
Left Arrow - Left
Right Arrow - Right
W - Spreader up
S - Spreader down
Space - Pick up/Drop
Controls of the crane
Camera view as if there is still a cabin in the crane
Camera view from the top of the spreader

***Figure 2:** An example of the default GUI shown in the simulator. There are four camera views that show different parts of the crane, which also show an indicator of where the container should be stacked (only visible if the target location is within the camera views). The bottom-left part visualizes the crane's current position and the location where the container should be stacked in the storage yard. The top-right part contains information about the buttons used for controlling the crane. The default GUI is also referred to as the blank interface in this paper.*

reality, Fraser et al. (2000) proposed to use a sphere that surrounds the avatar to visualize the magnitude of the latency. The size of the sphere increases as the latency increases, and vice versa.

3. METHOD

This section describes the crane simulator that we used to evaluate the proposed GUIs, the design rationales behind the proposed GUIs, and the procedure that we used for the experiment.

3.1. Overview of the Crane Simulator

The crane simulator used in this study was initially a port simulator developed by Fahlén et al. (2022) using the Unity game engine. The main function of the port simulator was to simulate the workflow of containers based on different port layouts. We modified the port simulator, so that its main function was to simulate the operation of an RTG crane only. The simulator was programmed to simulate the workflow of stacking containers in the yard, as described below:

1. The trucks brought the containers to the yard, one by one, and automatically stopped at the designated place.
2. The user moved the RTG crane so that it could grab and lift the container from the truck. The truck immediately left the yard, and then another truck would come automatically.
3. The user lifted and moved the container across the yard, and stacked it at the designated place.
4. After arriving at the designated place, the user released the container and moved toward the next truck.
5. The process was then repeated until the user had lifted and stacked ten containers in the yard. The simulator automatically stopped after the tenth container has been stacked in the yard.

We also modified the simulator to simulate three different rates of latency: 0 ms (no latency), 500 ms, and 800 ms. We simulated the latency by delaying the execution of each input based on the latency duration, since the outcome was the same as if we delay the view of the camera.

The RTG crane in the simulator was controlled by pressing seven different buttons on the keyboard. The user could move the entire crane left and right by respectively pressing the left and right buttons on the keyboard. The user could move the crane's spreader forward and backward by respectively pressing the up and down buttons on the keyboard. The user could also raise and lower the crane's spreader by respectively pressing the "W" and "S" buttons on the keyboard. Finally, the user could grab and release the container by pressing the space bar on the keyboard. To help the user remember the controls of the crane, we also overlaid this information on the user interface (see the top-right part in Figure 2). Note that, to replicate the safety feature of real RTG cranes, the simulator accepted only one input at a time.

3.2. The Graphical User Interfaces

This section describes the three different GUIs that we used in this study.

3.2.1. Blank Interface

The first GUI was also the default GUI shown in the simulator (see Figure 2). The default GUI showed four camera views that show different parts

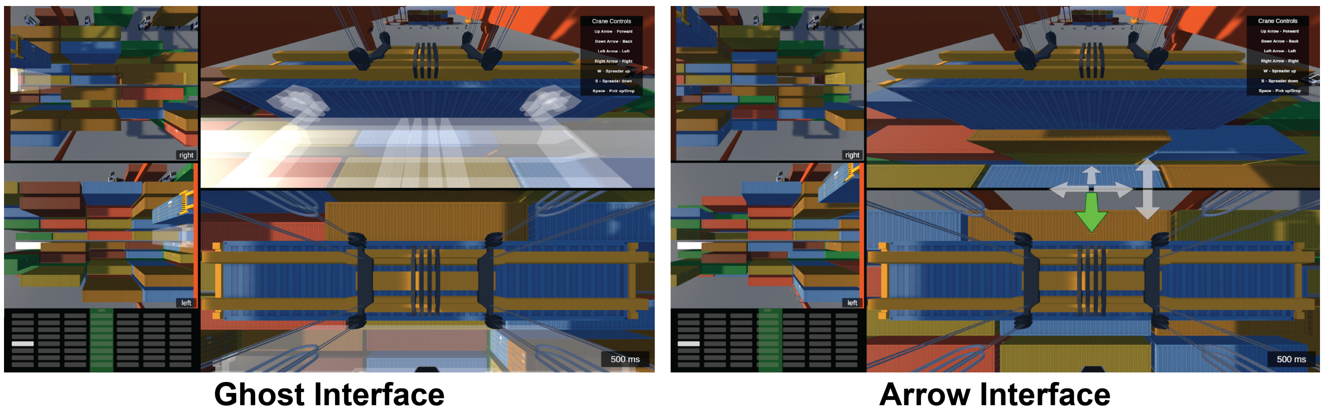

Figure 3: *The left image shows an example of the ghost interface. The ghost interface shows the white semi-transparent replica of the spreader (and also the replica of the container if the spreader grabs a container) in the camera views. The semi-transparent replica moves immediately based on the user's input, regardless of the latency. The right image shows an example of the arrow interface. There are six arrows that correspond to the six possible movements of the crane. One of the arrow is turned green to indicate that such input is currently being executed. Regardless which of the interfaces is used, there is also information about the latency rate in the bottom right, so that the user understands why the inputs are not executed immediately.*

of the crane, information about the crane's current position in the storage yard, information about where the container should be stacked, and information about the buttons used for controlling the crane. As shown in the right image in Figure 1, the GUI for operating remote RTG cranes usually contains more information than what was shown in the simulator. However, we decided to stick with the minimum amount of mandatory information shown in Figure 2, since the focus of the study was evaluating the GUIs for handling latency. The default GUI also served as the baseline to compare with the two proposed GUIs. The default GUI is hereinafter called as the "blank interface", since it does not show particular information that could be used to work in the presence of latency.

3.2.2. Ghost Interface

The second GUI was basically the blank interface that also showed a white semi-transparent replica of the spreader (see the left image in Figure 3). The semi-transparent replica (hereinafter referred to as the "ghost interface") moved immediately based on the user's input, regardless of the latency rates. In other words, the ghost interface provided a prediction of where the spreader would precisely move if there was no latency. Note that the semi-transparent replica of the container was also shown when the spreader grabbed a container. We decided to not show the "ghost" version of the entire RTG crane, since only the spreader and the lifted container that could possibly collide with trucks and containers in the storage yard. Using the GUI classification mentioned in Section 2, the ghost interface belongs to the GUI that visualizes the prediction of the near-future state.

3.2.3. Arrow Interface

The third GUI was basically the blank interface that also visualized six arrows, which corresponded to the possible movements of the crane (see the right image in Figure 3), and thus hereinafter referred to as the "arrow interface". We designed the arrow interface for two purposes: (1) to give a visual cue to the user that the input was not executed immediately if there was latency and (2) to give a visual cue which input that was currently being executed, since one of the arrows would turn green (see the right image in Figure 3 for an example). We decided to place the arrows in the center between the right-side camera views, since this position did not occlude the spreader in any of the camera views. Moreover, by presenting the arrows in the center, it would help the user look at the arrows much easier. Using the GUI classification mentioned in Section 2, the arrow interface belongs to the GUI that visualizes the magnitude of the latency, since one of the arrows turned green according to the latency of the inputs.

3.3. Experimental Procedure

We conducted an experiment to evaluate the proposed GUIs described in Section 3.2 to determine which of them provided the most desirable outcomes. The experiment setup consisted of a desktop computer to run the crane simulator described in Section 3.1, a stand-alone monitor, and a dedicated keyboard to control the crane. This setup was chosen due to its resemblance to typical desktop workstations. Therefore, we assumed that all the participants would be familiar with this kind of setup. As shown in the right image of Figure 1, real remote cranes are operated using joysticks. We decided to use a keyboard instead of using joysticks to operate the virtual crane, as we also expected all the participants were familiar with using keyboards.

The experiment consisted of nine test scenarios, since we wanted to evaluate the three different GUIs with three different rates of latency. Every test scenario contained the same task of stacking ten containers in the storage yard, while using one of the GUIs (blank, ghost, or arrow interfaces) and having one of the latency rates (0 ms, 500 ms, or 800 ms). Note that the latency rate was constant in each test scenario. To reduce the potential training effect, we randomized the order of the test scenarios that should be taken for each participant. In addition, we also mirrored the container arrangement in every scenario, to prevent the participants from remembering the target locations but still keeping the distance required to complete the scenario the same.

Before starting the experiment, we explained to the participants about the objective of this study, the test scenarios that they had to complete, and what kinds of data that we wanted to collect. After the participants provided their informed consent, we collected some background information, such as their age, computer usage, experience with online games, and experience with heavy machinery. After that, we continued with a trial session to allow the participants familiarize themselves with the controls to operate the crane. During the trial session, every participant was asked to stack ten containers while using the blank interface and the latency was 0 ms. The real experiment was started afterward.

We collected three types of data when the participants were completing the test scenarios: (1) completion time, (2) accuracy of container stacking, and (3) number of collisions. The completion time was automatically calculated by the crane simulator from when the crane's spreader touched the first container until the tenth container has been stacked in storage yard. The accuracy of container stacking was defined as the distance between the center of the lifted container and the center of the container underneath. The distance was automatically calculated by the crane simulator when the lifted container touched the target location. The number of collisions was defined as the frequency that the lifted container hit other container(s). The number of collisions was manually counted and classified into three categories, i.e., minor, medium, and severe, based on their severity. Minor collisions were the collisions that happened when the lifted container hit another container, but that container still remained in its place. Medium collisions were the collisions that happened when the lifted container hit another container, and that container fell from its place. Finally, severe collisions are the collisions that happened when the lifted container hit multiple containers simultaneously.

After completing all the test scenarios, we asked the participants to fill in the System Usability Scale (SUS) questionnaire, which is a commonly used questionnaire to document the perceived usability of a product (Lewis 2018). The participants had to fill in the SUS questionnaire for each type of the interfaces. In addition, we also asked the participants about their personal preferences. The participants had to rank the GUIs based from the one that they preferred the most to the one that preferred the least, as well as to describe the reasons behind their decisions. Each participant received a gift card worth around USD 15 for taking part in the experiment.

A total of 20 participants (17 males and 3 females) completed all the test scenarios. The participants were aged between 22 and 37 years old. All the participants reported that they used computers on daily basis. 18 out of 20 participants were familiar with the latency issue due to their experience with playing online games. Four of the participants had experience with operating heavy machinery, such as tractors and forklifts. Note that we did not use any inclusion criteria when recruiting the participants, and thus the participants were recruited based on their availability when we conducted the experiment.

4. RESULTS

The section presents the results from the experiment according to the types of data that were collected.

4.1. Completion Time

As mentioned in Section 3.3, the completion time was automatically counted from when the crane's spreader touched the first container until the participants placed the tenth container in the storage yard. When the latency was 0 ms, the participants required the shortest time to complete the test scenarios when they used the blank interface (*M* = 314.5 s, *SD* = 74.2 s), and then followed by the ghost interface (*M* = 319.2 s, *SD* = 75.1 s) and the arrow interface (*M* = 320 s, *SD* = 68.1 s). A one-way ANOVA suggested that there was no significant difference in terms of completion time between the three interfaces when the latency was 0 ms ($F(2, 57) = 0.033$, $p = .967$).

The results changed when the latency was 500 ms, since the arrow interface produced the shortest time to complete the test scenarios (*M* = 422.5 s, *SD* = 75.7 s), followed by the blank interface (*M* = 424.7 s, *SD* = 86.8 s) and the ghost interface (*M* = 445.9 s, *SD* = 106.7 s). A one-way ANOVA still suggested that there was no significant difference in terms of completion time between the three interfaces when the latency was 500 ms (*F*(2, 57) = 0.407, *p* = .667).

When the latency was 800 ms, the blank interface once again produced the shortest time to complete the test scenarios (*M* = 494.8 s, *SD* = 100.5 s), followed by the ghost interface (*M* = 495.6 s, *SD* = 110.6 s) and the arrow interface (*M* = 515.1 s, *SD* = 87.7 s). However, a one-way ANOVA still suggested that there was no significant difference in terms of completion time between the three interfaces when the latency was 800 ms (*F*(2, 57) = 0.265, *p* = .768).

4.2. Stacking Accuracy

As described in Section 3.3, the accuracy of container stacking is defined as the distance between the center of the lifted container and the center of the container underneath. The distance was measured in the Unity scale, where 1 unit was set to 1 m by default. Note that the accuracy values presented here are for the combined accuracy for stacking ten containers, and not the accuracy for individual containers. In this context, lower numbers represent higher accuracy, since the distance between the center of the two containers is closer, and vice versa.

When the latency was 0 ms, the participants had the highest accuracy when they used the arrow interface (*M* = 3.73 m, *SD* = 1.28 m), followed by the blank interface (*M* = 3.75 m, *SD* = 1.63 m) and the ghost interface (*M* = 3.79 m, *SD* = 1.83 m). A one-way ANOVA suggested that there was no significant difference in terms of accuracy between the three interfaces when the latency was 0 ms (*F*(2, 57) = 0.006, *p* = .993).

The results changed when the latency was 500 ms, since the blank interface produced the highest accuracy (*M* = 4.01 m, *SD* = 1.33 m), followed by the arrow interface (*M* = 4.08 m, *SD* = 1.27 m) and the ghost interface (*M* = 5.17 m, *SD* = 2.62 m). A one-way ANOVA still suggested that there was no significant difference in terms of accuracy between the three interfaces when the latency was 500 ms (*F*(2, 57) = 2.455, *p* = .094).

When the latency was 800 ms, the arrow interface once again produced the highest accuracy (*M* = 4.84 m, *SD* = 2.28 m), followed by the ghost interface (*M* = 4.93 m, *SD* = 2.01 m) and the blank interface (*M* = 5.11 m, *SD* = 2.56 m). However, a one-way ANOVA still suggested that there was no significant difference in accuracy between the three interfaces when the latency was 800 ms (*F*(2, 57) = 0.067, *p* = .934).

4.3. Number of Collisions

As described in Section 3.3, we manually counted how many collisions that happened when the participants were completing the scenarios. In addition, we also classified the collisions based on their severity into three categories: (1) minor, (2) medium, and (3) severe (see Section 3.3 for the description of the collision categories). Figure 4 shows the number of collisions based on their severity, latency rates, and the types of GUIs being used.

4.3.1. Collisions When the Latency Was 0 ms

The participants produced the lowest number of minor collisions when they used the ghost interface (*M* = 1.0, *SD* = 1.1) and the blank interface (*M* = 1.0, *SD* = 1.3), and then followed by the arrow interface (*M* = 1.1, *SD* = 1.3). A one-way ANOVA suggested that there was no significant difference in terms of minor collisions among the three interfaces when the latency was 0 ms (*F*(2, 57) = 0.010, *p* = .989).

In terms of medium collisions, the blank interface produced the lowest number of collisions (*M* = 0.5, *SD* = 0.8), followed by the ghost interface (*M* = 0.6, *SD* = 0.7), and the arrow interface (*M* = 0.8, *SD* = 1.0). A one-way ANOVA still suggested that there was no significant difference in terms of medium collisions among the three interfaces when the latency was 0 ms (*F*(2, 57) = 0.478, *p* = .622).

Regarding severe collisions, the blank interface once again produced the lowest number of collisions (*M* = 0.1, *SD* = 0.2), followed by the ghost interface (*M* = 0.2, *SD* = 0.5) and the arrow interface (*M* = 0.5, *SD* = 0.9). A one-way ANOVA still suggested that there was no significant difference in terms of major collisions among the three interfaces when the latency was 0 ms (*F*(2, 57) = 2.015, *p* = .142).

4.3.2. Collisions When the Latency Was 500 ms

The participants produced the lowest number of minor collisions when they used the blank interface (*M* = 1.3, *SD* = 1.5), followed by the arrow interface (*M* = 1.4, *SD* = 1.4) and the ghost interface (*M* = 1.5, *SD* = 1.1). A one-way ANOVA suggested that there was no significant difference in terms of minor collisions among the three interfaces when the latency was 500 ms (*F*(2, 57) = 0.067, *p* = .934).

In terms of medium collisions, the participants made the lowest number of medium collisions when they used the arrow interface (*M* = 0.8, *SD* = 0.7) and the ghost interface (*M* = 0.8, *SD* = 1.1), and then the blank interface (*M* = 1.1, *SD* = 0.9), A one-way ANOVA still suggested that there was no significant difference in terms of medium collisions among the

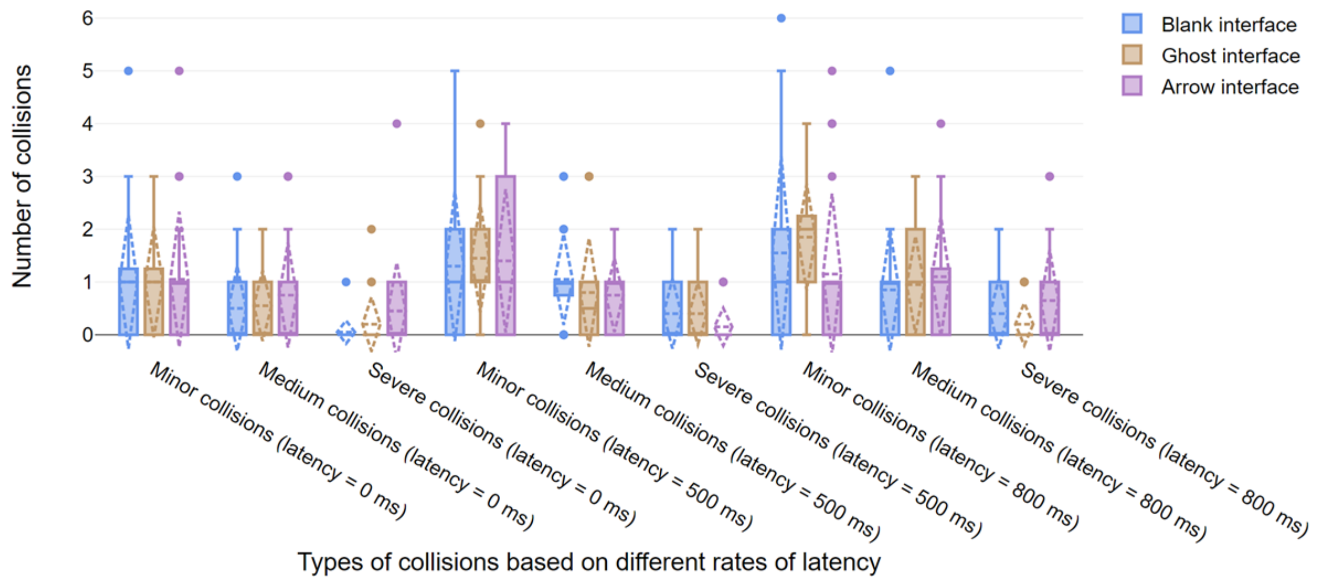

Figure 4: *The collisions are categorized based on their severity and the different rates of latency. The solid lines indicate the medians, while the dashed lines indicate the means. The dashed diamonds indicate the standard deviations and the dots outside the boxes indicate the outliers.*

three interfaces when the latency was 500 ms (*F*(2, 57) = 0.641, *p* = .530).

Regarding severe collisions, the participants also made the lowest number of collisions when they used the arrow interface (*M* = 0.2, *SD* = 0.4), followed by the ghost interface (*M* = 0.4, *SD* = 0.6) and the blank interface (*M* = 0.4, *SD* = 0.7). A one-way ANOVA still suggested that there was no significant difference in terms of major collisions among the three interfaces when the latency was 500 ms (*F*(2, 57) = 1.308, *p* = .278).

4.3.3. Collisions When the Latency Was 800 ms

The participants made the lowest number of minor collisions when they used the arrow interface (*M* = 1.2, *SD* = 1.6), followed by the blank interface (*M* = 1.6, *SD* = 1.9) and the ghost interface (*M* = 1.9, *SD* = 1.0) . A one-way ANOVA suggested that there was no significant difference in terms of minor collisions among the three interfaces when the latency was 800 ms (*F*(2, 57) = 1,048, *p* = .356).

Regarding medium collisions, the participants had the lowest number of collisions when they used the blank interface (*M* = 0.9, *SD* = 1.2), followed by the ghost interface (*M* = 1.0, *SD* = 0.9), and the arrow interface (*M* = 1.1, *SD* = 1.2). A one-way ANOVA still suggested that there was no significant difference in terms of medium collisions among the three interfaces when the latency was 800 ms (*F*(2, 57) = 0.260, *p* = .771).

Lastly, the participants produced the least number of severe collisions when they used the ghost interface (*M* = 0.2, *SD* = 0.4), followed by the blank interface (*M* = 0.4, *SD* = 0.7) and the arrow interface (*M* = 0.6, *SD* = 1.0). However, a one-way ANOVA still suggested that there was no significant difference in terms of major collisions among the three interfaces when the latency was 800 ms (*F*(2, 57) = 1.896, *p* = .159).

4.4. System Usability Scale Scores

After completing all the scenarios, we also asked the participants to fill in the System Usability Scale (SUS) questionnaire to document the perceived usability of the different GUIs. The blank interface received the highest SUS score (*M* = 79.5, *SD* = 13.5), followed by the arrow interface (*M* = 75.9, *SD* = 15.9) and the ghost interface (*M* = 70.5, *SD* = 20). However, a one-way ANOVA suggested there was no significant difference in terms of the SUS scores between the three interfaces (*F*(2, 57) = 1.47, *p* = .238).

4.5. Personal Preferences

As described in Section 3.3, we also asked the participants to rank the different GUIs based on their preferences and describe the reason behind their decisions. As shown in Figure 5, the blank interface was the most preferred one, as 9 out 20 participants chose the blank interface as their first choice. Those participants chose the blank interface as their first choice because they could see the camera views without any obstructions, as indicated by the quote below:

> In my opinion, this one was better. I don't know why, but maybe because I could see all the camera views without any hindrance.

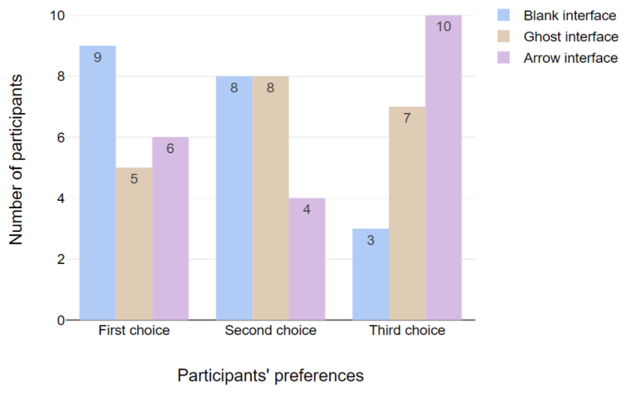

Figure 5: *The ranks of the different GUIs based on the participant's personal preferences.*

As shown in Figure 5, there were 3 out of 20 participants who selected the blank interface as their third choice, which is also the lowest vote for the third choice. The three participants disliked the blank interface because the lack of support made them less confident in the presence of latency, as expressed by the quote below:

> A little bit awkward to use, actually, because you don't have any support. I felt less confident than before.

The ghost interface received the lowest vote as the first choice, since only 5 out of 20 participants chose it as their first choice (see Figure 5). The participants, who selected the ghost interface as their first choice, did so because the ghost interface allowed them to operate the crane without having to wait all the time, as suggested by the quote below:

> I think I would use it frequently. It helps find the direction instead of waiting all the time. So I guess it's good that you have something like a hint or like a direction where the crane is moving.

As shown in Figure 5, there were 7 out of 20 participants who selected the ghost interface as their third choice. Those participants disliked the ghost interface because the presence of the semi-transparent replica of the spreader obstructed their views and they were not able to distinguish the real spreader and its semi-transparent replica, as indicated by the quote below:

> The ghost kind of blocked my view. When I used it, I became confused. When I was focusing on the ghost, I really got confused, mixed up between the ghost and the real spreader, and did wrong things. For me it was a really bad experience to use it.

Regarding the arrow interface, there were 6 out of 21 participants who selected it as their first choice (see Figure 5). Those participants selected the arrow interface as their first choice, since the arrows provided an indication whether their inputs are being executed or still on the way due to the latency, as expressed by the quote below:

> The arrows were, kind of, very necessary to have because otherwise I think I would have made more mistakes because it's difficult to figure out in your head how much delay was 800 milliseconds, for instance. So some kind of feedback on that is useful.

Lastly, as shown in Figure 5, there were 10 out of 20 participants who selected the arrow interface as their third choice, which is also the highest vote for the third choice. Those participants disliked the arrow interface because they did not perceive any advantages from using the arrow interface, as highlighted by the quote below:

> I don't know what I am supposed to use the arrows for. Because what the arrows are showing is already happening. If the crane was going the wrong way, it was already too late, usually, to change the direction. So I couldn't really do anything about what they were showing. So I consider them not important to keep an eye on.

5. DISCUSSION

Based on the results presented from Section 4.1 to Section 4.3, none of the GUIs were significantly better over the others in terms of completion time, stacking accuracy, and number of collisions. In terms of completion time (see Section 4.1), the blank interface produced the shortest time when the latency was 0 ms, but the arrow interface produced the shortest time when the latency was 500 ms. However, When the latency was 800 ms, the blank interface produced the shortest time again. Similar patterns were also found in terms of stacking accuracy (see Section 4.2), since the participants were the most accurate when they used the arrow interface when the latency was 0 ms, the blank interface when the latency was 500 ms, and then the arrow interface again when the latency was 800 ms. Similar patterns can also be observed in terms of the number of collisions. When the latency was 0 ms, the ghost interface caused the least minor collisions, but the blank interface caused the least medium and severe collisions. When the latency was 500 ms, the blank interface produced the least minor collisions, but the arrow interface produced the least medium and major collisions. Lastly, when the latency was 800 ms, the blank interface produced the least minor collisions, the arrow interface produced the least medium collisions, and the ghost interface produced the least severe collisions.

We were surprised with the results of the SUS scores (see Section 4.4) and the personal preferences (see Section 4.5), since the participants did not perceive the presence of the ghost interface positively. Although we did not expect that the ghost interface would produce better results in terms of performance, such as completion time, accuracy, and collisions, we initially assumed that the ghost interface would at least be perceived as more usable and preferred by the majority of the participants. As described in Section 3.2.2, we designed the ghost interface so that the user would be able to predict the crane movement as if there was no latency, since they could continuously operate the crane without having to wait every time they made an input. However, having both the real spreader and its semi-transparent replica was not perceived positively by the participants, since they felt confused with having to focus on two things simultaneously and the presence of the semi-transparent replica also obstructed their views.

6. LIMITATIONS AND FUTURE WORK

As described in Section 3.3, each test scenario in the experiment had a constant latency rate, i.e, 0 ms, 500 ms, or 800 ms. We decided to use constant latency rates, since it enabled us to easily distinguish how specific latency rates affected the participants' performance. However, in practice, the latency in remote crane operation is not constant. Luck et al. (2006) found that people perform differently under conditions with constant and varying latencies. They also found that it is easier for people to adapt when the latency is constant than when the latency is changing continuously. Therefore, it would be interesting to replicate the study presented in this paper under conditions with varying latency.

As mentioned in Section 3.3, none of the participants who took part in the experiment worked as crane operators. In the review of crane-related studies, Sitompul (2022a) notes that the majority of crane-related studies involved non-operators as their participants due to the difficulty in recruiting crane operators. However, in few studies that involved both operators and non-operators, there are some differences in the results between the two groups of participants. However, the results presented in this paper still provide insights into how the different GUIs could affect people's performance and experience under different latency rates. Nevertheless, it would be interesting to replicate this study by involving crane operators to determine to what extent the results presented in this paper remain applicable. In addition, involving crane operators would also help us determine to what extent the ghost interface and the arrow interface are appropriate for remote crane operation.

7. CONCLUSION

This study proposes two types of GUIs that could support crane operators perform their work in the presence of latency. The first GUI provides prediction on how the crane would move based on the real-time input made by the user, while the second GUI visualizes what kind of inputs that are currently being executed. To evaluate both GUIs, we involved 20 participants in an experiment, where they had to move and stack containers while using either of the proposed GUIs and without using any of our GUIs under three different rates of latency, i.e., 0 ms, 500 ms, and 800 ms. The results suggest that none of the proposed GUIs were significantly better than the others in terms of performance and user experience. However, this finding should not be used to diminish the potential of using GUIs to help crane operators perform their work in the presence of latency. Instead, it should be seen that further research is needed to explore how GUIs could help crane operators handle latency in their remote operations.

ACKNOWLEDGMENTS

This research was partially funded by the Department of Design at NTNU and the SFI AutoShip Centre (the Research Council of Norway under project number 309230).

REFERENCES

Bejczy, A., W. Kim, and S. Venema (1990). The phantom robot: predictive displays for teleoperation with time delay. In *Proceedings., IEEE International Conference on Robotics and Automation*, pp. 546–551 vol.1. IEEE.

Bidwell, J., A. Holloway, and S. Davidoff (2014). Measuring operator anticipatory inputs in response to time-delay for teleoperated human-robot interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, New York, NY, USA, pp. 1467–1470. ACM.

Brunnström, K., E. Dima, M. Andersson, M. Sjöström, T. Qureshi, and M. Johanson (2019). Quality of experience of hand controller latency in a virtual reality simulator. *Electronic Imaging 2019*(12), 218:1–218:9.

Chen, J. Y. C., E. C. Haas, and M. J. Barnes (2007). Human performance issues and user interface design for teleoperated robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 37*(6), 1231–1245.

Chen, L., G.-C. Chen, H. Chen, J. March, S. Benford, and Z.-G. Pan (2007). An hci method to improve the human performance reduced by local-lag mechanism. *Interacting with Computers 19*(2), 215–224.

Fahlén, T., M. Drobnjak, J. Finn, I. Hutchings, V. Jakovljevic, R. Nilsson, J. Oyola, and M. Savic (2022). Graphical port simulator. Technical report, Mälardalen University, Västerås, Sweden. Unpublished.

Fraser, M., T. Glover, I. Vaghi, S. Benford, C. Greenhalgh, J. Hindmarsh, and C. Heath (2000). Revealing the realities of collaborative virtual reality. In *Proceedings of the Third International Conference on Collaborative Virtual Environments*, CVE '00, New York, NY, USA, pp. 29–37. Association for Computing Machinery.

Gutwin, C., S. Benford, J. Dyck, M. Fraser, I. Vaghi, and C. Greenhalgh (2004). Revealing delay in collaborative environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, New York, NY, USA, pp. 503–510. ACM.

He, F., S. K. Ong, and A. Y. C. Nee (2021). An integrated mobile augmented reality digital twin monitoring system. *Computers 10*(8), 99.

Karvonen, H., H. Koskinen, and J. Haggrén (2012). Enhancing the user experience of the crane operator: Comparing work demands in two operational settings. In *Proceedings of the 30th European Conference on Cognitive Ergonomics*, ECCE '12, New York, NY, USA, pp. 37–44. ACM.

Koskinen, H., H. Karvonen, and H. Tokkonen (2013). User experience targets as design drivers: A case study on the development of a remote crane operator station. In *Proceedings of the 31st European Conference on Cognitive Ergonomics*, ECCE '13, New York, NY, USA. ACM.

Lewis, J. R. (2018). The system usability scale: Past, present, and future. *International Journal of Human–Computer Interaction 34*(7), 577–590.

Liu, S., X. Xu, and M. Claypool (2022). A survey and taxonomy of latency compensation techniques for network computer games. *ACM Computing Surveys 54*(11s), 1–34.

Luck, J. P., P. L. McDermott, L. Allender, and D. C. Russell (2006). An investigation of real world control of robotic assets under communication latency. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, HRI '06, New York, NY, USA, pp. 202–209. ACM.

Major, P., G. Li, H. Zhang, and H. P. Hildre (2021). Real-time digital twin of research vessel for remote monitoring. In *Proceedings of the 35th European Council for Modeling and Simulation*. ECMS.

Matheson, A., B. Donmez, F. Rehmatullah, P. Jasiobedzki, H.-K. Ng, V. Panwar, and M. Li (2013). The effects of predictive displays on performance in driving tasks with multi-second latency: Aiding tele-operation of lunar rovers. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting 57*(1), 21–25.

Neumeier, S., P. Wintersberger, A.-K. Frison, A. Becher, C. Facchi, and A. Riener (2019). Teleoperation: The holy grail to solve problems of automated driving? Sure, but latency matters. In *Proceedings of the 11th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '19, New York, NY, USA, pp. 186–197. ACM.

Shirmohammadi, S., N. Woo, and S. Alavi (2005). Network lag mitigation methods in collaborative distributed simulations. In *Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems, 2005.*, pp. 244–250.

Sitompul, T. A. (2022a). Human–machine interface for remote crane operation: A review. *Multimodal Technologies and Interaction 6*(6), 45.

Sitompul, T. A. (2022b). The impacts of different work locations and levels of automation on crane operators' experiences: A study in a container terminal in Indonesia. In *Proceedings of the 34th Australian Conference on Human-Computer Interaction*, OzCHI '22, New York, NY, USA, pp. 193–198. ACM.

Vaughan, J. and W. Singhose (2014). The influence of time delay on crane operator performance. In T. Vyhlídal, J.-F. Lafay, and R. Sipahi (Eds.), *Delay Systems: From Theory to Numerics and Applications*, pp. 329–342. Cham, Switzerland: Springer.

Villaverde, A. F., C. Raimúndez, and A. Barreiro (2012). Passive internet-based crane teleoperation with haptic aids. *International Journal of Control, Automation and Systems 10*, 78–87.

Wilde, M., M. Chan, and B. Kish (2020). Predictive human-machine interface for teleoperation of air and space vehicles over time delay. In *Proceedings of the 2020 IEEE Aerospace Conference*, pp. 1–14. IEEE.