Spotlight on Off-Screen Points of Interest in Handheld Augmented Reality: Halo-based techniques

Patrick Perea [1, 2] **Denis Morand** [2] **Laurence Nigay** [1]
[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France
[2] Schneider Electric, F-06510 Carros, France
{patrick.perea@etu.univ-grenoble-alpes.fr, {patrick.perea, denis.morand}@se.com,
laurence.nigay@univ-grenoble-alpes.fr

ABSTRACT

Navigating Augmented Reality (AR) environments with a handheld device often requires users to access digital contents (i.e. Points of Interests - POIs) associated with physical objects outside the field of view of the device's camera. Halo3D is a technique that displays the location of off-screen POIs as halos (arcs) along the edges of the screen. Halo3D reduces clutter by aggregating POIs but has not been evaluated. The results of a first experiment show that an enhanced version of Halo3D was 18% faster than the focus+context technique AroundPlot* for pointing at a POI, and perceived as 34% less intrusive than the arrow-based technique Arrow2D. The results of a second experiment in more realistic settings reveal that two variants of Halo3D that show the spatial distribution of POIs in clusters (1) enable an effective understanding of the off-screen environment and (2) require less effort than AroundPlot* to find POIs in the environment.

Author Keywords

Augmented reality; visualization; off-screen point of interest; Halo.

ACM Classification Keywords

• **Human-centered computing → Mixed / augmented reality**

INTRODUCTION

When performing tasks in handheld AR, users often need to look for nearby Points of Interests (POIs). Because of the restricted field of view of the AR device camera, users must turn around the spot to explore their augmented environments. We motivate the need to acquire information on off-screen objects with two industrial use cases.

ISS '19, November 10–13, 2019, Daejeon, Republic of Korea

ACM ISBN 978-1-4503-6891-9/19/11 …$15.00
https://doi.org/10.1145/3343055.3359719

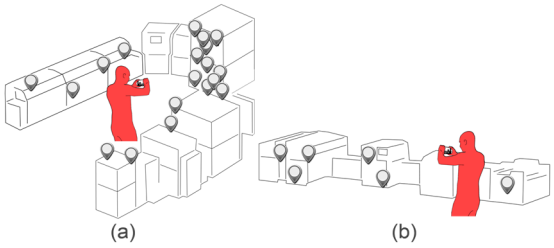

Figure 1: (a) Machine operator at the center of a U-shaped equipment line. AR maintenance POIs are located around the user. (b) Machine operator in front of a cuboid assembly line. AR maintenance POIs are located along the entire length of the line.

Scenario 1: U-shaped lines (see Figure 1a) are defined by a set of machines that form a "U". Machine operators move within the "U" to perform maintenance or examine the state of the line. As an example, UV coating lines are used to provide surface protection against harmful radiation effects. Using AR maintenance applications, machine operators can access data such as the temperature of the oven in real time and technical documentation. This information is distributed as POIs on the machine. Dense environments involve multiple POIs close to each other. For instance the control command of the UV coating machine is a small device (60cm x 2m) that gathers all the information on the state of the machine. This limited space can contain up to twenty POIs. As the line is U-shaped, the machine operators rotate around the spot to examine the state of the POIs located around them (see Figure 1a).

Scenario 2: With cuboid lines (see Figure 1b) such as production lines (up to 60 meters length) operators are always facing the machine as opposed to U-shaped lines. The POIs are consequently located in front of the operators. Examples of such cuboid lines include the set of machines used to make electronic circuits. When performing maintenance campaigns, the operators walk in front of the assembly line to perform a single task repeated X times at different locations on the assembly line. Each location is marked by a POI, and contains information (e.g. status, production rate, failure) about the machine. Knowing the location of POIs helps operators to optimize the campaign completion time by identifying the closest POIs from their positions.

From these scenarios, we derive two challenges for handheld AR industrial maintenance guidance systems: (1) manage POI-dense environments and (2) keep intrusion minimal and constant along the screen's edges to avoid overloading the focus area, used to consult data of selected POIs.

Perea et al. introduced Halo3D ([17],[18]), an AR technique to visualize off-screen POIs on handheld devices. Halo3D is adapted from Halo [1] a 2D visualization technique. Halo3D indicates the location of off-screen POIs by displaying halos (i.e. circle arcs) and manages high density environments by aggregating POIs when halos overlap. However, we raise two concerns: (1) no study has been conducted to compare Halo3D with existing visualizations, and (2) the Halo3D visualization of clusters does not provide information on the spatial distribution of aggregated POIs. In response to these concerns, the contributions of this paper are threefold:

(1) A comparative study between an enhanced version of Halo3D and two off-screen visualization techniques: AroundPlot* a technique inspired by AroundPlot [15] that indicates the location of off-screen POIs by drawing dots on the edges of the display, and Arrows2D, an arrow-based technique commonly used in industrial maintenance guidance systems.

(2) Halo3D* and Halo3D**, two halo-based visualization techniques that overcome Halo3D's limits respectively by allowing overlapping halos and by showing the spatial distribution of the aggregated POIs without overloading the user interface.

(3) A comparative study between Halo3D, Halo3D*, Halo3D**, AroundPlot* and Arrows2D in a pseudo-industrial environment.

RELATED WORK

As presented by Cockburn et al. [7] and Gustafson et al. [12], three main approaches are used to display a large information space on screen in 2D: Overview+Detail, Focus+context and Contextual views. We focus on Contextual Views as the context area does not hide or reduce the focus area.

We review Contextual Views by distinguishing the off-screen visualizations with proxies that appear entirely on screen (in-frame proxies-based views) from the visualizations with proxies that partially appear on screen (partially out-of-frame proxies-based views). For each type of visualization, we differentiate the techniques that work on an orthographic top-down view (2D off-screen object visualizations), from the techniques that show directions in 3D space (3D off-screen object visualizations).

In-frame Proxies-Based Techniques

EdgeRadar, Arrows and AroundPlot are three in-frame proxies-based techniques. They provide cues whose shape is fully visible on screen.

2D off-screen object visualizations

Arrows are commonly used to indicate directions towards off-screen POIs. Scaled Arrows and Stretched Arrows are two 2D techniques proposed by Burigat et al. [4] for displaying direction and distance to off-screen POIs. For Scaled (resp. Stretched) Arrows, the bigger (resp. longer) the arrow, the closer the off-screen POI to the user. However, the visualizations have not been evaluated with dense environments, only with up to 8 POIs.

EdgeRadar [13] is a 2D visualization technique that uses a small portion on each edge of the screen to display the locations of the off-screen POIs. The center of the screen defines the main work zone, and information about the context is only displayed on the borders of the screen. Distant POIs are indicated with square icons in the edge area. The main benefit is that it provides information on the spatial relationship between the POIs and the current work zone. However, EdgeRadar reaches its limit with high-density environments: in such cases the edge area becomes overloaded with small square icons.

3D and AR off-screen object visualizations

3D arrows [3] provide 3D direction information to reach off-screen POIs. Implemented in an egocentric point of view, 3D arrows allow users to consider all the 3D directions around them. 3D arrows were compared to 2D arrows and 2D Radar. The task was to find five POIs in a specified order. Results showed that users completed the tasks faster with 3D Arrows. However, when the density of off-screen POIs is high, the occlusion between the arrows seriously reduces users' navigation performance.

For handheld AR, Hyungeun et al. proposed Aroundplot [15]. Inspired by EdgeRadar [13], AroundPlot displays visual cues of the off-screen POIs in a small portion of each screen's edge. The technique maps 3D spherical coordinates of the environment to 2D orthogonal fisheye view. It then displays a small point for each off-screen POI in the edge area. However, the small size of this area often leads to perceptibility issues. To address this issue, the technique dynamically magnifies the edge area along the direction of the movement of the handheld device. An experimental study [15] compared AroundPlot with 2D radar and 3D arrows visualization for the cases of sparse (5 POIs) and dense environments (50 POIs). Results showed that users performed better using AroundPlot. Indeed 2D radar prevents users from estimating the height (y position) of the POIs and the occlusion between 3D arrows makes it difficult to interpret the visual cues.

Partially Out-of-the-frame Proxies-Based Techniques

Halo [1] and Wedge [12] are two off-screen visualization techniques based on the theory of amodal completion. The amodal perception theory suggests that the human visual system will complete parts of a partially visible shape [21].

2D off-screen object visualizations

The Halo technique [1] indicates the location and distance of off-screen POIs using 2D arcs: the visualization

computes a circle whose center is the distant POI and the radius is the distance between the POI and the border of the screen. Only a section of the shape appears on screen to prevent screen overload. The user mentally completes the figure from the visible arc to understand the location of the off-screen POI (amodal perception). Halo has been compared to both Scaled and Stretched Arrows [4]. The study required users to locate and compare off-screen POIs on a 2D map. Results showed that users identified the correct off-screen POI's position more efficiently with Halo than with the two arrow-based techniques. As identified in [1] for static targets and in [13] for moving targets, Halo reaches its limits in POI-dense environments: the arcs overlap and lead to substantial clutter.

To reduce screen clutter, one solution is to aggregate overlapping halos. Baudish et al. suggested that Halo arcs that strongly overlap are merged into a single multi-arc [1]. The multi-arc consists of thinner concentric arcs centered at the barycenter of the POIs that aggregate. When the group exceeds 3 POIs, the multi-arc displays a thick double ring on screen. While this approach slightly reduces the number of overlapping halos on screen, the mechanism suffers from two drawbacks: (1) it does not prevent overlapping halos on screen as it only applies to strongly overlapping halos, and (2) it increases the visual intrusion of the Halo technique on the edges of the screen. The HaloDot [11] technique adopts another approach: a hidden grid is overlaid on the off-screen space to split it into cells. Two off-screen POIs are aggregated if they are located within the same cell. For each cell containing POIs, HaloDot calculates a Halo and displays the number of aggregated POIs on the corresponding arc. However, the HaloDot technique (1) does not provide any information on the spatial distribution of the aggregated POIs of a cell and (2) does not prevent overlapping halos: if two POIs are close to each other but located in different cells, the corresponding halos will overlap on screen. Finally to reduce screen clutter, filtering the POIs to be displayed is a complementary approach. Baudish et al. [1] proposed to display only the off-screen POIs whose distance from the border of the screen is lower than a predefined threshold. In any case, users partially loose awareness of the surrounding POIs.

The Wedge visualization technique [12] also addresses the issue raised by dense environments. Wedge displays triangles instead of circles as with Halo. The user mentally extends the segments of the triangle to estimate where the off-screen POI is. The technique provides distance indications thanks to the aperture between the segments and the intrusion into the visible space. Wedge avoids overlapping visual cues by rotating them. However, the main drawback of Wedge is the intrusion on screen. Users also preferred Halo when looking for the closest POIs from their position: the visual difference between close and distant POIs was easier to perceive with Halo, as the visual difference was greater. Moreover, Wedge has only been tested with 5 POIs packed in a small area. Burigat et. al [5] showed that when increasing the number of POIs up to 30 POIs, Wedge was not robust enough to prevent overlap.

The two visualization techniques Halo and Wedge have been adapted to 3D and AR environments.

3D and AR off-screen object visualizations

The 3D halo circle technique by Trapp et al. [22] is an adaptation of Halo2D [1] for 3D Virtual Reality (VR) environments. The circle is placed in perspective in the scene and the radius of the circle conveys the physical distance as the crow flies between the POI and the user. However, the further an object, the greater the intrusion of the shape on screen. It then becomes difficult to mentally complete the circle and estimate the location of the POI.

Perea et. al ([17],[18]) provided a first attempt to adapt Halo [1] to 3D Augmented Reality (AR) environments using a handheld device. The radius of the halo is computed by first projecting the POI onto the device's screen plane, and then considering the distance between the projected POI and the edge of the display (see Figure 2a). Halo3D includes an aggregation mechanism to avoid any overlap on screen: Halo3D aggregates two circles only if they visually overlap on screen. This is made by detecting if the intersection point between two circles is located on screen. If so, a new halo replaces the overlapping halos and is centered on the cluster's centroid. However, no study has been conducted to compare Halo3D with existing off-screen visualization techniques.

More recently Gruenefeld et al. [10] adapted Halo [1] and Wedge [12] to 3D AR environments using a head-mounted display (HMD). Compared to 2D arrows, the HMD-Halo technique has been found to perform better on task completion time and error, whereas the HMD-Wedge technique was subjectively perceived as the best. However, the three techniques have been compared in an environment with less than 10 POIs. Moreover, POIs were all placed in a 90° field of view. Gruenefeld et al. also compared Halo and Wedge's guidance performances in AR environments with their performances in VR environments using a HMD [9]. But again, the studies only involved up to 8 POIs, and no mechanism has been proposed to manage a large amount of off-screen POIs.

Our review of existing techniques suggest that only Halo3D fulfills the requirements derived from industrial scenarios 1 and 2: Halo3D has been introduced as a visualization technique for handheld AR that manages dense environments without being intrusive on screen. Although AroundPlot does not have constant intrusion along the edges of the screen, it is the only other technique that has been developed for handheld AR and tested with a large number of POIs. Additionally, the 2D arrows are widely used in todays' industrial handheld AR systems to point towards maintenance POIs. We compare both visualization techniques (AroundPlot and Arrows) with Halo3D and its variants in the following two user studies.

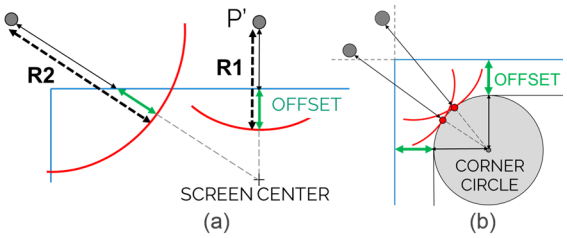

Figure 2: Algorithms to compute halos' radius (a) Halo3D ([17],[18]): as the projection P' of a POI shifts towards the corner of the display, its halo's radius increases (b) Extension of Halo3D: an invisible inner rectangle with rounded edges guarantees a constant visual intrusion on the screen edges.

USER STUDY 1

Inspired by scenario 1, the study considers an abstract pointing task to acquire off-screen POIs located around the participants.

Implemented visualization techniques

The implementation of Halo3D follows the design described by Perea et. al ([17],[18]) with two extensions. First we extend Halo3D to guarantee constant visual intrusion on screen. To do so we consider an invisible inner rectangle with rounded edges as shown in Figure 2b. We kept the radius of the inner rectangle's corner circles to 5% of the screen's width, as the visual intrusion is minimal. Second the number of aggregated POIs is displayed directly on the circle (see Figure 3c) and not next to the circle as in ([17],[18]).

We implemented the Arrows2D technique with the same design as the Schneider's AR off-screen guidance system (see Figure 3a). The displayed arrows are 150 pixels wide and long. We also implemented the same aggregation mechanism as for Halo3D: two arrows aggregate when they visually overlap on screen. The size of the arrows was chosen as a compromise between the number of arrows displayed and their intrusion on screen. Indeed smaller arrows (< 100 pixels) lead to less overlap between them, but imply more arrows on screen as less aggregation is performed.

AroundPlot* is a point-based technique inspired by AroundPlot [15]. We replicated the visualization of the original AroundPlot technique: first, a white rectangle is drawn on screen and delimits the current view from the surrounding context displayed on the edges. Small black dots with white outline are then rendered on the edges of the display to show the location of off-screen POIs (see Figure 3b). We also replicated the dynamic resizing of AroundPlot: the inner rectangle is not fixed on screen but magnifies in the direction of the movement of the handheld device. This mechanism improves the perceptibility and intelligibility of the off-screen environment.
For a fair comparison between the three visualization techniques, the implemented version of AroundPlot, namely AroundPlot*, is based on the same POI's projection

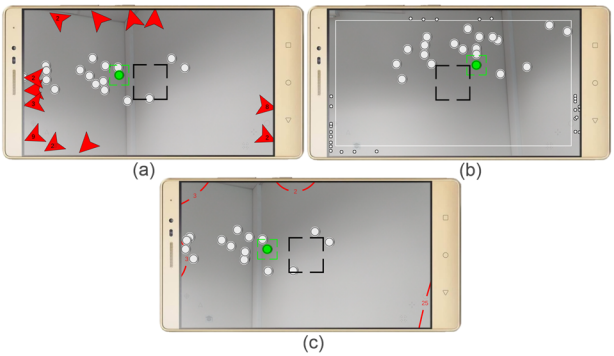

Figure 3: Implemented visualization techniques (a) Arrows2D (b) AroundPlot* (c) Extented Halo3D

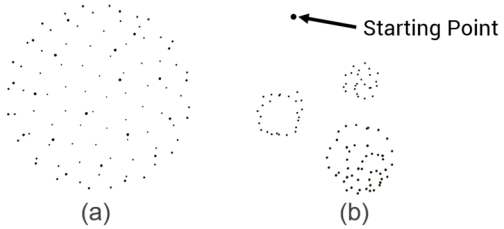

Figure 4: Study 1: Environments generated with 100 POIs (a) placed on a support sphere (b) arranged as 3 clusters

algorithm as both Halo3D and Arrows2D: the off-screen POI is projected on the screen's plane of the handheld device. AroundPlot* positions the small icon representing a POI on the edge of the display along the line defined by the screen's center and the POI's projection. Finally, with AroundPlot* no aggregation was needed as the points in the context area are small and do not often collide.

All three techniques were developed in C# and rendered on the handheld device's screen using Unity3D engine.

Experimental Task and Environment

The task and the environment are inspired by scenario 1: point at off-screen POIs distributed around users. Participants were asked to bring a green highlighted off-screen POI into the camera field of view. When the POI was displayed on screen, the participants validated the acquisition by positioning the POI fully inside the square cursor placed at the center of the screen (see Figure 3).

Two parameters guide the design of the environments: first, the density of POIs is variable, as illustrated by scenario 1. To measure the performance of the visualization techniques according to the density of off-screen POIs, we generated two environments as done in the conducted evaluation of AroundPlot [15]: a low-density environment with 5 POIs, and a high-density environment with 50 POIs.

A second parameter is the movement amplitude. As illustrated in scenario 1 (see Figure 1a), operators may have to completely turn around the spot to acquire a POI or just perform a slight movement. In the experiment we consider two movement amplitudes: a small movement defined by

an angular distance inferior to 80 degrees and a large movement defined by an angular distance superior to 120 degrees (a POI behind the user). This distance is measured from a special POI called the starting point, identical for all pointing tasks. Thus, participants were asked to return to the starting point after each pointing task.

Based on these parameters, we generated the environments as follows. We first placed 100 POIs on a sphere using Fibonacci algorithm (see Figure 4a). The 100 points were then grouped around 3 centers to create a clustered environment: one on each side of the participants and one behind them (see Figure 4b). The POIs placed on the side of the participants involve small amplitude movements. The one placed behind them implies large amplitude movements. As participants performed 6 iterations for each density condition, we created 6 environments for both sparse and dense conditions to avoid any learning effect. Thus, we generated 12 different environments.

Apparatus and Participants

The study was carried out on a Lenovo phab2PRO phone featuring Google's Tango technology. It provides the device with full spatial awareness thanks to a combination of motion tracking and depth perception. The phone also provides a 6.4 inches screen with a resolution of 2560 x 1440 pixels which was fully used in this experiment. The study involved 12 unpaid volunteers from a computer science university lab (3 females, 9 males; ranging in age from 27 to 40 years, mean=30.5, sd=3.8).

Hypotheses

We formulated the following hypotheses on Halo3D:

H1. It will be faster to complete the tasks with Halo3D as compared to AroundPlot*. As the AroundPlot*'s points are small, it will be longer to locate a green POI's point than a larger green circle.

H2. With a high density of POIs, Halo3D is the less cognitively tiring technique in comparison with AroundPlot* and Arrows2D. This is due to the clutter-free and constant intrusion properties of Halo3D.

Measured data

During the experiment, we recorded the time participants took before moving towards a target (visualization interpretation time) as well as the time spent until the green highlighted POI appeared on screen (off-screen guidance time). We also recorded the number of overshoots for each task. An overshoot happens when the participant misses the targeted POI: the participant follows the visual cue and the movement is so fast that the POI appears on screen and then disappears. At the end of the entire session, participants were asked to rate, for each visualization technique, the intrusion on screen, how mentally demanding the task was and how easy the techniques were to learn. Questions were to be answered on a 5-level Likert scale [16]. Finally, the participants were also asked to rank the visualization techniques and to fill a SUS form [2] for each of them. The entire experiment had an average duration of 30 minutes.

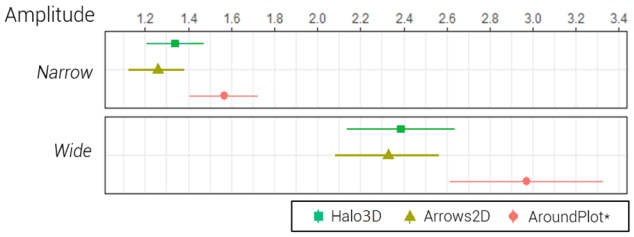

Figure 5: Off-screen pointing times in seconds for the 3 techniques and the 2 movement amplitudes. Error bars reveal 95% confidence intervals.

Experimental design and procedure

Design

The study was based on a 3 (Visualization: Halo3D, Arrows2D, Aroundplot*) x 2 (Density: number of off-screen POIs: 5, 50) x 2 (Movement Amplitude: narrow and wide movements) within-subjects design. The visualization techniques were counter-balanced across participants using a Latin square. For each visualization technique, all combinations of the Density and Amplitude conditions were presented in a random order to counter learning effects [8]. Each condition was repeated 6 times. This corresponds to 24 pointing tasks (6 iterations x 2 density of POIs x 2 amplitudes) for each visualization technique and 72 tasks per participant. We defined 12 equivalent environments (6 iterations x 2 density of POIs) as explained above.

Procedure

Participants carried out the experiment in a standing position. To look for the POIs, the participants had to rotate on the spot using the visual cues displayed on the phone's screen. They first completed a training phase for each visualization technique. The training phase involved a different environment from those of the trial phase to avoid any learning effect. Once they felt completely comfortable with one visualization technique, they switched to the following visualization technique.

Average training phase with Arrows2D was 19.1s, 24.2s with AroundPlot* and 26.8s with Halo3D.

Results

We report the results by considering the effects of each of the three factors (Visualization, Density, and Amplitude) on the time to bring the off-screen POI on screen and the number of overshoots. All the iterations of each trial were aggregated for each participant. Table 1 presents the average results. A Shapiro-Wilk test indicated that our data were significantly different from the normal distribution. Thus, we applied an Aligned Rank Transformation on the data [23] and then ran repeated measures ANOVA on the aligned ranks to investigate possible interactions between factors.

Technique	H3D	Arr2D	Ar*
Interpretation time (s)	0.9(0.16)	0.91(0.18)	1.17(0.37)
Off-screen guidance time (s)	1.86(0.47)	1.79(0.46)	2.27(0.64)
Overshoots	9(0.57)	10(0.71)	10(0.49)

Table 1: Average (and standard deviations) interpretation and off-screen guidance times (s) as well as number of overshoots for Halo3D (H3D), Arrows2D (Arr2D) and AroundPlot* (Ar*)

Time and Overshoots

A three-way ANOVA showed that Visualization ($F(2,132)=11.3$ and $p < 0.001$) and Amplitude ($F(1,132) = 210.1$ and $p < 0.001$) had a significant impact on the off-screen guidance time (see Figure 5). As expected, the off-screen guidance time was significantly higher when performing large movements (M=2,56s, SD=0,28s) than narrow movements (M=1,39s, SD=0,14s). Post-hoc test using Wilcoxon signed rank indicated that participants spent significantly more time completing the tasks with AroundPlot* than both Halo3D ($p < 0.001$) and Arrows2D ($p < 0.001$). No significant difference had been found between Halo3D and Arrows2D ($p > 0.05$).

A study of the impact of Visualization on the interpretation time with a Wilcoxon signed rank post hoc test confirmed that statistically, users spent significantly more time interpreting AroundPlot* visualization than both Halo3D ($p < 0.001$) and Arrow2D ($p < 0.001$) visualizations.

Finally, we did not find any significant impact of the factors on the number of overshoots.

Subjective preferences

The SUS scores indicate that Halo3D (74.75/100) and Arrows2D (81/100) were perceived more usable than AroundPlot* (70/100).

A Friedman test showed that the visualization techniques had a significant impact (with $\alpha = 0.05$) on the perceived intrusion on screen ($\chi^2_{(2)} = 9.41$, $p = 0.009$), how mentally demanding the task was ($\chi^2_{(2)} = 11.2$, $p = 0.004$) and how easy the visualization techniques were to learn ($\chi^2_{(2)} = 7.3$, $p = 0.03$). But no significant difference has been found on the overall ranking. An analysis with post-hoc revealed that Arrows2D was perceived significantly more intrusive than both Halo3D ($p = 0.03$) and AroundPlot* ($p=0.01$). AroundPlot* was also found to be significantly more mentally demanding than Arrows2D ($p = 0.003$). Finally, participants found AroundPlot* more difficult to learn than Arrows2D ($p = 0.03$).

Discussion

Task completion time

Results provide evidence in support of our hypothesis **H1**: participants spent significantly more time completing the tasks with AroundPlot* than both Halo3D and Arrows2D. The time to complete the tasks include (1) the time needed to interpret the visualization and understand which direction needs to be followed and (2) the time needed to bring the off-screen POI back on the field of view.

As shown in Table 1, participants brought the POIs in the camera's field of view 18% faster using Halo3D than AroundPlot*. AroundPlot*'s low performance is due to the dense cloud of points in the context area. As reported in Table 1, the time spent looking for the AroundPlot's green point was 24% significantly higher than interpreting both Halo3D and Arrows' visual cues. The 150 pixels wide arrows helped participants to easily locate the green visual cue on screen. Halo3D offered similar average time to locate the green visual cue on screen (i.e. circle arc) while being less intrusive on screen than Arrows2D. These results show the benefits of Halo3D.

Mental difficulty

The ranking of the visualization techniques is correlated with the mental difficulty felt during the tasks.

Arrows2D was slightly preferred over Halo3D: as the shape of the arrow intuitively conveys a direction, it is mentally easier to deduce a direction from an arrow than from a circle arc. This could explain why users have found Halo3D slightly more mentally demanding than Arrows2D, in contrast to our hypothesis **H2**.

AroundPlot* was rated significantly more demanding than Arrows2D: AroundPlot* provides a complete overview of the off-screen space in a small section of the screen, the context area. In high density environments, those points are close to each other and it requires a strong focus to find and follow the green point on screen. The mental fatigue is thus linked to the density of information displayed and the size of the screen's area in which they are displayed.

Effect of the environments' configuration

The number of POIs (5 vs 50) had no significant impact on users' performance. This result shows that the aggregation mechanism does not impact on the performance while providing a solution to deal with high-density environments.

None of the visualization techniques had a significant impact on the number of overshoots. This was mainly caused by the spatial distribution of the POIs. The points were gathered into clusters to simulate a real-world scenario. As participants get closer to a POI, the tip of the cluster that contains the POI starts appearing on screen. The participants knew that they were getting closer to the POI and slowed down their movements, resulting in less overshoots.

Results indicate that Halo3D is an effective technique for 3D pointing tasks in mobile AR, especially when dealing with dense environments. Halo3D performs as well as Arrows2D while being less intrusive. Moreover Halo3D performs faster than AroundPlot*. However, two participants verbally raised concerns about Halo3D's

cluster visualization: "The point of interest that entered on screen was outside of the circle", or "I expected the point of interest to enter on screen towards the center of the circle. But it entered at a different place and I had to re-adjust my movement". These comments indicate that participants were not able to fully understand the spatial distribution of the POIs inside a cluster. Participants had to change direction when POIs were un-aggregated since the followed direction was wrong. To address this issue, we propose in the following section two variants of Halo3D.

HALO3D'S CLUSTER VISUALIZATION TECHNIQUE

As stated by Baudisch et. al in [1], Halo is not suitable for POI-dense environments: the circle arcs overlap and overload the user interface. With Halo3D, Perea et al. ([17],[18]) proposed an aggregation mechanism that removes any visible overlap on screen. However, this aggregation may lead to incorrect guidance information on screen as observed in Study 1. Let us consider a group of POIs which centroid is located towards the upper left section of the off-screen space (see Figure 6). To reach a POI located at the bottom of this cluster, Halo3D will guide users in two steps: first towards the centroid of the cluster and then down towards the POI. This misinformation is caused by the loss of the spatial distribution of POIs when aggregation is performed. To minimize this issue, the two variants of Halo3D, namely Halo3D* and Halo3D** respectively (1) reduces aggregation by displaying overlapping halos according to a threshold (2) modifies the way the aggregated POIs are displayed on screen.

Halo3D*: Overlapping halos with a threshold

Halo3D* is based on the hypothesis that small overlaps between halos do not decrease the understanding of the AR environment by the user. To avoid displayed arcs with large amounts of overlap along the entire length of the arc, we fixed by trial and error a threshold of 70 pixels for the maximum size of overlap between two arcs. The Halo3D algorithm that determines if two circles overlap takes as parameters the center and radius of the circles to analyze. To implement Halo3D*, we reused to same algorithm applied to smaller circles (i.e. radius – threshold=70 pixels). On screen, the size of the circles remains unchanged and small overlaps are authorized.

Halo3D**: Aggregated POIs displayed as curves

Halo3D** show the spatial distribution of off-screen aggregated POIs by displaying curves instead of circles.

A curve allows users to visualize the extent of an off-screen cluster and is computed in 5 steps: first, the halo of all the off-screen POIs are calculated based on the same principle as Halo3D ([17],[18]) (see Figure 7a). The algorithm then determines the intersection points with the edges of the screen and between the different circles (see Figure 7a). As a third step, the algorithm determines the arcs that will be part of the final curve (see Figure 7b): an arc is part of a circle between two intersection points located on that circle. The arcs contained inside a halo (e.g. arc (AB) in Figure 7a)

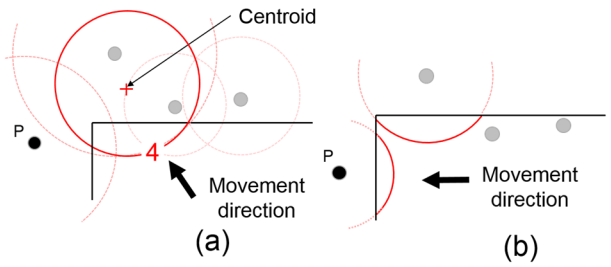

Figure 6: Halo3D guides the user towards POI P in two steps: (a) Halo3D first guides the user towards the centroid of the aggregated POIs (b) once the POIs disaggregate, Halo3D guides towards POI P.

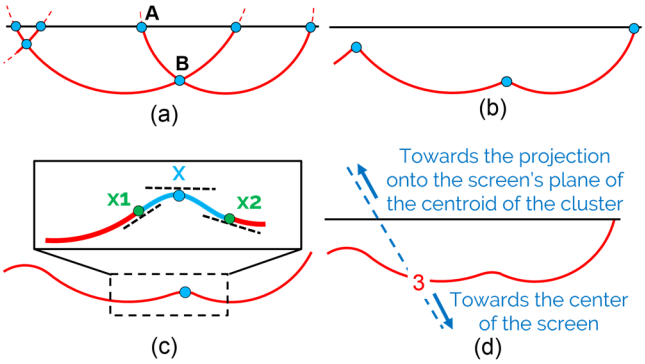

Figure 7: (a): Step 1 and 2 - All halos and their intersections are computed. Arc (AB) is contained inside a halo and will not be part of the final curve. (b): Step 3 - Arcs that compose the final curve are computed. (c): Step 4 - Example of a quadratic curve binding two arcs together, using three points (x1, X, x2) and three slopes (black dashed lines). (d): Step 5 – The number of aggregated POIs is displayed on top of the curve.

are not part of the final curve. The intersections between the arcs are then smoothed out. To do so, the algorithm uses quadratic curves to bind the arcs together. As seen in Figure 7c, the intersection point X between two arcs is one point of the quadratic curve that binds the arcs together. Two other points (x1, x2) are also computed, one on each intersected arc, such as their distance to X is a third of the corresponding arc's length. By knowing the coordinates of each point and the slopes at those points, quadratic curves can easily be processed. As a result, the algorithm displays a smooth curve on screen that represents the spatial distribution of the aggregated POIs. Finally, the number of aggregated POIs is displayed on top of the curve, at the intersection point between the curve and the line that goes through the center of the screen and the projection onto the screen's plane of the cluster centroid (see Figure 7d).

USER STUDY 2

In Study 2, we move our investigation further toward real-world use in order to examine which technique better improves the understanding of the off-screen environment. Inspired by scenario 2, the objective of the task is to explore a pseudo-industrial environment and to find all the POIs

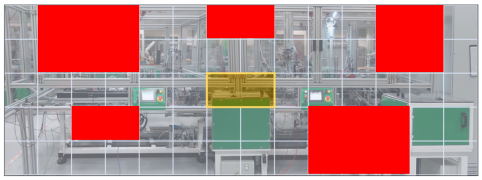

Figure 8: Locate Task: Screen capture of the grid. In red: cells selected by a participant. In yellow: position of the view displayed on screen.

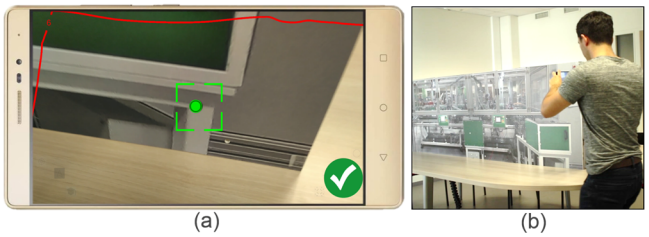

Figure 9: Study 2: Exploration task (a) Selection of a POI using Halo3D (b) A participant moving in front of the model of a production line to find the POIs.**

anchored to a production machine. This study differs from Study 1 in three ways: 1) The users' field of view on screen shows a visually loaded environment with cables and sensors on a face of a production machine. 2) Users are free to follow any trajectory to select all the POIs. 3) We compare five visualization techniques, Arrows2D, AroundPlot*, Halo3D, Halo3D* and Halo3D**.

Experimental Task and Environment

The experimental task was designed to reproduce our industrial scenario 2: participants were asked to search and select POIs placed in front of them on a machine. They were presented 35 POIs on one face of a production line. The experimental task was divided into two parts: (1) a task that involved locating on a grid all the off-screen POIs of the machine, based on the visualization on screen; (2) an exploration task for searching and selecting all the POIs on the machine.

Locate Task. The first task was inspired by the "Locate Task" of both Halo [1] and Wedge [12] studies: participants were asked to locate all the off-screen POIs using a given technique. Participants were presented a grid on a computer, made of 14 columns and 5 lines. The grid had a background picture of the machine (see Figure 8). Participants were asked to click on the cells in which they thought one or more POIs were located.

Exploration Task. Participants were asked to search and select one by one each POI in the order they wanted. A POI visible on screen becomes selectable when the participant is closer than 30 cm. If so, a "Validate" button appears next to the POI (see Figure 9a). The selection is made by aligning the POI with the screen's central cursor and pressing the "Validate" button. When selected, the POI disappears from the AR environment. This task (1) simulates a maintenance operation performed on a POI (e.g. control the temperature value at a specific spot of a machine) and (2) forces participants to rely on the off-screen visualization technique to bring the remaining POIs on screen as in Figure 9. Participants being close to the machine, the remaining POIs will be out of the field of view of the handheld device camera.

To avoid any learning effect from one technique to another, we generated 5 equivalent environments (one for each technique) containing 35 POIs each. Each environment was composed of 5 clusters of POIs. The clusters of POIs contained respectively 8,6,8,8 and 5 POIs. We then built the 5 environments by changing the location of the clusters on the machine, such as the clusters had different positions in each environment. The generated environments were the same for all participants.

Apparatus and Participants

The experiment was carried out using the same device as in Study 1. To replicate a complex industrial environment, we took photos of a 9m x 3m Schneider's production machine. We print them on 3mm depth forex panels to build the unit (see Figure 9b). The size of the model was 3m long and 0.75m large.

We recruited 15 unpaid volunteers (3 females, 12 males; ranging in age from 23 to 47 years, mean=31, sd=6.4) from a computer science university lab. None of the participants had any expertise regarding production machines. They were different participants than in Study1.

Hypotheses

We formulated the following hypotheses:

H3. Halo3D*, Halo3D** and AroundPlot* are less error-prone for the *Locate Task* than other techniques. Indeed, Arrows2D does not provide any distance information and Halo3D does not display the spatial distribution of POIs.

H4. Halo3D and its variants are less mentally tiring compared to AroundPlot*, as AroundPlot* displays all the POIs in a small section of the screen.

Measured data

For each technique, we recorded the time spent to (1) locate the off-screen POIs on the grid and (2) to validate all the POIs. After completing both the *Exploration* and *Locate* tasks for one technique, participants were asked to fill out a NASA-TLX **Error! Reference source not found.** form. At the end of the entire session, participants were asked to rate, for each visualization technique, the intrusion on screen, how mentally demanding the main exploration task was and how easy the techniques were to learn. Questions were to be answered on a 5-level Likert scale [16]. Finally, the participants were also asked to rank the 5 techniques.

In the *Locate* Task, we computed of the number of incorrect answers in the grid. An error is computed when participants selected an incorrect cell (i.e. a cell that does not contain an off-screen POI) or missed a correct cell (i.e. a cell containing one or more off-screen POIs).

Experimental design and procedure

The task involved a 5 (visualization techniques: Arrows2D, AroundPlot, Halo3D, Halo3D*, Halo3D**) within-subjects design. All techniques were counterbalanced among participants: we divided the participants into 5 groups of 3 individuals, such as each group started the experiment with a different technique. To ensure that the couple (technique x environments) was counterbalanced amongst all participants, participants were presented the environments in the same order. As techniques are counterbalanced, each technique is associated to all 5 environments across all participants. Finally, each participant conducted 175 pointing tasks (5 environments x 35 POIs).

Subjects were first given a short explanation about the experimental tasks and all 5 techniques. Due to the number of techniques, participants were invited, for each technique to perform the trial session immediately after the training session [6]. The training tasks consisted of locating the off-screen POIs on the grid and selecting the POIs on the machine. Training environments contained 12 POIs, placed at different locations from the trial tests. For each technique, the experiment began only when the participants felt comfortable with the task.

Participants began the task from a specific location of the machine. The spot was marked with a white cross on the ground. After looking at the presented visualization on the device' screen, they were first asked to fill in the grid of the *Locate Task* (see Figure 8) on a computer while the experimenter was holding the handheld device at a fixed position (corresponding to the yellow part in Figure 8). Then, participants were invited to search and select the POIs on the machine. When there were only 10 POIs left, participants were asked to fill in again the grid of the *Locate Task*. This allowed us to study which technique better supports the understanding of the off-screen environment for the cases of a high-density (first *Locate Task*) and low-density environments (second *Locate Task*).

After completing both *Exploration* and *Locate* tasks with one technique and one environment, participants filled out the NASA TLX form and returned to the starting location to start with the following technique. The overall session lasted approximately 40 minutes.

Results

Error rate and Completion time

We report the results by considering the effects of each Visualization on the error rate during the *Locate Task* (see Figure 10) and the time needed to select all the POIs on the machine during the *Exploration Task* (see Figure 11). A Shapiro-Wilk test indicated that our data did not follow a normal distribution. The non-parametric Kruskal-Wallis test revealed no effect of the Visualization on the exploration time ($\chi^2_{(4)} = 6.17$, p=0.18), and on the error rate in high density ($\chi^2_{(4)} = 6.69$, p=0.15) and low density ($\chi^2_{(4)} = 7.89$, p=0.09) environments.

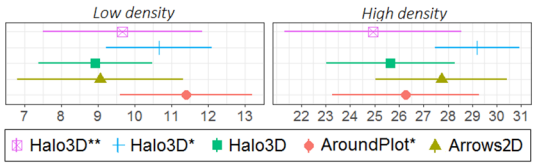

Figure 10: Error rates in both low and high density environments. Error bars reveal 95% confidence intervals.

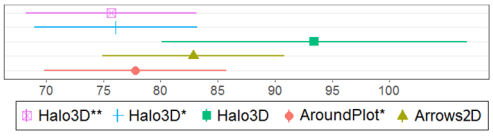

Figure 11: Exploration times for all techniques. Error bars reveal 95% confidence intervals.

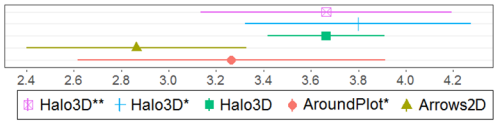

Figure 12: Notes out of 5 (5 = very good) assigned to the techniques. Error bars reveal 95% confidence intervals.

Subjective preferences

NASA TLX results. A Friedman test finds a strong effect of the Visualization on the effort ($\chi^2_{(4)} = 10.69$, p=0.03), and the physical demand ($\chi^2_{(4)} = 11.4$, p=0.02) felt during the *Exploration Task*. A post-hoc test with pairwise comparisons revealed that AroundPlot* required significantly more effort than Halo3D*, and Arrows2D was significantly more physically demanding than Halo3D**. No statistically significant differences in all other measurements were found: frustration ($\chi^2_{(4)} = 8.7$, p=0.06), mental demand ($\chi^2_{(4)} = 6.87$, p=0.14), performance ($\chi^2_{(4)} = 9.6$, p=0.05) and temporal demand ($\chi^2_{(4)} = 2.14$, p=0.71).

Final survey results. A Friedman test finds a strong effect of the Visualization on the intrusion on screen ($\chi^2_{(4)} = 10.33$, p=0.03). A look at the pairwise comparisons revealed that Arrows2D was rated significantly more intrusive than Halo3D. No statistically significant differences in all other measurements were found: easy to learn ($\chi^2_{(4)} = 2.93$, p=0.57), mental tiredness ($\chi^2_{(4)} = 6.08$, p=0.19) and overall note ($\chi^2_{(4)} = 4.76$, p=0.31).

Discussion

Amount of information on the off-screen environment

AroundPlot* is the only technique that directly displays the off-screen environment including the spatial relationships between off-screen POIs. There is a direct mapping between the off-screen POIs and the displayed dots in the context area. Although we did not find statistically significant differences in the error rate, 8 participants stated that AroundPlot* induced them to fill the grid with more precision in comparison with the other techniques. AroundPlot* enables a good understanding of the off-screen environment but this precision was reported not useful for the *Exploration Task* by participants. One

participant stated that "I don't need to know where exactly are located the POIs on the machine, only the area and how many they are: it is enough for me". Another participant made an analogy with human peripheral vision: "when you look at an object in front of you, you can see it clearly. However, all the objects located in your peripheral vision are blurred, but you know approximately where they are". This analogy perfectly suits halo-based techniques, as they provide the direction and the distance where off-screen objects might be located.

In contrast to AroundPlot*, Arrows2D only provides a direction to an off-screen POI. Arrows2D was rated the worst technique for the *Locate Task* as well as in the final ranking (see Figure 12). All the participants stated "I have no idea how far the points are, because no distance information is provided". Such feedback from participants supports our hypothesis **H3**, but the experiment did not reveal a statistically significant impact of the techniques on the error rate. In addition the effect of halo-based techniques being slower than Arrows2D was not revealed during the *Exploration Task*. This confirms the results of Study 1: a halo-based visualization has similar time performances as Arrows2D while providing additional distance information.

Halo-based techniques

Although we did not find statistically significant higher error rate during the *Locate Task*, contradicting our hypothesis **H3**, 7 participants orally reported being misled by Halo3D: the POIs they positioned in the grids did not match what they found on the machine during the exploration phase. Indeed, Halo3D only provides location and distance towards the centroid of off-screen clusters. As a cluster may contain spread out POIs, some POIs indicated by an aggregation arc could be located outside of the corresponding circle.

Halo3D* overcomes above issues by performing less aggregation. The overlapping circles did not disturb participants, as Halo3D* was not rated more cognitively demanding than Halo3D. Participants performed similarly using Halo3D* and Halo3D**. However, participants' comments provided insight into why they felt more comfortable using Halo3D*. One participant stated that "it was easier to guess the POI location using Halo3D*, as the complete arc is visible on screen". With Halo3D**, the arcs that overlap are cut at their intersection to smooth the curve (see Figure 7d). Thus, Halo3D** curves never display complete arcs from screen's border to screen's border. The more arcs that compose the curve, the harder it becomes to determine the location of the aggregated POIs.

High vs low density environments

We considered a 35-POI environment to be a high density environment, as more POIs would lead to longer and more exhausting exploration tasks. However, 35 POIs were enough to reveal the limitations of several techniques: when starting the *Exploration Task*, Halo3D** displayed a single continuous curve that started from the left edge of the screen, went through the top edge and ended up on the right border of the display. As stated by 6 participants, we end up with a visualization that "does not provide any useful information about the off-screen environment", just as thirty arrows would be useless on screen.

In high density environments, AroundPlot* lead to groups of points overlapping in the context area. As mentioned by 9 participants, AroundPlot* force users to intensively focus on the tiny context area, in order to determine how many and how far the POIs are. Knowing how many POIs are off-screen using AroundPlot* is more difficult as users would need to count every dot in a tiny screen section. With the aggregation mechanism and the number of aggregated POIs displayed on screen, halo-based techniques better suit industrial use cases, as in Scenario 2, where machine operators need to know how many maintenance tasks are left. Thus, participants rated AroundPlot* with a higher cognitive load as compared to halo-based techniques. Although such participants' feedback is coherent with our hypothesis **H4**, the experiment did not reveal a statistically significant impact of the techniques on mental tiredness.

In low-density environments, 5 users reported not using the AroundPlot* visualization during the *Exploration Task* as "it was difficult to locate the points left on screen". Unlike Study 1, this behavior did not impact the time to complete the task: those participants were still able to manually search for the points on the machine. Finally, low density environments characterize the only situation where participants found Arrows2D useful during the *Exploration Task*, as stated by 4 participants. Indeed, the arrows do not often overlap and are easier to locate on screen.

CONCLUSION

We first conducted an evaluation of Halo3D with a pointing task. Results show that Halo3D performed faster than AroundPlot* and as well as Arrows2D while being less intrusive on screen. We then presented Halo3D* and Halo3D**, two variants of Halo3D that provide additional information on the spatial dispersion of POIs. In a more ecological setting, results show no statistically significant impact of the techniques on error rate and mental tiredness. Nevertheless participants reported being misled by Halo3D but not by Halo3D* and Halo3D**.

However, no technique is perfect in every situation: Halo3D may mislead if too many aggregations are performed, Halo3D* may display too many overlaps on screen and Halo3D** may draw a single curve displayed all over screen's edges if POIs are spread out. Therefore, future work will (1) focus on a visualization technique that automatically adapts to the surrounding context: density of POIs and location of POIs (clustered or spread out), and (2) conduct an in-field evaluation. Halo3D has already been integrated in a professional AR industrial maintenance application of a Schneider's client in order to collect machine operators' feedback.

REFERENCES

[1] Patrick Baudisch and Ruth Rosenholtz. Halo: a technique for visualizing off-screen objects. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03), 481-488.

[2] John Brooke. SUS: A Quick and Dirty Usability Scale. In Usability Evaluation in Industry. Taylor & Francis, 189-194. 1996.

[3] Luca Chittaro and Stefano Burigat. 3D location-pointing as a navigation aid in Virtual Environments. In Proceedings of the working conference on Advanced visual interfaces (AVI '04), 267-274.

[4] Stefano Burigat, Luca Chittaro, and Silvia Gabrielli. 2006. Visualizing locations of off-screen objects on mobile devices: a comparative evaluation of three approaches. In Proceedings of the 8th conference on Human-computer interaction with mobile devices and services (MobileHCI '06), 239-246.

[5] Stephano Burigat, Luca Chittaro, and Andrea Vianello. 2012. Dynamic visualization of large numbers of off-screen objects on mobile devices: an experimental comparison of wedge and overview+detail. In Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services (MobileHCI '12), 93-102.

[6] Géry Casiez, Daniel Vogel, Qing Pan, and Christophe Chaillou. 2007. RubberEdge: reducing clutching by combining position and rate control with elastic feedback. In Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07), 129-138.

[7] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. 2009. A review of overview+detail, zooming, and focus+context interfaces. ACM Comput. Surv. 41, 1, Article 2 (January 2009), 31 pages.

[8] Barrett Ens, David Ahlström, Andy Cockburn, and Pourang Irani. 2011. Characterizing user performance with assisted direct off-screen pointing. In Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11), 485-494.

[9] Uwe Gruenefeld, Abdallah El Ali, Susanne Boll, and Wilko Heuten. 2018. Beyond Halo and Wedge: visualizing out-of-view objects on head-mounted virtual and augmented reality devices. In Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '18). ACM, New York, NY, USA, Article 40, 11 pages.

[10] Uwe Gruenefeld, Abdallah El Ali, Wilko Heuten, and Susanne Boll. 2017. Visualizing out-of-view objects in head-mounted augmented reality. In Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '17). ACM, New York, NY, USA, Article 81, 7 pages.

[11] Tiago Gonçalves, Ana Paula Afonso, Maria Beatriz Carmo, Paulo Pombinho. HaloDot: Visualization of the Relevance of Off-Screen Objects. In Proceedings of SIACG, 117–120. 2011.

[12] Sean G. Gustafson, Patrick Baudisch, Carl Gutwin, and Pourang P. Irani. Wedge: clutter-free visualization of off-screen locations. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08), 787-796.

[13] Sean G. Gustafson and Pourang P. Irani. Comparing visualizations for tracking off-screen moving targets. In CHI '07 Extended Abstracts on Human Factors in Computing Systems (CHI EA '07), 2399-2404.

[14] Hart, Sandra G., and Lowell E. Staveland. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. Advances in psychology. Vol. 52. North-Holland, 1988. 139-183.

[15] Jo Hyungeun, Sungjae Hwang, Hyunwoo Park, Jung-hee Ryu. 2011. Around plot: Focus context interface for off-screen objects in 3D environments. Computers & Graphics, Volume 35, Issue 4, 841-85.

[16] Rensis Likert. A technique for the measurement of attitudes. Archives of Psychology, 22 140:55–, 1932.

[17] Patrick Perea, Denis Morand, and Laurence Nigay. 2017. Halo3D: a technique for visualizing off-screen points of interest in mobile augmented reality. In Proceedings of the 29th Conference on l'Interaction Homme-Machine (IHM '17), 43-51.

[18] Perea, P., Morand, D., & Nigay, L. (2017, October). [POSTER] Halo3D: A Technique for Visualizing Off-Screen Points of Interest in Mobile Augmented Reality. In 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct) (pp. 170-175). IEEE.

[19] Mathieu Raynal, Emmanuel Dubois, Benedicte Schmitt. Towards Unication for Pointing Task Evaluation in 3D Desktop Virtual Environment. 1st International Conference on Human Factors in Computing & Informatics (SouthCHI 2013), Jul 2013, Maribor, Slovenia. 7946, pp. 562-580, 2013. E. Catmull. A tutorial on compensation tables. In Computer Graphics, volume 13, pages 1–7. ACM SIGGRAPH, 1979.

[20] Manojit Sarkar and Marc H. Brown. 1992. Graphical fisheye views of graphs. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92), Penny Bauersfeld, John Bennett, and Gene Lynch (Eds.), 83-91.

[21] Allison Sekuler, and Richard Murray, (2001). Amodal completion: A case studying grouping. In T. Shipley & P. Kellman (Eds.), From Fragments to Objects: Segmentation and Grouping in Vision, New York: Elsevier, 265–293.

[22] Matthias Trapp, Lars Schneider, Norman Holz, Jürgen Dollner. Strategies for visualizing points-of-interest of 3D virtual environments on mobile devices. In Proceedings of the sixth international symposium on LBS & TeleCartography. 2009.

[23] Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11), 143-146.