EVALUATION OF AI-SUPPORTED INPUT METHODS IN AUGMENTED REALITY ENVIRONMENT

Akos Nagy
Department of Networks & Digital Media
School of Computer Science & Maths,
ECE, Kingston University
Kingston upon Thames, UK
A.Nagy@kingston.ac.uk

Thomas Lagkas
Department of Computer Science
International Hellenic University Kavala Campus
Greece & South-East European Research Centre
tlagkas@ieee.org

Panagiotis Sarigiannidis
Department of Electrical
and Computer Engineering
University of Western Macedonia
Kozani,Greece
psarigiannidis@uowm.gr

Vasileios Argyriou
Department of Networks & Digital Media
School of Computer Science & Maths,
ECE, Kingston University
Kingston upon Thames, UK
Vasileios.Argyriou@kingston.ac.uk

ABSTRACT

Augmented Reality (AR) solutions are providing tools that could improve applications in the medical and industrial fields. Augmentation can provide additional information in training, visualization, and work scenarios, to increase efficiency, reliability, and safety, while improving communication with other devices and systems on the network. Unfortunately, tasks in these fields often require both hands to execute, reducing the variety of input methods suitable to control AR applications. People with certain physical disabilities, where they are not able to use their hands, are also negatively impacted when using these devices. The goal of this work is to provide novel hand-free interfacing methods, using AR technology, in association with AI support approaches to produce an improved Human-Computer interaction solution.

Augmented Reality, HCI, hands-free interfaces, AI-guided interaction

1 Introduction

Augmented Reality is a rapidly evolving technology that has the potential to transform a wide range of industries, from healthcare [1–3] to manufacturing [4, 5]. By overlaying digital information onto the physical world, provided by systems on a network [6], AR offers a powerful tool for training, visualization, and work scenarios, enabling users to increase efficiency, reliability, and safety. However, one of the challenges facing AR is the limited range of input methods that are available for controlling AR applications, particularly in scenarios where both hands are required to perform a task. This limitation not only impacts the usability of AR but also has significant implications for people who, either require both hands for work processes, observed in industrial or medical fields, or with certain physical disabilities, who may not be able to use their hands to interact with these devices [7, 8].

A variety of input methods are available using AR technologies [9, 10], most of which face different limitations in working environments [11–13]. Hand tracking allows the user to use their hand to interact with virtual objects or perform certain gestures to communicate with the device, which limits the use case when both hands are required while working. Additional peripherals can provide inputs, but these devices add additional cost to purchase, while operating them can require non-natural, or cause limited movement [14–17]. To realise these AR interfaces and solutions, technologies based on computer vision, 3D reconstruction, scene analysis, motion estimation, and object detection are required [18, 19]. Freehand input control offers potential solutions to these challenges, allowing users to interact with AR applications without the need to use their hands. This technology enables users to control AR applications through natural body movements, such as head movements and voice commands [20], freeing up their hands for other tasks. Voice commands can be limited in certain environments, where the general background noise level is too high for reliable voice perception, [21, 22].

As a viable interaction solution, this work focuses on improving head orientation-based input methods by evaluating head- and image-based input solutions assisted by Interpolation and Gravity-Map AI support approaches and compares them with classical input methods, such as mouse and gamepad. The main contributions of this paper are the following:

a) Introduce a novel AI-supported interfacing approach to improve user experience
b) Defined evaluation framework for comparing input methods, including task specification and evaluation metrics
c) Evaluation of AI-support-assisted alternative input approaches and comparison with classical input methods

The paper is structured as follows: Section 2 provides a review off related research and methods. The proposed AI-enabled interaction methods for hands-free applications are detailed in section 3. Section 4 showcases the results using both the classic and the proposed interaction methods in a comparative study. Finally, the conclusion is presented in section 5.

2 RELATED WORK

Human-Computer Interaction (HCI) is a core element of AR use cases. Augmentation allows adaptive HCIs, that form to the human body, or allows itself to be an input device. Numerous studies contribute to finding the most optimal solutions in a wide range of scenarios.

Frusto-Pascual [23] examines different virtual keyboard positions and interaction feedback methods for character input in augmented reality. Two categories of keyboard position conditions are presented: viewpoint bound and non-dominant hand bound conditions. Viewpoint bound conditions are defined based on the position and orientation of the head-mounted display (HMD), while non-dominant hand bound conditions are defined based on the position and orientation of the non-dominant hand. Two forms of visual feedback and guidance for interaction with the keyboard are evaluated: Raycast and Glow. Raycast is a continuous guidance ray to the keyboard, while Glow is a visual representation of the dominant hand index fingertip. Both feedback methods are combined in the "Both combined" condition.

A system proposed by Chen [24] is developed within a VR environment that allows freehand manipulation of 3D objects. The system includes a VR disambiguation menu with preview cubes, which vary in timing and modality based on the techniques. The three promising input modalities, explored for hands-free operation in VR are head gaze, speech, and foot. Also, the timing of disambiguation is explored. Although the methods are proposed related to VR environment, are applicable using AR devices.

The foot-based method presents some limitations that may make it unsuitable, as mentioned before, for some users and certain environments. One significant limitation is that not everyone can use their feet effectively, such as individuals with certain disabilities or injuries that affect their lower extremities. Consequently, the foot-based method may not be an inclusive solution, and alternative selection methods should be considered to accommodate a broader range of users.

Moreover, the foot-based method requires additional equipment to track the user's foot movements accurately, which could increase the complexity of the system and may not be practical in industrial environments. For example, workers in manufacturing or construction sites may be wearing heavy-duty work boots, steel-toed shoes, or other safety equipment that could interfere with the accuracy of the foot-tracking.

Eye gaze-based interfaces for human-computer interaction are becoming increasingly popular and are being used in various sectors such as tele-operation [25]. In tele-operation, the operator's eyes are occupied with monitoring the environment and their hands are busy controlling tasks. Eye gaze-based interfaces provide a natural user interface that is becoming increasingly popular and could become the interface of choice for many devices. Researchers have introduced various eye gaze tracking systems for tele-operation, including those for people with disabilities. A cost-effective and precise eye tracker has been developed, using an eye camera with two IR-emitting LEDs next to the camera lens [26]. A gaze tracking system for tele-operation for people with physical disabilities was introduced, using an analog video-type camera to capture images and a gaze estimation algorithm to control the remote [27]. An eye tracking-based mobile robot for tele-operation was also proposed, using a Microsoft Kinect 3D camera sensor for eye tracking. An eye gaze-based interface for a smart wheelchair was developed to assist people with locomotor disabilities. The system includes image processing for eye tracking and a module for controlling the movement of the wheelchair. It also includes added functionality such as sending messages through a smartphone and controlling electrical devices [28].

A study [13] conducted in 2021 investigates the effectiveness of a gaze-adaptive user interface (UI) for Augmented Reality in comparison to an always-on interface, presented. Previous research has suggested that eye gaze could be a potential solution to the attention dilemma that arises when users must shift their focus between the real world and AR content [11, 12]. The study presents the findings of an empirical investigation that evaluated the performance of gaze-adaptive UI in comparison to the always-on interface. The participants were asked to perform tasks that involved shifting their focus between the real world and virtual content. The results showed that most participants preferred the gaze-adaptive UI and found it less distracting. The gaze UI was faster, more intuitive, and perceived as easier to use when focusing on reality. When focusing on virtual content, the always-on interface was faster, but the user preferences were divided.

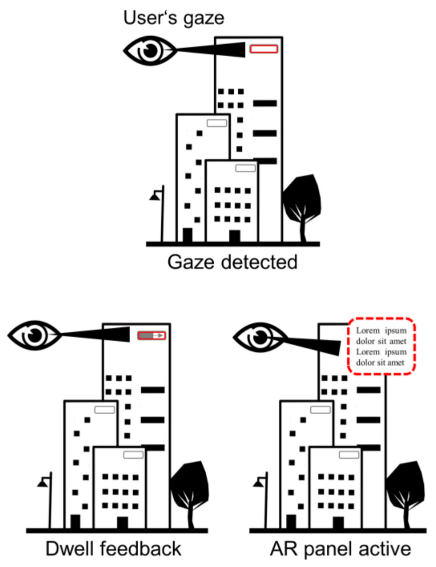

Figure 1: AR - AI Control Support Architecture [13]

The proposed system, as presented with Figure 1, provides additional information when the user gazes at a particular point, triggered by the temporal and spatial activation dimensions. Temporal activation determines how long a user needs to look at an object to trigger the AR panel, while spatial activation defines the size of the area a user needs to gaze at. The paper aims to formalize the properties of a gaze-adaptive AR system and better ground the design and study efforts.

3 METHODOLOGY

Augmented Reality devices have the potential to revolutionize the way industrial workers receive or interact with information. AR devices can provide hands-free access to information and instructions and communication with network-connected devices, allowing workers to focus on their tasks while still having the necessary information at their disposal. This can be achieved through the use of head-mounted displays, smart glasses, or even smartphones. The AR technology overlays digital information in the real-world environment, providing workers with real-time access to important data such as instructions, diagrams, and technical specifications.

The present study focuses on Hand-free controls supported HCI in cases where using hands is not applicable. These cases include work environments where the person requires both hands to perform the tasks required by the processes, i.e.: industrial or medical fields. Other scenarios where due to an injury or a medical condition, where the person is not capable of using their hands also limit the usage of AR devices. Alternatively, a voice command could be used in these

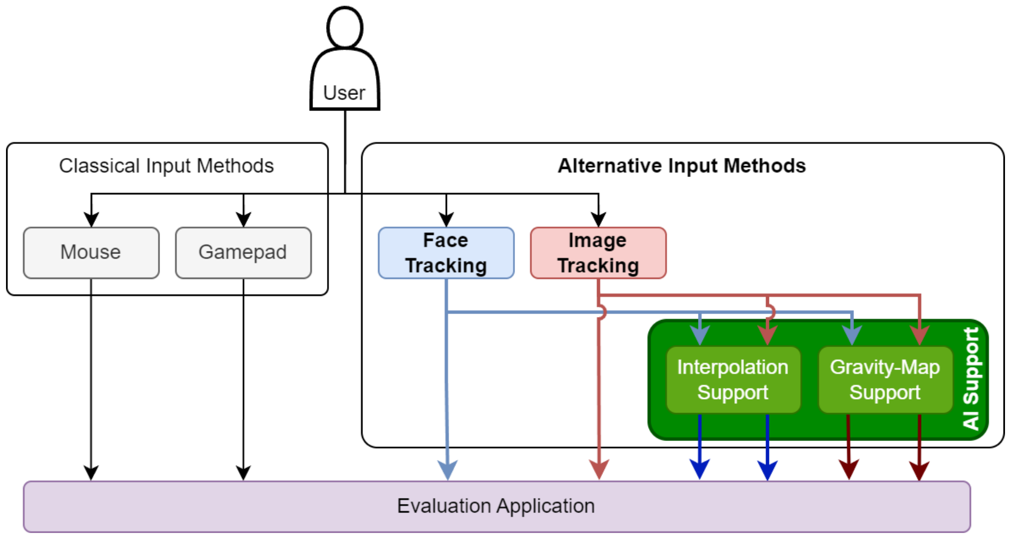

Figure 2: AR - AI Control Support Architecture

cases, but in certain environments, due to the high volume of background noise or the necessity of a quiet environment may not allow a such solution.

The following section describes a methodology of an AI-assisted architecture that supports hand-free control using AR technologies. The comparative study provides an evaluation of the performance and usability of the proposed hand-free control solutions compared to traditional input methods such as keyboard and mouse. The aim is to assess the potential of AR technologies for creating more intuitive and natural user interfaces for various applications.

While the mouse on personal computers, gamepads on console devices, and touch screens on mobile devices are part of the most common human-computer interfaces, in certain scenarios, these approaches might not be the most suitable. Physical limitations may be caused by the working environment, like industrial settings, where personnel requires both hands to perform the required actions, or due to disabilities or injuries, where the person's physical capabilities are impaired, these devices, due to their design, are not suitable for human-computer interactions.

The goal of the following proposed framework for HCI is to design a general pipeline architecture, demonstrated in Figure 2, involving AI-assisted solutions, that allows the use of a variety of alternative input methods while minimizing the discrepancy in usability compared to classical input devices. In our study, we consider two input mechanisms as classical input devices, the mouse, most commonly used with personal computers, and gamepads, which are mainly used with console devices and dominantly for entertainment purposes. As alternative input methods, assisted image and face tracking-based solutions are proposed with applications in immersive environments. The face tracking method uses computer vision algorithms to determine the position and orientation of the user's head, and based on these parameters provides constantly input to the corresponding system, while the image tracking solution again based on computer vision methods, detects the selected image target. and constantly tracks its position and orientation in the 3D world coordinates.

3.1 AI Support Approaches

The proposed AI-assisted solutions, integrated into the architecture, are designed to reduce the difference between the classical input devices and the alternative input solutions or devices. Using the input vector these solutions will alter the data before providing it to the testing application. The current study considers three different solutions as part of the experiment:

Interpolation is a powerful technique in computer graphics and computer vision that can be used to estimate and derive values between known data points. In the context of hand-free control, the interpolation technique allows for the deviation of the input vector toward the intended target based on a specific set of limitations and criteria. The input vector's deviation is determined by considering the distance of the target from the input vector and the original orientation of the input vector.

This balance between forced, where the AI is influencing the input method, and free control, where there is no influence, is crucial in ensuring that the system is both responsive and intuitive to the user's input, while also maintaining a level of control over the target object. The technique allows for smooth and gradual adjustments to the input vector so that the target can be reached in an efficient and accurate manner.

The Pseudo code for the implementation is as seen in Algorithm 1.

Algorithm 1 InterpolationPrediction

```
Require:
 1: $start$: current coordinate,
 2: $moveVec$: current input vector ,
 3: $influence$, default(0.8f)
 4: $number$: number of iterations, how many possible future positions need to be predicted; default(1)
Ensure: $p$: an array of predicted influenced points
 5: function InterpolationPrediction($start, moveVec, target, influence = 0.8, number = 1$)
 6:     $p \leftarrow [start, start + moveVec]$
 7:
 8:     for $i \leftarrow 0$ to $number$ do
 9:         $c \leftarrow p[\text{length}(p) - 2]$
10:         $tVec \leftarrow target - c$
11:
12:         if $moveVec.\text{mag}() == 0 \vee tVec.\text{mag}() == 0$ then
13:             append $c$ to $p$
14:             continue
15:         end if
16:
17:         $tVecNorm \leftarrow \text{normalize}(tVec)$
18:         $moveVecNorm \leftarrow \text{normalize}(moveVec)$
19:
20:         $mod \leftarrow \max(tVecNorm.\text{dot}(nVecNorm), influence) - influence$
21:         $mod \leftarrow mod \times (1/(1 - influence))$
22:
23:         $fVec \leftarrow \text{lerp}(moveVecNorm, tVecNorm, mod)$
24:         $modDist \leftarrow \min(tVec.\text{mag}(), moveVec.\text{mag}())$
25:         $next \leftarrow fVec \times modDist + c$
26:         append $next$ to $p$
27:     end for
28:
29:     return elements of $p$ starting from index 1
30: end function
```

The **Gravity-Map** approach solution for controlling input vectors is a more flexible approach compared to the Interpolation approach and can be used to control the deviation of the input vector toward multiple target objects. In this approach, the interface is divided into two areas, with one area being defined as the area of effect, and the other being the unaffected area. The area of effect is determined by the proximity of the target objects to the current focus position. When the cursor position is within the area of effect, the input vector is deviated toward the intended target object based on its distance and original orientation.

The Gravity-Map approach is different from other interpolation techniques in that it considers the relationship between the input vector and multiple target objects, rather than just one target object at a time. This allows for a more intuitive control experience, as the input vector is pulled towards the target objects that are closest and within reach. Furthermore, dividing the interface into two areas provides a balance between forced and free control. In the area of effect, the input vector is automatically directed toward the target objects, providing assistance for users. However, outside of this area, the input vector remains unaltered, giving users the freedom to move and control the cursor as they see fit.

The Pseudo code for the implementation is as seen in Algorithm 2.

3.2 Data-collection Methods

Data collection has been performed via a data evaluation application (Figure 3), which encompasses three modes, each with its own task to perform and metrics to evaluate. The study involved 20 participants.

Locate Mode

In this mode, a series of static targets are displayed within the 3D environment. The participant's objective is to align the focus point, which is represented with a crosshair in the application, with the target. The task consists of multiple

Algorithm 2 Calculate influence vector at coordinate

```
Require:
 1: targets (list of rectangles with $x, y$, width, height attributes),
 2: $PX$ (x coordinate of a point),
 3: $PY$ (y coordinate of a point),
 4: $influenceDistance$ (the max distance a given target is allowed to influence the outcome)
Ensure: influence vector at point described by $PX$ and $PY$
 5: function GravityMapInfluence($targets$, $PX$, $PY$, $influenceDistance$)
 6:     $retInfluence \leftarrow (0, 0)$;
 7:     for $i \leftarrow 0$ to length($targets$) do
 8:         $closestPoint \leftarrow$ calculate closest point within the target area to $(PX, PY)$;
 9:         if $(PX, PY)$ is within the target area then
10:             return $(0, 0)$;
11:         end if
12:         if distance between $closestPoint$ and $(PX, PY)$ is greater than $influenceDistance$ then
13:             continue;
14:         end if
15:         $direction \leftarrow$ calculate direction vector towards $closestPoint$;
16:         $distanceRatio \leftarrow$ distance between $closestPoint$ and $(PX, PY)$ / $influenceDistance$;
17:         $influence \leftarrow direction \times (1 - distanceRatio^2)$;
18:         $retInfluence \leftarrow retInfluence + influence$;
19:     end for
20:     return retInfluence;
21: end function
```

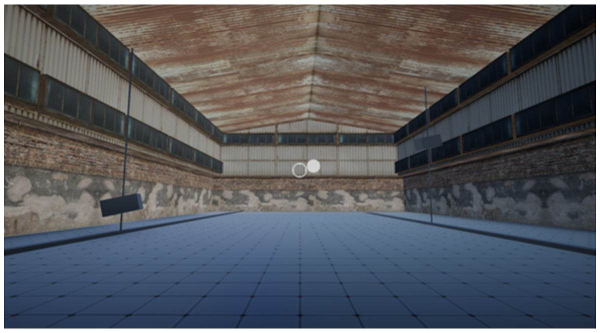

Figure 3: Evaluation application (in follow mode) screenshot with crosshair and target (white)

targets, each of which constitutes a separate subtask, organized in a sequential manner with only one target being visible at any given time. In this mode, we measure the average time it takes for a participant to reach the target position with the various input methods and solutions. Each target has a defined window of availability, beyond which it will be removed, and the corresponding subtask will be considered as failed.

Select Mode

In this mode, the participant's task is to precisely position the crosshair, which serves as the center of the screen, onto the static targets in a 3D environment. The evaluation consists of a series of subtasks, each of which involves positioning the crosshair onto a target and maintaining it there continuously for a set amount of time. The targets are presented in a sequential manner, with only one target being visible at a time.

The accuracy of the various input methods and solutions was assessed by conducting experiments and measuring the ability of the input method to stay within a designated region of the screen. The success rate of each input method was calculated based on the number of successfully completed subtasks, and the extra time spent on the target was recorded for subtasks that took longer than necessary to complete.

Follow Mode

In the Follow Mode, the participants are tasked with tracking a moving target, as opposed to a static target in the other modes. The targets move along a predefined path at predetermined speeds to control the experiment and determine the relationship between success and speed. The objective is to position the crosshair, which serves as the center of the screen, onto the target and follow it along the designated path between the start and end marks. This mode evaluates the precision of the input solutions in a dynamic target scenario, similar to the Select Mode.

4 Results

4.1 Quantitative Metrics

In this section, we will discuss the various metrics used for the Quantitative results. In the section, the focus point refers to the center of the screen, represented by a crosshair in the testing application, as displayed on Figure 4.

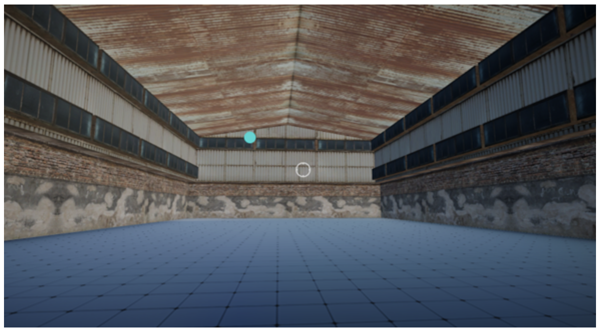

Figure 4: Evaluation application screenshot with crosshair and target (cyan)

Locate Mode

Avg. Reach Time: Let S_{locate} be the set of sub-tasks and let n be the total number of sub-tasks in S_{locate}. Let S'_{locate} be the subset of S_{locate} consisting of the sub-tasks that were completed successfully. A sub-task constitutes success if the focus point overlaps with the target object. For every sub-task, we record two timestamps, $TargetAppearedTimeStamp$, when the target object appeared on the game, and $TargetReachedTimeStamp$, when the focus point reached the target object. From these values, we calculate a $ReachTime$ for every sub-task, which is the difference between the previous two timestamps.

$$\text{Avg. Reach Time} = \frac{1}{n} \sum_{s \in S'_{locate}} s_{ReachTime}$$

$$s_{ReachTime} = s_{TargetAppearedTimeStamp} - s_{TargetReachedTimeStamp}$$

Target Reached %: Let S_{locate} be the set of sub-tasks and let n be the total number of sub-tasks in S_{locate}. Let S'_{locate} be the subset of S_{locate} consisting of the sub-tasks that were completed successfully. This formula calculates the average of successful sub-tasks by dividing the sum of all successful sub-tasks by the total number of sub-tasks in S_{locate}.

$$\text{Target Reached \%} = \frac{1}{n} \sum_{s \in S'_{locate}} s$$

Select Mode:

Avg. Extra Time Required: Let S_{Select} be the set of sub-tasks in Locate Mode and let n be the total number of sub-tasks in S_{Select}. Let S'_{Select} be the subset of S_{Select} consisting of the sub-tasks that were completed successfully. A sub-task constitutes success if the focus point overlaps with the target object for a set $SelectTarget$ time.

$$\text{Avg. Extra Time Required} = \tfrac{1}{n} \textstyle\sum_{s \in S'_{select}} (s_{T_{overlap}} - SelectTarget)$$

Target Selected %: Let S_{Select} be the set of sub-tasks in Select Mode and let n be the total number of sub-tasks in S_{Select}. Let S'_{Select} be the subset of S_{Select} consisting of the sub-tasks that were completed successfully. This formula calculates the average of successful sub-tasks by dividing the sum of all successful sub-tasks by the total number of sub-tasks in S_{Select}.

$$\text{Target Selected \%} = \frac{1}{n} \sum_{s \in S'_{select}} s$$

Follow Mode:

Avg. Follow %: Let S_{Follow} be the set of sub-tasks in Follow Mode and let n be the total number of sub-tasks in S_{Follow}. Let S'_{Follow} be the subset of S_{Follow} consisting of the sub-tasks that were completed successfully. This formula calculates the average percentage of overlap time during the successful sub-tasks by dividing the sum of all percentages of overlap time for sub-tasks by the total number of sub-tasks in S'_{Follow}. The percentage of overlap time for a sub-task is the overlap time divided by the fly-time of the sub-tasks. Fly-time is the total amount of time the target object is available on the screen, and overlap time is the time the focus point is overlapping with the target object during the sub-task.

$$\text{Avg. Follow \%} = \frac{1}{n} \sum_{s \in S'_{Follow}} (s_{OverlapTime} / s_{FlyTime})$$

Moving Target Touched % : Let S_{Follow} be the set of sub-tasks in Follow Mode and let n be the total number of sub-tasks in S_{Follow}. Let S'_{Follow} be the subset of S_{Follow} consisting of the sub-tasks that were completed successfully. This formula calculates the average of successful sub-tasks by dividing the sum of all successful sub-tasks by the total number of sub-tasks in S_{Follow}. A sub-task constitutes success if the focus point overlaps with the target object.

$$\text{Moving Target Touched \%} = \frac{1}{n} \sum_{s \in S'_{Follow}} s$$

4.2 Data evaluation

The evaluated data is separated into three tables, Table 1, Table 2 and Table 3, one for each of the data-collection modes, displaying the previously discussed metrics associated with the modes. Each table is also separated into two sections to showcase the differences between the input methods without AI support, as baseline values, and with AI support, to evaluate the efficiency of the support approaches. Each section has the best-performing values highlighted.

The results show that the Mouse input method is the overall best-performing among the input methods evaluated in this study. Among the AI-supported input methods, the Head movement-based, Gravity-Map approach performed the best, in certain aspects even better than the Mouse input.

Locate Mode

While the Mouse input method performed 48% better than the overall average in terms of Reach Time, it's only 18% better than the Gravity-Map assisted Head input, which has gained a 42% improvement compared to not having AI support. Image-based input was the worst-performing, not AI-supported method, and while the Gravity-Map AI-support approach slightly improves the method, using the Interpolation approach increased the Average Reach Time by 18%.

Table 1: OUTCOMES FOR ALL THE INPUT METHODS IN LOCATE MODE

	Input	Target Reached %	Avg. Reach Time
No AI Support	Mouse	**98.3%**	**0.89s**
	Gamepad	97.8%	1.34s
	Head	96.0%	1.86s
	Image	75.5%	2.17s
AI Support	Head - Interpolation	96.2%	1.78s
	Head - Gravity-Map	**99.78%**	**1.08s**
	Image - Interpolation	76.1%	2.56s
	Image - Gravity-Map	83.6%	1.82s
	Overall Average	90.4%	1.69s

Select Mode

The Mouse input method performed over 9X better than the overall average in terms of Extra Time Required, due to the fact that a mouse does not have continuous input like the alternative input methods. While using the alternative methods, there's a continuous movement, even in the target (head, image) is seemingly motionless. The Head-based input using the Gravity-Map approach performed over 4X better, and became the third fastest solution, than the Head-based input without AI support, which required 2X more time to perform the tasks than the overall average, being the overall slowest solution. In terms of the Select Mode, a higher Extra Time Required value indicates a less stable input solution when required to dwell on a fixed point.

Table 2: OUTCOMES FOR ALL THE INPUT METHODS IN SELECT MODE

	Input	Target Selected %	Avg. Extra Time Required
No AI Support	Mouse	**100%**	**0.042s**
	Gamepad	95.8%	0.099s
	Head	85.5%	0.779s
	Image	70.5%	0.470s
AI Support	Head - Interpolation	81.9%	0.659s
	Head - Gravity-Map	**99.6%**	**0.178s**
	Image - Interpolation	72.6%	0.494s
	Image - Gravity-Map	90.1%	0.343s
	Overall Average	87.0%	0.383s

Follow Mode

The worst-performing solution was the Image-based input assisted by the Interpolation AI-support, resulting in only 50% Avg. Follow % compared to the average and 20%-25%, compared to the best-performing solutions. The two best-performing solutions were the Mouse and Head-based input method, assisted by the Gravity-Map AI-support approach. Only these two solutions were able to perform better than the overall average, in terms of Avg. Follow %, with the Mouse-based input method performing 16% better than the second-best solution.

Table 3: OUTCOMES FOR ALL THE INPUT METHODS IN FOLLOW MODE

	Input	Moving Target Touched %	Avg. Follow %
No AI Support	Mouse	**99.4%**	**48.1%**
	Gamepad	99.1%	19.3%
	Head	83.3%	14.2%
	Image	75.5%	17.9%
AI Support	Head - Interpolation	81.5%	11.9%
	Head - Gravity-Map	**99.5%**	**41.3%**
	Image - Interpolation	66.2%	11.1%
	Image - Gravity-Map	83.9%	14.9%
	Overall Average	86.0%	22.3%

Overview

We consider Target Reached %, Target Selected % and Moving Target Touched % as measures of success within their own data-collection mode, for all of these values relate to the ratio between successful and failed sub-tasks. Figure 5 displays the relative success rate for each input method. Input methods are shown along the X-axis while the success rate is along the Y-axis. Vertical bars represent the success rate for each input method with different colors correlating to different data-collection modes. Horizontal lines represent the average success values for each of the input methods. We calculate this value by averaging Target Reached %, Target Selected %, and Moving Target Touched % values. Each color represents each input method. Input methods without AI support are represented by dashed lines, while AI-supported methods are represented by dotted lines, to further increase readability. The diagram was also limited to the [65-100] range on the Y-axis, to emphasize the differences, that on a wider range would be not clear.

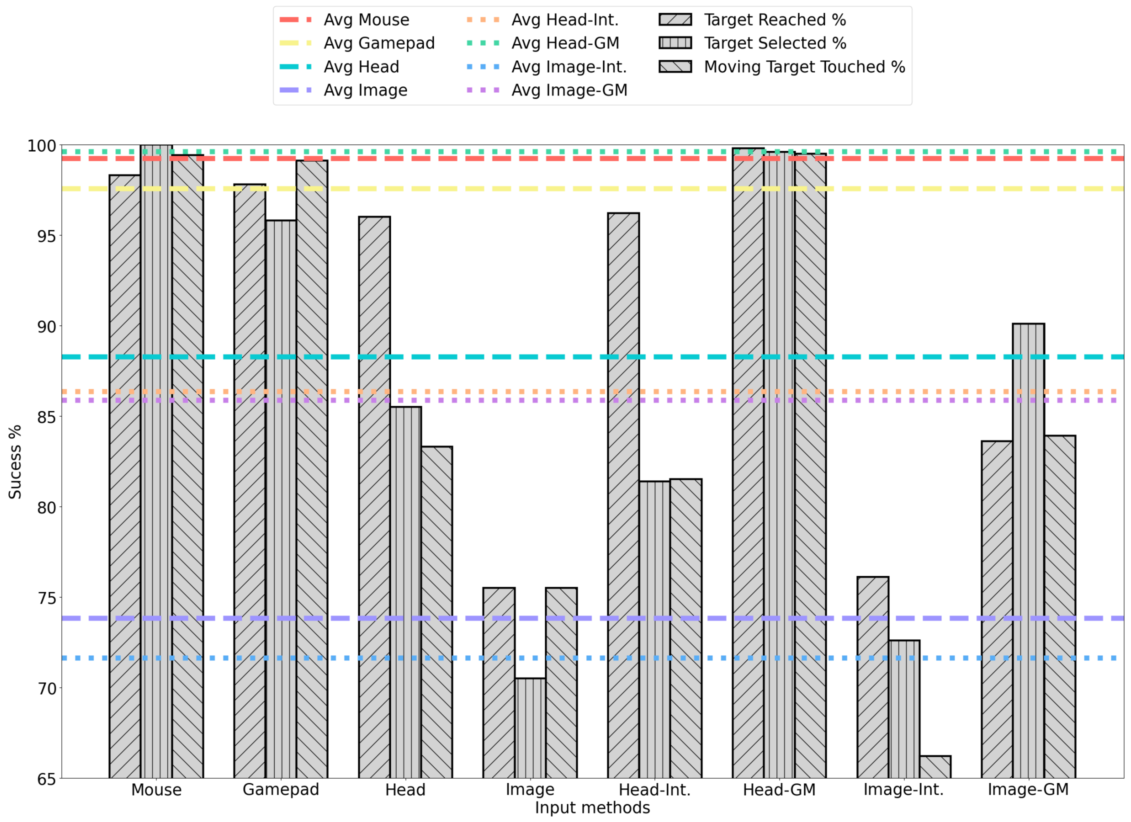

Figure 5: Graphical overview of "Success %" for each input-method and their relative average values

5 CONCLUSIONS

Considering that mouse (and keyboard) are the main input peripherals used in everyday life when controlling a personal computer, it was expected that the input method will produce the best performance. Based on the results we can conclude that using the Gravity-Map AI Support to aid alternative input methods, such as face tracking, significantly improves the viability of the method. The results show, that Head-tracking based input using the Gravity-Map AI support approach performs the closest to the mouse-based input. Taking into account that most of the participants only saw head tracking-based input methods first time during the testing process, we can presume that increased familiarity with the technology will lead to higher performance.

6 ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon Europe Research and Innovation programme under grant agreement No. 101070181.

References

[1] D. Parsons and K. MacCallum, “Current perspectives on augmented reality in medical education: Applications, affordances and limitations,” *Advances in Medical Education and Practice*, vol. 12, pp. 77–91, 2021.

[2] C. Dennler, D. E. Bauer, and A.-G. Scheibler, “Augmented reality in the operating room: a clinical feasibility study,” *BMC Musculoskeletal Disorders*, vol. 22, no. 1, p. 451, 2021.

[3] S. Hell and V. Argyriou, “Machine learning architectures to predict motion sickness using a virtual reality rollercoaster simulation tool,” in *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pp. 153–156, 2018.

[4] V. Reljić, I. Milenković, S. Dudić, J. Šulc, and B. Bajči, “Augmented reality applications in industry 4.0 environment,” *Applied Sciences*, vol. 11, no. 12, 2021.

[5] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, “Industry 4.0 and industry 5.0—inception, conception and perception,” *Journal of Manufacturing Systems*, vol. 61, pp. 530–535, 2021.

[6] J. C. Kim, T. H. Laine, and C. Åhlund, “Multimodal interaction systems based on internet of things and augmented reality: A systematic literature review,” *Applied Sciences*, vol. 11, no. 4, 2021.

[7] M. Žilak, Car, and I. Čuljak, “A systematic literature review of handheld augmented reality solutions for people with disabilities,” *Sensors*, vol. 22, no. 20, 2022.

[8] Z. Rashid, J. Melià-Seguí, R. Pous, and E. Peig, “Using augmented reality and internet of things to improve accessibility of people with motor disabilities in the context of smart cities,” *Future Generation Computer Systems*, vol. 76, pp. 248–261, 2017.

[9] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana, “Virtual object manipulation on a table-top ar environment,” in *Proceedings of the International Symposium on Augmented Reality (ISAR 2000)*, pp. 111–119, Oct. 2000.

[10] M. Billinghurst, H. Kato, and S. Myojin, “Advanced interaction techniques for augmented reality applications,” *The Human Interface Technology New Zealand (HIT Lab NZ)*, vol. 1, 2001.

[11] K. Pfeuffer and et al., “ARtention: a design space for gaze-adaptive user interfaces in augmented reality,” *Comput. Graph.*, vol. 95, pp. 1–12, 2021.

[12] R. Vertegaal *et al.*, “Attentive user interfaces,” *Communications of the ACM*, vol. 46, no. 3, pp. 30–33, 2003.

[13] R. Piening, K. Pfeuffer, A. Esteves, T. Mittermeier, S. Prange, P. Schröder, and F. Alt, “Looking for info: Evaluation of gaze based information retrieval in augmented reality,” in *Human-Computer Interaction – INTERACT 2021*, (Cham), pp. 544–565, Springer International Publishing, 2021.

[14] M. Whitlock, E. Harnner, J. R. Brubaker, S. Kane, and D. A. Szafir, “Interacting with distant objects in augmented reality,” in *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 42–48, IEEE, 2018.

[15] F. Müller, J. McManus, S. Günther, M. Schmitz, M. Mühlhäuser, and M. Funk, “Mind the tap: Assessing foot-taps for interacting with head-mounted displays,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, (New York, NY, USA), pp. 477:1–477:13, 2019.

[16] V. Bloom, D. Makris, and V. Argyriou, “Clustered spatio-temporal manifolds for online action recognition,” in *2014 22nd International Conference on Pattern Recognition*, pp. 3963–3968, 2014.

[17] R. J. K. Jacob, *Eye Tracking in Advanced Interface Design*, pp. 258–290. 1995.

[18] V. Argyriou, M. Petrou, and S. Barsky, “Photometric stereo with an arbitrary number of illuminants,” *Computer Vision and Image Understanding*, vol. 114, no. 8, pp. 887–900, 2010.

[19] V. Argyriou, “Performance study of gradient correlation for sub-pixel motion estimation in the frequency domain,” *IEE Proceedings - Vision, Image and Signal Processing*, vol. 152, pp. 107–114(7), February 2005.

[20] M. Bates, “Models of natural language understanding,” *Proc. Natl. Acad. Sci. USA*, vol. 92, no. 22, p. 9977, 1995.

[21] F. Manuri and G. Piumatti, “A preliminary study of a hybrid user interface for augmented reality applications,” IEEE, 8 2015.

[22] M. H. Ali, M. M. Jaber, S. K. Abd, A. Rehman, M. J. Awan, D. Vitkutė-Adžgauskienė, R. Damaševičius, and S. A. Bahaj, “Harris hawks sparse auto-encoder networks for automatic speech recognition system,” *Applied Sciences*, vol. 12, no. 3, 2022.

[23] M. Frutos-Pascual, C. Gale, C. Harrison, Jake M.and Creed, and I. Williams, “Character input in augmented reality: An evaluation of keyboard position and interaction visualisation for head-mounted displays,” in *Human-Computer Interaction – INTERACT 2021*, (Cham), pp. 480–501, Springer International Publishing, 2021.

[24] L. Chen, R. Balakrishnan, and T. Grossman, “Disambiguation techniques for freehand object manipulations in virtual reality,” in *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 285–292, Mar. 2020.

[25] S. Mahmud, X. Lin, and J.-H. Kim, “Interface for human machine interaction for assistant devices: A review,” in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0768–0773, 2020.

[26] H. Ho, “Low cost and better accuracy eye tracker,” in *2014 International Symposium on Next-Generation Electronics (ISNE)*, pp. 1–2, May 2014.

[27] M. Yu, X. Wang, Y. Lin, and X. Bai, “Gaze tracking system for teleoperation,” in *The 26th Chinese Control and Decision Conference (2014 CCDC)*, pp. 4617–4622, May 2014.

[28] D. Gego, C. Carreto, and L. Figueiredo, “Teleoperation of a mobile robot based on eye-gaze tracking,” in *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, (Lisbon), 2017.