Five Ways Function Models Enable Accessible Mixed Reality Interfaces

Per Ola Kristensson
University of Cambridge

ABSTRACT

Mixed reality interfaces are used in a variety of contexts, and only some of these can be fully anticipated or controlled by the designer. Accessible mixed reality interfaces further exacerbate this design problem since users will have a variety of individual capabilities, limitations, needs, wants, and values, and frequently accessible interfaces are used in more complex interaction contexts that may involve close family or assistants. In addition, it is often difficult to carry out extensive user research or involve end-users throughout the design process. This paper explains how function models, which model the organization of high-level functions a system has to carry out, can be helpful as a complementary design method when designing accessible mixed reality interfaces.

1 INTRODUCTION

Designing mixed reality user interfaces is challenging as they are part of highly complex, tightly coupled systems where the key emergent property is *interactivity*. As it is an emergent quality, interactivity in itself cannot be directly created by the designer—it has to be implicitly designed. This challenges the design team to create a more general *model* of interaction [10, 19] that will give rise to effective, efficient, and safe interactivity in a variety of future interaction contexts, only some of which can be fully anticipated at design time.

This is illustrated in Figure 1. The design team creates a system using design knowledge, such as elicited requirements, technical know-how, experience, results from user research, and so on. A part of this system design is determining the functions that need to be implemented and their underlying parameters. Some of these parameters are under the designer's control and can give rise to optimization. Parameters that are not governable can sometimes be analyzed using some form of model and can thus be studied in sensitivity analyses to provide an understanding of how a system would react to hypothetical parameter settings, such as motor tremor.

As indicated in Figure 1, the design activity yielding a system happens at a certain time. However, the system itself will be deployed and used in the future in many contexts. A user achieving their goals with such a deployed system will give rise to interactivity, which is emergent from the joint human-system. However, while it is emergent and arising in the future, this does not men it is not possible control qualities of interactivity. Aspects, such as the overall presentation of the user interface, options for interactivity presented to the user, the design of interaction techniques, and so on, are all qualities of interaction that a design team can study, modify, tune, and optimize. However, there are also qualities of interaction that are uncontrollable, such as users' individual preferences, specific use contexts, and uncertainty about user intent. The design challenge is therefore to create a system that can yield good interactivity in the future for as many people as possible by indirectly influencing such interactivity through a model, which includes an understanding of key functions and their underpinning parameters.

While this central user interface design challenge is not unique to mixed reality interfaces, it is particularly challenging for such interfaces for two reasons. First, the interaction context is fundamentally *uncontrollable*—since mixed reality devices blend physical reality and virtual content it is not possible to directly control what the user observes through the display. Second, mixed reality interaction mechanisms based on eye gaze, head, finger, or hand tracking are fundamentally *uncertain* as they require the system to infer user intent from noisy observations of user behavior.

In addition to these challenges, accessible mixed reality interfaces require designers to enable interaction for highly heterogeneous groups of users with their own individual capabilities, limitations, needs, wants, values, and practices. Further, some accessible interfaces, such as augmentative and alternative communication devices for nonspeaking individuals with motor disabilities, frequently involve additional people, such as speaking partners, close family, or assistants.

Following recent calls to action (e.g. [13]), researchers have begun to identify requirements (e.g. [21] and barriers (e.g. [22]) hindering inclusive mixed reality interface design. A recent review of accessibility in virtual and augmented reality [2] revealed that three major challenges are (1) addressing the diversity of the user population; (2) providing guidelines and design tools; and (3) overcoming obstacles in empirical research. Further work in the area of eye typing posited that a fundamental reason for lack of substantial progress in the field was a failure to engage with end-users and viewing the design challenge as a system design problem [4].

Recent work in human-computer interaction (HCI) has investigated the use of established design engineering methods commonly used to create electromechanical systems (e.g. [11]) to help design HCI systems for nonspeaking individuals with motor disabilities [7, 20]. Such work has made progress by modeling such systems at the functional level by creating function models that explain how key functions are organized in the system. This allows a design team to parameterize these functions and explore emergent interactivity issues at an early stage of the design [7, 20]. Note that such simulation is not in itself new, in fact, KLM-GOMS [1] is a popular method for performing ballpark performance simulations. In addition, HCI researchers have investigated exploring optimal parameter settings through simulation before, in particular for statistical decoder-based interfaces (e.g. [3, 5, 8, 15–18]. However, such prior simulation approaches have focused on systematically exploring different solution-oriented parameter values. That is, the simulations have been carried out after the design team have made a final determination of their overall implementation and the simulations are not used to explore more general what-if scenarios but to essentially fine-tune a final system design.

In contrast, a function model describes a system at a solution-neutral level insofar as possible by addressing *what* needs to be done—the functions—as opposed to *how* to do it—a specific implementation. In addition, a function model approach can help explain interaction *mechanisms* that give rise to observed user performance. For example, a large-scale empirical study of mobile typing revealed that word predictions are in themselves unlikely to improve user performance [12]. By modeling the interaction of users typing with the presence of word prediction as a function model, it is possible to identify the key parameters underpinning such interaction and simulate a wide range of outcomes [9]. This approach made it possible to explain that the efficacy of word predictions for able-bodied users is strongly tied to the individual user's strategy in using them, and very few strategies would be able to eke out a notable performance

2771-1110/24/$31.00 ©2024 IEEE
DOI 10.1109/ISMAR-Adjunct64951.2024.00054

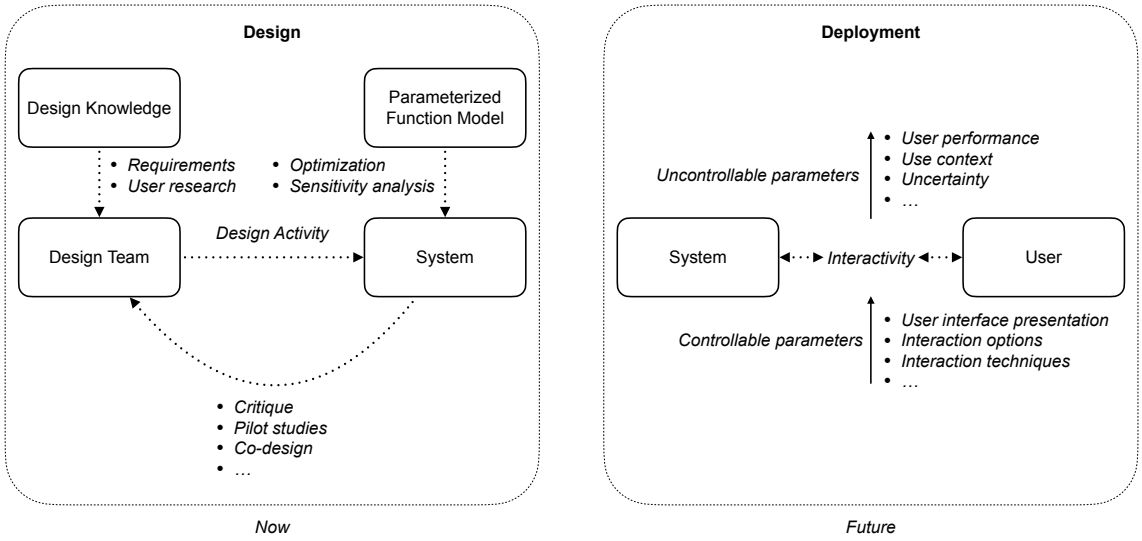

Figure 1: An illustration of some of the inherent challenges in designing mixed reality interfaces for use in future situations that cannot be fully anticipated by the design team.

gain on average [9].

This work follows recent work in attempting to introduce function models either as part of a design [14], or as a general methodology for mixed reality design [6]. We here give a brief explanation of what is meant by a function model and explain two methods for arriving at such descriptions. We then proceed to explain five ways in which function models may help design accessible mixed reality interfaces: (1) making design problems tractable; (2) understanding what needs to be done; (3) knowing what to look for; (4) systematically exploring possibilities; and (5) linking technology advances to user needs.

2 Systems

A *system* encompasses all concerns that are relevant to consider for the system to able to fulfill the user's goals. This includes hardware, software, user interfaces, but also any individuals involved, machine learning algorithms and their training data, and applicable regulations, norms, and organizational practices.

Unbounded, a system is infinite and will by definition encompass everything. A *system boundary* is a delineation of a system to ensure a system is tractable. A system boundary thus ensures that a design team clearly understands what should be included and modeled by a function model and what is excluded.

It is important to realize that an accessible mixed reality interface forms part of a wider system, and as a such it will most likely include concerns beyond immediate hardware and software. Further, as a system, it can operate at different levels of abstraction. Everything does not need to be modeled in detail but it is important that anything that could be vital to the design can in principle be modeled at a level of detail that is useful for the design team.

Figure 2 shows an example of a system boundary for a hypothetical machine-learning driven object selection technique in a mixed reality interface. The dashed rounded rectangle indicate the system boundary. Within the boundary there are six high-level entities: (1) the user, which have capabilities, limitations, goals, needs, wants, and values; (2) the headset, which gives rise to concerns around interaction and ergonomics; (3) datasets, which the user will interact with; (4) the interaction technique itself, which by interpreting observations of user behavior will infer hypotheses of user intent and act upon them; (5) a machine learning system, which bestows the interaction technique with the ability to infer user intent; and

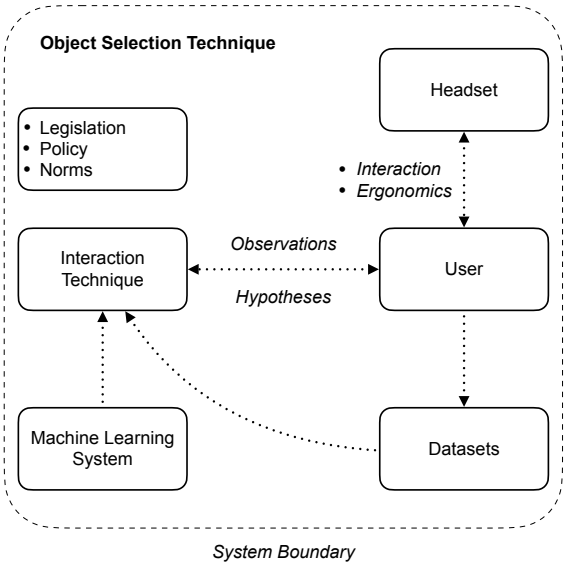

Figure 2: An example of a system boundary for an initial design of an object selection technique in a mixed reality interface.

(6) governance, such as applicable legislation, policies, and norms, among other concerns.

3 Function Models

A *function model* is an organization of a set of *functions* that must be carried out by a system for it to be able to carry out its *overall function*—its purpose, which typically involves achieving users' goals in an efficient and safe manner.

It is important to understand what is meant by a function in this context. A *function* is an abstraction of something that the system must be able to carry out. It explains *what* needs to be done. Examples of hypothetical functions include: `Predict Word`, `Select Object`, `Undo Previous Action`.

Note that while a function explains an aspect of what a system

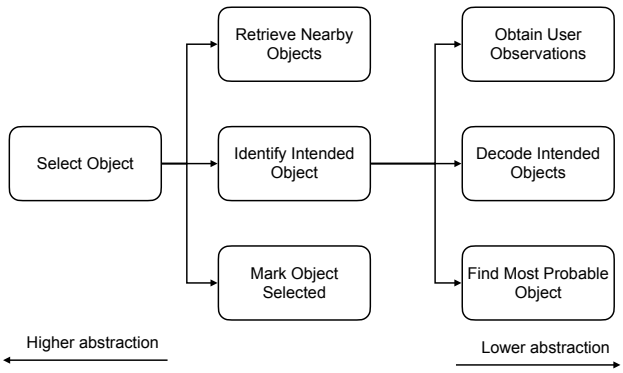

Figure 3: An example of a FAST diagram for high-level modeling of an object selection task in a mixed reality interface. The horizontal axis denotes abstraction while the vertical axis denotes the sequencing of function execution.

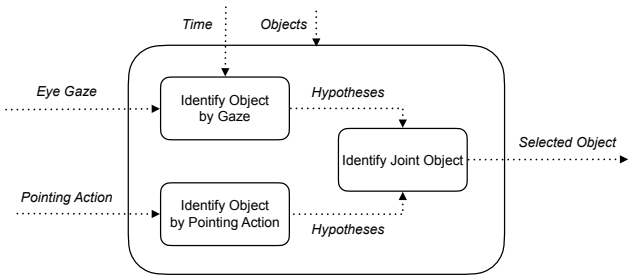

Figure 4: An example of a function structure diagram for an object selection task in a mixed reality interface. The arrows indicate the flow of signals between the functions.

must be able to achieve, it does not explain how to do it. A *function carrier* is a possible solution to a function, that is, an implementation that can execute a function. As such, a function carrier explains *how* to carry out a function. Since there are normally many ways to achieve a function, a function tends to be map to several candidate function carriers. For example, a function such as `Predict Word` can map to several function carriers, including *Prefix Tree*, *Token-passing Decoder*, and so on.

While it may seem initially counter-intuitive, it is useful to consider a design in terms of functions first. This means we refrain from exploring and assigning function carriers until we have fully understood all necessary functions in the system. Understanding the essential functions in a system, and the ways they are interconnected to realize an overall function, allows the construction of diagrams that elucidate the organization of functions in a system.

Such a diagram is called a *function model*, and it can be constructed in various ways, depending on the need of the design team.

3.1 Decomposing Functions in Terms of Abstraction

One way to construct such a function model diagram is to realize that functions have order in terms of which abstraction level they are situated at. In other words, there are higher-order functions that are at a higher level of abstraction than lower-order functions, which are more concrete.

Function Analysis Systems Technique (FAST) diagrams leverage this property by constructing a function model diagram where the horizontal axis models abstraction. Higher-order functions to the left are at a higher level of abstraction than lower-order functions to the right, which are increasingly concrete as they go further to the right. The vertical axis denotes the ordering of the functions in terms of time.

Figure 3 shows an example of a FAST diagram for a hypothetical mixed reality interface for an object selection task in mixed reality. The horizontal axis is mapped to abstraction while the vertical axis is mapped to a relative point in time when each subfunction at the same abstraction level is executed. The leftmost function `Select Object` denotes the overall function for this task. As a function it tells us what we need to do, but not how to do it. As it is leftmost in the diagram it is also of the highest order of abstraction.

The overall function `Select Object` is then further decomposed into its key subfunctions `Retrieve Nearby Objects`, `Identify Intended Object`, and `Mark Object Selected`. These functions are intended to be executed in order, from top to bottom. Note they are more concrete than the overall function as they provide further details on how to carry out the overall function. We can now ask *how* do we `Select Object` and in response observe that by following the arrows to the right of `Select Object` we do so by executing `Retrieve Nearby Objects`, `Identify Intended Object`, and `Mark Object Selected` in order. Conversely, we can also ask *why* do we, for example, `Mark Object Selected`, and by follow the arrow to the left observe that we do this because we need to `Select Object`. In general, we can ask why a function is in the diagram by going to the left and how we can address a function by going to the right.

As we go further to the right we gradually reduce the level of abstraction by being increasingly concrete. As an example, `Identify Intended Object` is further decomposed into subfunctions `Obtain User Observations`, `Decode Intended Objects`, and `Find Most Probable Object`. As we continue to decompose functions in this way, at some point we transition from being solution-neutral to prescribing a specific solution principle. In this example, the decomposition of `Identify Intended Object` suggests an interaction technique solution that receives input sensor signals about the user's intention, such as eye gaze and pointing gestures, and uses a decoder to order arrive at an ordering of candidate objects the user may intend to select, and then chooses the most probable object. Although this function decomposition is not prescribing a precise solution, since the choices of input signals, statistical decoder, training data, and criteria for finding the intended object are not described, it already explicitly rules out some solutions to this interaction problem, notably any solution that does not view this interaction problem as an inference problem.

FAST diagrams are particularly useful early in the design process to help the design team elaborate on all essential functions in the system. A design team can start by considering the key overall functions that a system must be able achieve, and gradually decompose these functions further. In this way, FAST diagrams end up reducing the risk of a design team failing to take into account critical functions. For this reason, FAST diagrams are frequently used in tandem with a requirements specification process.

It ends up being a professional judgment call when to decide to stop the process of further elaborating on the functions in a FAST diagram. At some point, as alluded to above, the decomposition process becomes solution-specific and prescribes particular strategies to address the overall function.

3.2 Decomposing Functions in Terms of Flow

Another way to construct a function model diagram is to model the flow between functions. Flow can be operationalized in different ways. The traditional electromechanical design view considers flows of energy, materials, and signals. In mixed reality interface design tasks we are typically primarily interested in how signals flow between functions in a system design.

Figure 4 shows an example of a function structure for the task of selecting an object. The overall function, which is delineated

by the large rounded rectangle, is `Select Object`. It has been decomposed into three key subfunctions: (1) `Identify Object by Gaze`; (2) `Identify Object by Pointing Action`; and (3) `Identify Joint Object`.

The arrows indicate signals either emitted from or received by a function. For example, `Identify Object by Gaze` receives two signals, an *Eye Gaze* signal and a *Time* signal, and emits single signal *Hypotheses* in return. This indicates an interaction mechanism in which `Identify Object by Gaze` computes a set of hypotheses of intended object selections by integrating eye gaze and timing information. Importantly, the diagram does not elaborate on what exactly constitutes an *Eye Gaze* or *Time* signal, nor does the diagram explain how hypotheses of intended object selections are computed. While the diagram does prescribe a direction towards a solution by indicating the use of eye gaze and pointing actions, the precise function carriers that should be implemented to address these key functions and define the signals are left unspecified at this stage. This makes it easy to explore alternatives or arrive at different descriptions of a function model.

Even though a function structure is at a high level of abstraction, it can already be used to quantitatively explore emergent system outcomes, such as emergent performance due to hypothetical interactivity generated through the interaction of signals and functions in the model. Prior work describes in detail how such studies are carried out in the context of augmentative and alternative communication [9, 20], or could be carried out in the context of virtual and augmented reality [6].

4 Five Benefits of Function Models

4.1 Making Design Problems Tractable

It is easy to enter a state of *analysis paralysis* when designing an accessible mixed reality interface due to the sheer complexity of the design. For example, a requirements gathering process may feel endless as it is easy to overlook requirements which later turn out to be essential. As a result, the requirements are never considered complete and further design never progresses.

A function model helps by explicitly enforcing a boundary around the system it is modeling. By starting with one or several overall functions and gradually decomposing them, the boundary of the system is made explicit by the design team. This enables the design team to assess whether the design task is sufficiently well-defined to be addressable.

4.2 Understanding What Needs to be Done

A common issue in system design is to fail to describe requirements accurately, or fail to include them at all. This result in requirements being introduced at late stage in the design process, which necessitates expensive redesigns at a late stage of a design, when a prototype, or even a deployable system, already exists.

A function model helps reduce the risk of omitting or misconstruing critical requirements, as such requirements are more evident when all essential functions of a system have been described, including their relationships with other functions. It is easy for a design team to gain a holistic view of a system design by inspecting a function model diagram and readily observing how functions are interconnected to realize their purposes. This minimizes the risk of accidentally omitting functionality in a requirement specification.

4.3 Knowing What to Look for

User research is essential in any mixed reality interface design, however, user research can be challenging in practice. It is frequently difficult to recruit participants, and once recruited, engagement is typically limited due to practical constraints, fatigue, and budgets. Hence, once engaged in user research with end-users it is essential to know what to look for to maximize the value of such engagement.

Activities with people with disabilities, such as co-design, requirements elicitation, feature prioritization, observations, or contextual inquiry all require the user researcher to have a clear idea about the system as a whole in order to relate and make relevant the findings to the design team. Further, since user research is both time and resource constrained, function models can be used by user researchers to have a better idea about alternative designs, barriers, ideas for features, and so on, which can help structure user research sessions to maximize value.

4.4 Systematically Exploring Possibilities

A function model acts as a holistic map of a system as it describes what a system has to carry out to fulfill its purpose. Since a function can map to different function carriers, this system map gives the design team the ability to systematically explain a wide range of different combinations of function carriers to realize a system. Further, such a design can be done together with end-users, both for the system as a whole, or by explicitly considering specific subsystems or interaction mechanisms. This way of viewing design at a functional level prevents design fixation in the design team and opens up different ways to systematically explore different possible implementations.

In addition, it is possible to parameterize function models and signals and thereby study emergent outcomes of the system by varying parameter values in simulations. Controllable parameters, such as timeout settings or locations and sizes of interface elements, are governable by the designer and an understanding of some of the effects of such parameter values allows the design team to assess how to fine-tune or optimize aspects of the system. Uncontrollable parameters, such as users' individual strategies or performance, cannot be directly controlled by the design team but an understanding of such parameters allows the design team to explore how emergent outcomes of the system, such as accuracy in an object selection task, changes depending on assumptions of individual users, accuracy in any machine learning component, or noise inherent in input signals.

4.5 Linking Technology Advances to User Needs

Another way function models can help design accessible mixed reality interfaces is by communicating to the design team and end-users what *could* emerge if the system can be designed in a particular way. For example, prior work on designing a context-aware sentence retrieval system for individuals with motor disabilities [7] found through simulations that such a system could bring significant benefits, even if context sensing is assumed to be highly noisy and word prediction components are assumed to frequently fail. In other words, designing a function model allows predicting outcomes and thereby identifying potential. This can inspire further exploration and identification of technological solutions or advances that can act as function carriers for a particularly promising system design.

5 Conclusion

This paper has explained five ways in which function models can help design accessible mixed reality interfaces: (1) making design problems tractable; (2) understanding what needs to be done; (3) knowing what to look for; (4) systematically exploring possibilities; and (5) linking technology advances to user needs. Function models are rich descriptions of system designs at the functional level—they describe what must be achieved for a system to fulfill its purpose. However, it is important to realize that they are only part of a methodological toolbox for accessible mixed reality interface design, helping, but not replacing, any need for user research, co-design, and deployment studies.

Acknowledgments

This work was supported by EPSRC (grant EP/W02456X/1).

REFERENCES

[1] S. Card, T. Moran, and A. Newell. The Psychology of Human-Computer Interaction. 1983.

[2] J. Dudley, L. Yin, V. Garaj, and P. O. Kristensson. Inclusive immersion: a review of efforts to improve accessibility in virtual reality, augmented reality and the metaverse. *Virtual Reality*, 27(4):2989–3020, 2023.

[3] A. Fowler, K. Partridge, C. Chelba, X. Bi, T. Ouyang, and S. Zhai. Effects of language modeling and its personalization on touchscreen typing performance. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pp. 649–658, 2015.

[4] A. Hamid and P. O. Kristensson. 40 years of eye typing: Challenges, gaps, and emergent strategies. *Proceedings of the ACM on Human-Computer Interaction*, 8(ETRA):1–19, 2024.

[5] H. H. Koester and S. Levine. Model simulations of user performance with word prediction. *Augmentative and Alternative Communication*, 14(1):25–36, 1998.

[6] P. O. Kristensson. Designing virtual and augmented reality user interfaces using parameterized function structure models. In *2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 419–423. IEEE, 2024.

[7] P. O. Kristensson, J. Lilley, R. Black, and A. Waller. A design engineering approach for quantitatively exploring context-aware sentence retrieval for nonspeaking individuals with motor disabilities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–11, 2020.

[8] P. O. Kristensson, M. Mjelde, and K. Vertanen. Understanding adoption barriers to dwell-free eye-typing: Design implications from a qualitative deployment study and computational simulations. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, pp. 607–620, 2023.

[9] P. O. Kristensson and T. Müllners. Design and analysis of intelligent text entry systems with function structure models and envelope analysis. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2021.

[10] A. Oulasvirta, P. O. Kristensson, X. Bi, and A. Howes. *Computational interaction*. Oxford University Press, 2018.

[11] G. Pahl and W. Beitz. *Engineering Design: A Systematic Approach*. Springer Science & Business Media, 2013.

[12] K. Palin, A. M. Feit, S. Kim, P. O. Kristensson, and A. Oulasvirta. How do people type on mobile devices? observations from a study with 37,000 volunteers. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services*, pp. 1–12, 2019.

[13] T. C. Peck, K. A. McMullen, and J. Quarles. Divrsify: Break the cycle and develop vr for everyone. *IEEE Computer Graphics and Applications*, 41(6):133–142, 2021.

[14] S. K. Tadeja, P. Seshadri, and P. O. Kristensson. Aerovr: An immersive visualisation system for aerospace design and digital twinning in virtual reality. *The Aeronautical Journal*, 124(1280):1615–1635, 2020.

[15] K. Vertanen, C. Fletcher, D. Gaines, J. Gould, and P. O. Kristensson. The impact of word, multiple word, and sentence input on virtual keyboard decoding performance. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2018.

[16] K. Vertanen, D. Gaines, C. Fletcher, A. M. Stanage, R. Watling, and P. O. Kristensson. Velociwatch: Designing and evaluating a virtual keyboard for the input of challenging text. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2019.

[17] K. Vertanen and P. O. Kristensson. Parakeet: A continuous speech recognition system for mobile touch-screen devices. In *Proceedings of the 14th international conference on Intelligent user interfaces*, pp. 237–246, 2009.

[18] K. Vertanen, H. Memmi, J. Emge, S. Reyal, and P. O. Kristensson. Velocitap: Investigating fast mobile text entry using sentence-based decoding of touchscreen keyboard input. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 659–668, 2015.

[19] J. H. Williamson, A. Oulasvirta, P. O. Kristensson, and N. Banovic. *Bayesian Methods for Interaction and Design*. Cambridge University Press, 2022.

[20] B. Yang and P. O. Kristensson. Imperfect surrogate users: Understanding performance implications of augmentative and alternative communication systems through bounded rationality, human error, and interruption modeling. *Proceedings of the ACM on Human-Computer Interaction*, 7(MHCI):1–33, 2023.

[21] L. Yin, J. Dudley, V. Garaj, and P. Kristensson. Inclusivity requirements for immersive content consumption in virtual and augmented reality. In *Cambridge Workshop on Universal Access and Assistive Technology*, pp. 13–21. Springer, 2023.

[22] L. Yin, J. J. Dudley, V. Garaj, and P. O. Kristensson. An online survey assessing the accessibility barriers encountered by users of virtual and augmented reality. In *2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 330–334. IEEE, 2024.