zeroG: Towards an Integrated Development Environment for Deploying Radar-based Gesture User Interfaces

Arthur Sluÿters[0000−0003−0804−0106] and
Mehdi Ousmer[0000−0002−0222−0029]

Abstract Despite advancing at a tremendous pace recently, few real-world applications have stemmed from research on radar-based gesture interaction. This phenomenon can be attributed to the complexity of integrating signal processing techniques and gesture recognition algorithms into user-friendly applications, especially for developers lacking expertise in this field. In response, this paper introduces the main ideas behind ***zeroG***, an upcoming software framework designed to streamline the development of radar-based gesture interfaces. Once completed, its graphical user interface should enable developers to assemble standardized modules into complex gesture recognition dataflows, facilitating both testing and application development. By introducing a clear separation of concerns between application frontend and gesture recognition, ***zeroG*** will enable developers to effortlessly adapt existing dataflows to new applications or sensors, without requiring extensive experience in (radar-based) gesture recognition. This paper explores the current landscape of tools for creating radar-based gesture interfaces, introduces the core elements of the ***zeroG*** framework emerging from its early stages of design and development, and outlines future development stages and potential challenges.

1.1 Introduction

Research on radar-based gesture interaction has been advancing at a tremendous pace in recent years [1, 16, 42]. However, despite the involvement of large companies like Google with their Soli platform [25], few real-world applications have emerged from this research. One reason for this lack of enthusiasm is the complexity involved in leveraging signal processing and gesture recognition techniques to create user-

Arthur Sluÿters, Mehdi Ousmer
Université catholique de Louvain, Louvain Research Institute in Management and Organizations, Place des Doyens, 1, B-1348 Louvain-la-Neuve, Belgium e-mail: arthur.sluyters@uclouvain.be,mehdi.ousmer@uclouvain.be

friendly gesture interfaces. This complexity can be particularly discouraging for developers who lack expertise in radar-based gesture interaction

While there are development environments for creating user interfaces (UIs) based on just about every interaction modality, such as voice interaction [26] and touch interaction [14], the gesture modality still lacks comprehensive coverage [14]. Although we have capitalized on certain aspects of gesture interfaces since their inception for some devices [18, 21], such as touch screens [9] for 2D multi-stroke gestures [54, 27] or cameras for 3D mid-air gestures [22, 30], many devices [3] and contexts of use [14] remain inadequately covered. The development cycle of a gesture interface is often only partially addressed [50], usually focusing on a few stages like data acquisition and pre-processing [13], but rarely encompassing the complete cycle up to its integration into an interactive application [17].

This paper thus introduces ***zeroG***, a software framework in its early stages of design and development that aims to address these issues by streamlining the development of (radar-based) gesture interfaces. It makes multiple contributions in the domain of Human-Computer Interaction, including:

- A clear separation of concerns between the application frontend and gesture recognition.
- A standardized format for gestures and modules, which enables developers and practitioners to implement, share, and assemble modules into complex gesture recognition dataflows.
- A graphical UI that adapts to developers' experience levels.
- A seamless transition between development stages (design, implementation, testing, deployment) thanks to its tightly integrated set of tools.

Together, these contributions will help developers effortlessly create and maintain highly usable gesture-based interfaces without requiring extensive knowledge of (radar-based) gesture recognition. The rest of this paper is structured as follows:

- Section 1.2 explores related works, including techniques for radar-based gesture interaction and tools for assisting developers in the creation of gesture-based applications. Based on our observations, it introduces the main design rationale for ***zeroG***.
- Section 1.3 provides an overview of ***zeroG***'s key components, namely its dataflow testing and application development tools, and gesture recognition service.
- Section 1.4 then details elements of its software architecture, including gesture sets, modules, and dataflows.
- Section 1.5 discusses the main challenges and future directions for the development of ***zeroG***.
- Section 1.6 concludes the paper.

Through this approach, we aim to demonstrate how, once completed, ***zeroG*** could simplify the development process of (radar-based) gesture interfaces and contribute to their popularization.

1.2 Related Work

In this section, we explore existing works on radar-based gesture recognition (Section 1.2.1) and tools designed to facilitate the creation of gesture-based interfaces (Section 1.2.2). We then introduce the main rationale for the ***zeroG*** framework (Section 1.2.3).

1.2.1 Techniques for Radar-based Interaction

Recent years have seen significant advancements in research on radar-based gesture recognition [1, 16], with a wide variety of radar sensors being featured in the literature [42].

Among these, Google's Soli [25] is one of the most mature systems. Its extremely compact size enables integration into small portable devices like smartphones and smartwatches. Soli was initially evaluated with a limited set of four micro-gestures, but subsequent research has explored other gestures, contexts, and applications [51]. For instance, Hajika *et al.* [19] employed a wrist-worn Soli radar to identify on-skin hand gestures, such as swiping on the back of the hand. Hayashi *et al.* [20] used a Soli radar embedded in a smartphone to recognize swipe interactions. Their model was capable of identifying gestures from unsegmented data streams while being small and efficient enough to run on low-power devices. Pucihar *et al.* [34] explored Soli's ability to detect gestures through 75 materials of varying thicknesses. Similarly, Leiva *et al.* [24] investigated the feasibility of integrating Soli into clothing and furniture by capturing gestures through three different fabrics.

While Soli's high-frequency and low-power operation makes it ideal for close-range, high-precision applications, it is not suited for all use cases. For example, Palapina *et al.*'s Pantomime [31] focused on full-body gesture recognition, which required a radar with a greater range than Soli, at the cost of a lower resolution. They developed a technique based on point clouds to reduce the size of raw radar data before feeding it into feature extraction and classification algorithms. Sluÿters *et al.*'s RadarSense [42] tackled the challenge of interoperability across radar sensors. By applying normalization to radar signals, they aim to facilitate the reuse of datasets across different environments and radar systems, as long as these systems operate within the same frequency range.

Despite the wide range of radars, contexts, and gesture sets explored in these studies, the proposed techniques often lack flexibility or require substantial expertise to adapt them to new contexts, such as different sensors, gesture sets, or environments.

1.2.2 Tools for Creating Gesture-based Applications

Numerous tools have been proposed to facilitate different stages in the creation of gesture-based applications.

For instance, Ashbrook *et al.*'s MAGIC [4] is designed to aid in the design and testing of gestures, particularly to prevent unintended activations. MAGIC's implementation is adaptable to a variety of sensors and provides video feedback for system designers.

Jackknife [46] is a modality-agnostic gesture recognizer that comes with a collection of techniques, allowing it to handle a wide variety of contexts (sensors, environments). While it can be adapted to the needs of specific applications relatively easily, it may need to be combined with other techniques (*e.g.,* gesture segmentation) in some contexts (*e.g.,* real-time gesture interaction from a continuous stream of data), which would require more expertise in gesture recognition to perform, and may not be suited to all contexts (*e.g.,* another recognizer might be needed to recognize gestures accurately).

Leiva *et al.*'s Gesture à Go Go [23] augment stroke gesture datasets with synthetic examples derived from as few as one recording of each gesture. This tool significantly reduces the time and effort needed to collect enough examples of gestures for training algorithms for gesture recognition, such as Jackknife.

In [12], Caramiaux *et al.* introduce a technique for gesture recognition that characterizes gesture execution, including scaling, rotation, and speed. This technique can be leveraged by developers to construct highly adaptable gesture interfaces. For instance, the authors demonstrate a sound playback application where the way a gesture is performed changes how the corresponding sound is played, such as playback speed. The system also supports early recognition, enabling applications to react while a gesture is still being performed.

XDKinect [29] is an adaptable, extensible framework that facilitates the creation of cross-device applications using Kinect. With its client-server and event-driven architecture, it offers various APIs. The framework supports multi-device interactions and doesn't require a direct Kinect connection to the client computer. A user study found XDKinect easy to use and effective, particularly for users without prior Kinect experience.

The SoD-Toolkit [38], created for multi-device interactions and ubiquitous environments, features a "plug and play" architecture for easy sensor integration. It includes client libraries for major device and UI platforms, enabling designers to prototype without physical elements. This toolkit works with Leap Motion, Kinect v1 and v2, Apple iBeacon, and mobile device sensors. But, it doesn't allow for adding new devices and only works with certain app development environments. It also doesn't specifically support the development of new gesture-based interactions.

QuantumLeap [41] is a framework for the development of gesture interfaces. Its modular pipeline architecture can be configured to suit the needs of any gesture-controlled application using its GUI. The QuantumLeap Javascript API allows developers to add support for gestures to their applications.

While these tools are a great step towards facilitating the development of highly usable gesture-based applications, they still exhibit several major drawbacks. They typically do not cover all of the main stages of development. Combined with a lack of interoperability between tools, this complicates their integration into complete development or research workflows. They offer limited modularity and flexibility, making it challenging to adapt them to new sensors and contexts. For instance, XDKinect focuses solely on Kinect devices and Jackknife does not support deep learning-based algorithms for gesture recognition. The tools are usually quite low-level, requiring substantial expertise in gesture recognition to apply them effectively in real-world applications (*e.g.,* Jackknife, QuantumLeap). These limitations highlight the need for a comprehensive framework that can streamline the entire development process of gesture-based applications, from design to deployment.

1.2.3 Design Rationale of *zeroG*

With ***zeroG***, we aim to build on the effort of researchers and practitioners in gesture recognition by creating an integrated development environment for (radar-based) gesture interfaces. This environment will enable developers to move beyond low-level implementation challenges and focus on the high-level design of gesture-based interfaces.

The ***zeroG*** framework should be modular and flexible, accommodating a wide range of applications and contexts of use. It should provide a clear separation of concerns between application UIs and gesture recognition, facilitating their design, maintenance, and future evolution. In addition, ***zeroG*** should be accessible to developers of all experience levels: (1) experts could manually create gesture recognition logic from scratch by freely combining modules into dataflows, (2) advanced users could adapt predefined dataflow templates for their applications by filling placeholders with suitable module implementations, like in QuantumLeap [41], and (3) novice users could ask ***zeroG*** to suggest or even generate appropriate dataflows based on criteria such as the type of application, sensor(s), and gestures. Most importantly, ***zeroG*** should facilitate a smooth transition between the various development stages of gesture-based applications, from testing signal processing and gesture recognition techniques to developing gesture-based applications, and, ultimately, to their use by end users and their maintenance.

1.3 The *zeroG* Framework, a Multi-tool for Gesture-based Interaction

We envision the ***zeroG*** framework as a combination of three main components (Fig 1.1): (1) a testing tool for signal processing and gesture recognition techniques (Section 1.3.1), (2) a development tool for gesture-based applications (Section 1.3.2),

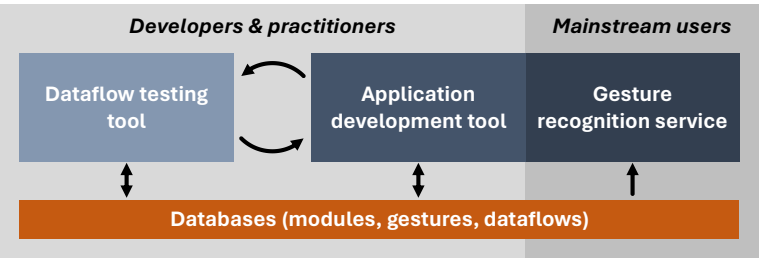


Fig. 1.1: Interaction between the main components of ***zeroG***.

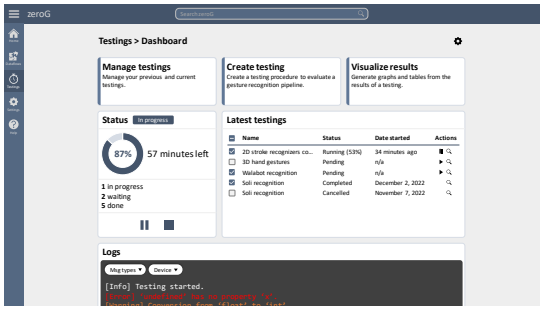
and (3) a real-time gesture recognition system (Section 1.3.3). It will target two main audiences, namely developers and practitioners, with its dataflow testing and application development tools, and mainstream users, as a gesture recognition service.

1.3.1 Dataflow Testing Tool

The ***zeroG*** dataflow testing tool is designed to assist researchers and practitioners in the development and evaluation of novel techniques for gesture interaction, spanning from signal processing techniques to algorithms for gesture segmentation and recognition. It enables users to construct and evaluate dataflows of varying complexity (Section 1.4.2 and Fig. 1.3), ranging from simple setups featuring a single gesture recognition algorithm to integrated solutions combining filters, segmentation techniques, and gesture recognition algorithms.

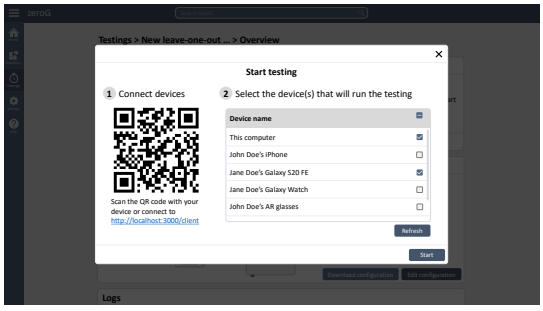
By leveraging a standardized data structure for gesture sets, a modular dataflow architecture, and shareable configuration files, the tool streamlines the integration and reuse of dataflow, modules, and gestures across testings scenarios (Section 1.4). This streamlined approach is especially useful for comparative testings of dataflows, like evaluating a new algorithm against state-of-the-art techniques in various conditions or selecting the dataflow configuration best suited to a specific context of use. In addition, the tool makes it trivial to replicate and reproduce testings from other research teams, as all the necessary data can be shared in a simple, standardized package. This could foster collaboration across researchers and ensure the integrity and reproducibility of experiments.

The tool's UI features a dynamic dashboard (Fig. 1.2a), providing users with information and control over completed, ongoing, and scheduled testings. This dashboard also serves as a hub for accessing other testing-related features, including creating new testings and visualizing the results of completed testings. Selecting a specific testing shows its dataflow configuration, description, status, and logs (Fig. 1.2b). Testings can be run on external devices, such as a smartphone or a smartwatch, to determine how a dataflow performs on devices with different capabilities (Fig 1.2c). Once a testing is completed, users can generate visual representations, including confusion matrices, to help visualize its results (Fig. 1.2d).

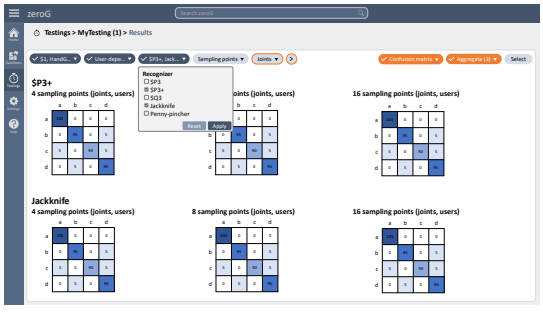


(a) Testings dashboard.

(b) Testing overview.



(c) Run a testing on various devices.



(d) Visualize the results of a testing.

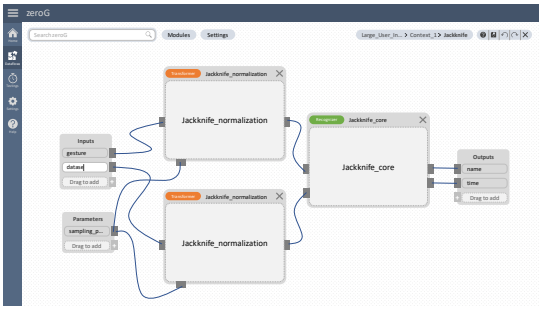
Fig. 1.2: Mockups of the ***zeroG*** UI (testing).

1.3.2 Application Development Tool

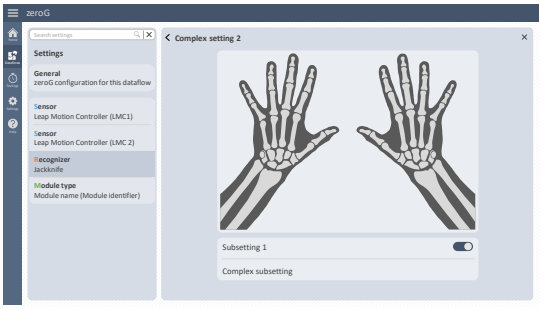
The application development tool extends the capabilities of the dataflow testing tool, enabling the creation of gesture-based interfaces and ensuring a seamless transition from the testing of dataflows and modules to their deployment in real applications.

The ***zeroG*** framework supports two-way communications with applications, allowing them to provide multiple alternative dataflow configurations on their first start, increasing the likelihood that one matches end-users' systems (in particular, the available sensors). Additionally, applications can switch contexts (*e.g.,* from UI navigation to drawings recognition), dynamically update dataflow elements (*e.g.,* gestures and settings), and send user inputs (*e.g.,* 2D touch-based gestures and button clicks) as input signals to the dataflow. Dataflows and modules are further described in Section 1.4.2.

An API is provided to simplify the integration of gesture recognition into applications, managing the assignment of actions in the application to user gestures and all other communications with ***zeroG***. In addition, ***zeroG*** can generate scaffolding code based on the API to perform, among other, initialization steps, such as connecting to the framework and submitting dataflow configurations. By reducing the need for front-end developers to delve into the intricacies of gesture recognition, the elements aim to facilitate the development of highly usable gesture-based applications.



(a) Dataflow creation tool.



(b) Dataflow settings.

Fig. 1.3: Mockups of the ***zeroG*** UI (dataflow).

1.3.3 Gesture Recognition Service

The gesture recognition service is the only component of ***zeroG*** visible to end-users (Fig. 1.4). As implied by its name, it operates as a background service on users' devices and performs gesture recognition for client applications. To improve the user experience, the service minimizes end-user involvement by automatically initializing gesture recognition dataflows when applications connect and terminating them when they become inactive. It also handles other tasks in the background, including managing all communication with gesture-based applications and downloading any missing modules and gestures required for an application's dataflow. End-users retain visibility and control over applications reliant on the ***zeroG*** gesture recognition service, including the ability to remove applications that they no longer trust.

1.4 Software Architecture

In this section, we introduce the main pieces of the ***zeroG*** software architecture. Section 1.4.1 introduces our data structure for gesture sets and Section 1.4.2 discusses the structure of modules and dataflows.

1.4.1 Gesture Sets

Gesture recordings play an important role in the development of gesture-based interfaces, serving for training and evaluating gesture recognition algorithms [11]. With this in mind, our objective was to design a standardized and versatile data

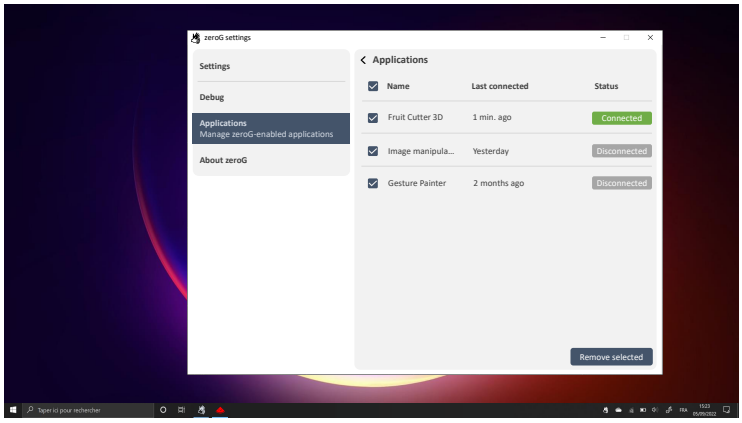


Fig. 1.4: Mockup of the ***zeroG*** gesture recognition service.

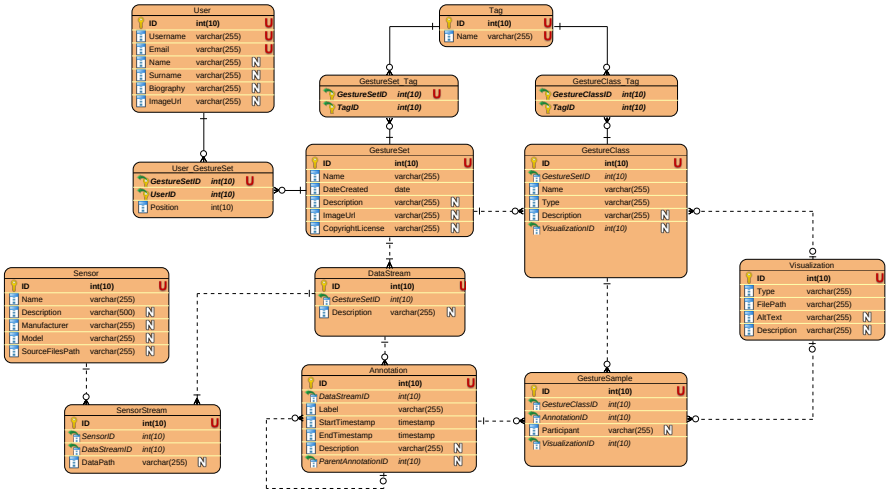


Fig. 1.5: Entity-relationship diagram of ***zeroG*** datasets.

structure for gesture sets within ***zeroG*** that accommodates signals from different types of sensors. This approach offers numerous benefits, such as facilitating gesture reuse across applications and testing environments, and enabling the fusion of data from multiple sensors (*e.g.,* radars signals, skeleton data, or IMU data). An entity-relationship diagram illustrating our data structure for gesture sets is provided in Fig. 1.5.

The `GestureSet` entity features one or more `DataStreams`, at least one `GestureClass` (*e.g.,* swipe left or thumbs up), and the list of its authors as `User` entities. `DataStreams` encapsulate continuous streams of data and comprise one or more `SensorStreams` and `Annotations` (Fig. 1.6). `SensorStreams` are uninterrupted streams of data captured by a single `Sensor`. They store the path to a JSON file featuring the sensor data to avoid storing this information directly in the database. `Annotations` serve to annotate a segment within a `DataStream` and feature a label, description, and timestamps. They are hierarchical, enabling the creation of sub-annotations, *e.g.,* if one annotation defines a gesture, sub-annotations can be used to highlight the start, middle, and ending of the gesture. A `GestureClass` defines a category of gestures. It may feature a `Visualization` (*e.g.,* an image or video) depicting the gesture. Each `GestureClass` features at least one `GestureSample`. A `GestureSample` represents one recording of a specific gesture class. In our data structure, each gesture sample corresponds to an `Annotation`, thus allowing us to accommodate both pre-segmented and unsegmented gestures. Each gesture sample may also include a `Visualization`. Both the `GestureSets` and `GestureClasses` entities can be assigned `Tags`, *i.e.,* descriptive labels that can help, *e.g.,* filter and classify them.

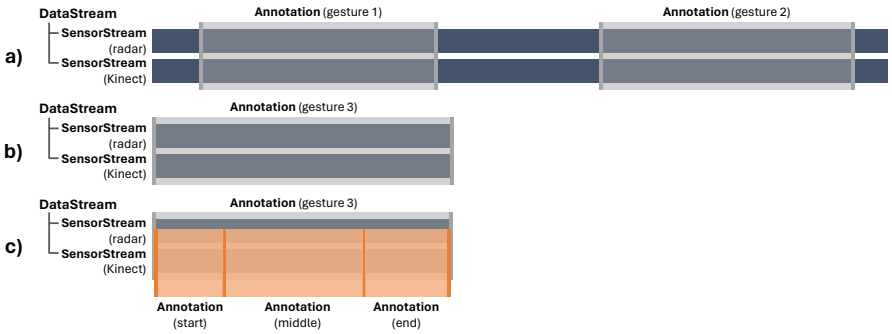


Fig. 1.6: Illustration of `DataStreams`, `SensorStreams`, and `Annotations`: a) a `DataStream` containing two gestures delimited using annotations, b) one gesture spanning the entire `DataStream`, c) a gesture with sub-annotations delimiting its start, middle, and end.

1.4.2 Modules and Dataflows

`Dataflows` serve as the foundation for constructing gesture recognition logic. They can be evaluated in the testing tool (Section 1.3.1) or used for real-time gesture interaction in the application development tool (Section 1.3.2). They comprise interconnected `Modules` of three types, namely `Sources`, `Transforms`, and `Sinks`.

`Source` modules generate signals that can be consumed by other modules within the dataflow. They can range from sensors like radars, Kinect devices, or Leap Motion Controllers, to user inputs such as buttons in the application UI, to system events. `Transform` modules process input signals and send the result to subsequent modules. For example, a gesture segmenter might take individual frames as input and produce sequences of frames corresponding to potential gestures. Meanwhile, a gesture recognizer might receive these sequences of frames and output the type of gesture detected. `Sink` modules receive and consume incoming signals. They may, for instance, send the signals to a connected application or save them in a log file. Finally, `Compound` modules are combinations of other modules, such as a segmenter and a gesture recognizer, into a single block for easy reuse across various dataflows.

Applications can feature different `Contexts` that they switch between during interaction, with each context potentially requiring a distinct dataflow composed of different modules tailored to the type of gestures to be recognized. For instance, one dataflow might be designed for basic UI navigation gestures, while another might focus on mid-air drawing recognition. This contextual separation enhances gesture recognition accuracy, by tailoring dataflows to specific sets of gestures, and ensures a better separation of concerns.

Finally, ***zeroG*** supports `Templates` to facilitate the creation of dataflows for testings and applications. Similar to the approach used in Sluÿters *et al.*'s QuantumLeap [41], ***zeroG***'s templates are essentially incomplete dataflows that include `Placeholder` modules. These placeholders can be replaced with specific modules

suited to a specific context. Templates promote efficient software reuse and can be employed by users of all experience levels to easily construct gesture recognition dataflows. In the future, ***zeroG*** could automatically suggest dataflows tailored to an application's use case by leveraging the templates, modules, and dataflows stored in its database. This capability would enable even novice users to create complex gesture-based interfaces from scratch with minimal effort.

1.5 Challenges and Future Directions

As ***zeroG*** is still in its early stages of design and development, several challenges remain to be addressed before a first version is ready to be deployed. A critical area concerns the communication between modules, which requires establishing a standardized data format compatible with various types of sensor data, including skeletons, radar signals, and IMU data. Another challenge lies in supporting multimodal sensing, such as combining radar- and vision-based inputs, which will require synchronization mechanisms between the source modules. In addition, implementing a suggestion system for dataflow configurations and modules, as well as automatically downloading missing components required by gesture-based applications, will both be essential for making ***zeroG*** accessible to all users. This will involve creating and maintaining a comprehensive database of dataflows, modules, and gestures. We could also look into supporting modules written in different programming languages to enhance the framework's flexibility. Finally, providing accurate visualizations of recorded gestures, especially radar-based ones, presents a significant challenge. For instance, after end-users record new gestures, they, or the system, may need a way to visualize it later, *e.g.,* to display gesture hints in the application.

1.6 Conclusion

In this paper, we introduced ***zeroG***, a novel integrated development environment designed to facilitate the development and deployment of (radar-based) gesture interfaces. We began by reviewing existing techniques for radar-based gesture interaction and tools for developing gesture-based applications. Our findings underscored a need to streamline the development process of gesture interfaces, leading us to introduce the design rationale for ***zeroG***. We then described the three main components of ***zeroG***, each designed to address some of these shortcomings: (1) the dataflow testing tool, which assists users in the development and evaluation of gesture interaction techniques, (2) the application development tool, which helps create gesture-based interfaces with minimal effort, and (3) the gesture recognition service, which runs in the background on end-users device and manages gesture recognition for client applications. Finally, we introduced the core elements of ***zeroG***'s software architecture, including gesture sets, modules, and dataflows.

Overall, by seamlessly integrating into research and development workflows ***zeroG*** has the potential to foster the emergence of new techniques for gesture recognition, as well as facilitate their efficient sharing across the community and their use in real-world applications.

Acknowledgements The authors of this paper are very grateful to the anonymous reviewers whose suggestions helped improve and clarify this manuscript. Arthur Sluÿters is funded by the "Fonds de la Recherche Scientifique - FNRS" under Grants n°40001931 and n°40011629.

References

1. Ahmed, S., Kallu, K., Ahmed, S. & Cho, S. Hand Gestures Recognition Using Radar Sensors for Human-Computer-Interaction: A Review. *Remote Sensing*. **13** (2021), https://www.mdpi.com/2072-4292/13/3/527
2. Almendros-Jiménez, J., Iribarne, L., Asensio, J., Padilla, N. & Vicente-Chicote, C. An Eclipse GMF Tool for Modelling User Interaction. *Visioning And Engineering The Knowledge Society. A Web Science Perspective*. pp. 405-416 (2009), https://doi.org/10.1007/978-3-642-04754-1_42
3. Aquino, N., Vanderdonckt, J., Condori-Fernandez, N., Dieste Tubío & Pastor, O. Usability evaluation of Multi-device/platform User Interfaces Generated by Model-Driven Engineering. *Proceedings of the ACM International Symposium on Empirical Software Engineering and Measurement, ESEM 2010, Bolzano/Bozen, Italy, 16-17 September 2010*. pp. 1-10 (2010), https://doi.org/10.1145/1852786.1852826
4. Ashbrook, D. & Starner, T. MAGIC: a motion gesture design tool. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2010, Atlanta, Georgia, USA, 10-15 April 2010*. pp. 2159–2168 (2010), https://doi.org/10.1145/1753326.1753653
5. Attygalle, N., Leiva, L., Kljun, M., Sandor, C., Plopski, A., Kato, H. & Pucihar, K. No Interface, No Problem: Gesture Recognition on Physical Objects Using Radar Sensing. *Sensors*. **21**, 5771 (2021), https://doi.org/10.3390/s21175771
6. Attygalle, N., Vuletic, U., Kljun, M. & Pucihar, K. Towards Hand Gesture Recognition Prototype Using the iwr6843isk Radar Sensor and Leap Motion. *Proceedings Of The 8th Human-Computer Interaction Slovenia (HCI SI) Conference 2023, Maribor, Slovenia, January 26, 2024*. **3657** pp. 78-88 (2023), https://ceur-ws.org/Vol-3657/paper9.pdf
7. Avrahami, D., Patel, M., Yamaura, Y., Kratz, S. & Cooper, M. Unobtrusive Activity Recognition and Position Estimation for Work Surfaces Using RF-Radar Sensing. *ACM Trans. Interact. Intell. Syst.*. **10** (2019,8), https://doi.org/10.1145/3241383
8. Berenguer, A., Oveneke, M., Khalid, H., Alioscha-Pérez, M., Bourdoux, A. & Sahli, H. GestureVLAD: Combining Unsupervised Features Representation and Spatio-Temporal Aggregation for Doppler-Radar Gesture Recognition. *IEEE Access*. **7** pp. 137122-137135 (2019), https://doi.org/10.1109/ACCESS.2019.2942305
9. Beuvens, F. & Vanderdonckt, J. Designing graphical user interfaces integrating gestures. *Proceedings of the 30th ACM International Conference on Design of Communication, SIGDOC '12, Seattle, Washington, USA, 3-5 October 2012, .* pp. 313-322 (2012), https://doi.org/10.1145/2379057.2379116
10. Campos, J., Fayollas, C., Martinie, C., Navarre, D., Palanque, P. & Pinto, M. Systematic automation of scenario-based testing of user interfaces. *Proceedings of the 8th ACM Symposium On Engineering Interactive Computing Systems, EICS 2016, Brussels, Belgium, 21-24 June 2016*. pp. 138-148 (2016), https://doi.org/10.1145/2933242.2948735
11. Caputo, A., Giachetti, A., Soso, S., Pintani, D., D'Eusanio, A., Pini, S., Borghi, G., Simoni, A., Vezzani, R., Cucchiara, R., Ranieri, A., Giannini, F., Lupinetti, K., Monti, M., Maghoumi,

M., Laviolajr, J., Le, M., Nguyen, H. & Tran, M. SHREC 2021: Skeleton-based hand gesture recognition in the wild. *Computers & Graphics*. **99** pp. 201-211 (2021), https://doi.org/10.1016/j.cag.2021.07.007

12. Caramiaux, B., Montecchio, N., Tanaka, A. & Bevilacqua, F. Adaptive Gesture Recognition with Variation Estimation for Interactive Systems. *ACM Transactions on Interactive Intelligent Systems*. **4**, pp. 1-34 (2014), https://doi.org/10.1145/2643204
13. Chioccarello, S., Sluÿters, A., Testolin, A., Vanderdonckt, J. & Lambot, S. FORTE: Few Samples for Recognizing Hand Gestures with a Smartphone-attached Radar. *Proc. ACM Hum. Comput. Interact.*. **7**, 1-25 (2023), https://doi.org/10.1145/3593231
14. Delimarschi, D., Swartzendruber, G. & Kagdi, H. Enabling integrated development environments with natural user interface interactions. *Proceedings of The 22nd International Conference on Program Comprehension, ICPC 2014, Hyderabad, India, 2-3 June 2014*. pp. 126-129 (2014), https://doi.org/10.1145/2597008.2597791
15. Dessart, C., Motti, V. & Vanderdonckt, J. Showing user interface adaptivity by animated transitions. *Proceedings of the 3rd ACM Symposium on Engineering Interactive Computing System, EICS 2011, Pisa, Italy, 13-16 June 2011*. pp. 95-104 (2011), https://doi.org/10.1145/1996461.1996501
16. Dong, Y. & Qu, W. Review of Research on Gesture Recognition Based on Radar Technology. *Artificial Intelligence For Communications And Networks*. pp. 390-403 (2021), https://doi.org/10.1007/978-3-030-69066-3_34
17. Genaro Motti, V., Raggett, D., Van Cauwelaert, S. & Vanderdonckt, J. Simplifying the development of cross-platform web user interfaces by collaborative model-based design. *Proceedings of the 31st ACM International Conference on Design Of Communication, SIGDOC 2013, Greenville, North Carolina, USA, 30 September 2013-1 October 2013*. pp. 55-64 (2013), https://doi.org/10.1145/2507065.2507067
18. Giacalone, A. XY-WINS: an integraded environment for developing graphical user interfaces. *Proceedings of the 1st Annual ACM SIGGRAPH Symposium on User Interface Software, UIST 1988, Alberta, Canada, 17-19 October 1988*. pp. 129-143 (1988), https://doi.org/10.1145/62402.62425
19. Hajika, R., Gunasekaran, T., Haigh, C., Pai, Y., Hayashi, E., Lien, J., Lottridge, D. & Billinghurst, M. RadarHand: A Wrist-Worn Radar for On-Skin Touch-Based Proprioceptive Gestures. *ACM Trans. Comput.-Hum. Interact.*. **31** (2024,1), https://doi.org/10.1145/3617365
20. Hayashi, E., Lien, J., Gillian, N., Giusti, L., Weber, D., Yamanaka, J., Bedal, L. & Poupyrev, I. RadarNet: Efficient Gesture Recognition Technique Utilizing a Miniature Radar Sensor. *Proceedings Of The ACM Conference On Human Factors In Computing Systems*. (2021), https://doi.org/10.1145/3411764.3445367
21. Henry, T., Hudson, S. & Newell, G. Integrating gesture and snapping into a user interface toolkit. *Proceedings of the 3rd Annual ACM SIGGRAPH Symposium on User Interface Software and Technology, UIST 1990, Utah, USA, 3-5 October 1990*. pp. 112-122 (1990), https://doi.org/10.1145/97924.97938
22. Kim, K., Kim, J., Choi, J., Kim, J. & Lee, S. Depth Camera-Based 3D Hand Gesture Controls with Immersive Tactile Feedback for Natural Mid-Air Gesture Interactions. *Sensors*. **15**, 1022-1046 (2015), https://www.mdpi.com/1424-8220/15/1/1022
23. Leiva, L., Martín-Albo, D. & Plamondon, R. Gestures à Go Go: Authoring Synthetic Human-Like Stroke Gestures Using the Kinematic Theory of Rapid Movements. *ACM Transactions on Intelligent Systems and Technology*. **7**. pp. 1–29 (2015), https://doi.org/10.1145/2799648
24. Leiva, L., Kljun, M., Sandor, C. & Copic Pucihar, K. The Wearable Radar: Sensing Gestures Through Fabrics. *Proceedings Of The 22nd International Conference On Human-Computer Interaction With Mobile Devices And Services*. (2021), https://doi.org/10.1145/3406324.3410720
25. Lien, J., Gillian, L., M. Emre, K., Amihoo,d P., Schwesig, C., Olson, E., Raja, H. & Poupyrev, I. Soli: ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics*. **35**, 142:1-142:19 (2016), https://doi.org/10.1145/2897824.2925953

26. Loo, J., Gemmeke, J., De Pauw, G., Driesen, J., Van hamme, H. & Daelemans, W. Towards a self-learning assistive vocal interface: vocabulary and grammar learning. *Proceedings Of The 1st Workshop On Speech And Multimodal Interaction In Assistive Environments*. pp. 34-42 (2012)
27. Magrofuoco, N., Roselli, P. & Vanderdonckt, J. Two-dimensional Stroke Gesture Recognition: A Survey. *ACM Comput. Surv.*. **54**, 155:1-155:36 (2022), https://doi.org/10.1145/3465400
28. Moldovan, A., Nicula, V., Pasca, I., Popa, M., Namburu, J., Oros, A. & Brie, P. OpenUIDL, A User Interface Description Language for Runtime Omni-Channel User Interfaces. *Proc. ACM Hum. Comput. Interact.*. **4**, 86:1-86:52 (2020), https://doi.org/10.1145/3397874
29. Nebeling, M., Teunissen, E., Husmann, M. & Norrie, M. XDKinect: development framework for cross-device interaction using kinect. *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS 2014, Rome, Italy, 17-20 June 2014*. pp. 65–74 (2020), https://doi.org/10.1145/2607023.2607024
30. Ousmer, M., Sluÿters, A., Magrofuoco, N., Roselli, P. & Vanderdonckt, J. Recognizing 3D Trajectories as 2D Multi-stroke Gestures. *Proc. ACM Hum.-Comput. Interact.*. **4** (2020,11), https://doi.org/10.1145/3427326
31. Palipana, S., Salami, D., Leiva, L. & Sigg, S. Pantomime: Mid-Air Gesture Recognition with Sparse Millimeter-Wave Radar Point Clouds. *Proceedings Of The ACM On Interactive, Mobile, Wearable And Ubiquitous Technologies*. **5**, 27:1-27:27 (2021,3), https://doi.org/10.1145/3448110
32. Pedersoli, F., Benini, S., Adami, N. et al. XKin: an open source framework for hand pose and gesture recognition using kinect. *The Visual Computer*. **30**. pp. 1107–1122 (2014), https://doi.org/10.1007/s00371-014-0921-x
33. Pucihar, K., Sandor, C., Kljun, M., Huerst, W., Plopski, A., Taketomi, T., Kato, H. & Leiva, L. The Missing Interface: Micro-Gestures on Augmented Objects. *Extended Abstracts Of The ACM CHI Conference On Human Factors In Computing Systems*. pp. 1-6 (2019), https://doi.org/10.1145/3290607.3312986
34. Pucihar, K., Attygalle, N., Kljun, M., Sandor, C. & Leiva, L. Solids on Soli: Millimetre-Wave Radar Sensing through Materials. *Proc. ACM Hum.-Comput. Interact.*. **6** (2022,6), https://doi.org/10.1145/3532212
35. Roland, D., Hainaut, J., Hick, J., Henrard, J. & Englebert, V. Database Engineering Processes with DB-MAIN. *Proceedings of the 8th European Conference on Information Systems, Trends In Information And Communication Systems For The 21st Century, ECIS 2000, Vienna, Austria, 3-5 July 2000*. pp. 244-251 (2000), http://aisel.aisnet.org/ecis2000/68
36. Samuelsson, S. & Book, M. Towards Sketch-based User Interaction with Integrated Software Development Environments. *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW 2020, Seoul, Republic of Korea, 27 June 2020-19 July 2020*. pp. 181-184 (2020), https://doi.org/10.1145/3387940.3392231
37. Sellier, Q., Sluÿters, A., Vanderdonckt, J. & Poncin, I. Evaluating gesture user interfaces: Quantitative measures, qualitative scales, and method. *Int. J. Hum. Comput. Stud.*. **185** pp. 103242 (2024), https://doi.org/10.1016/j.ijhcs.2024.103242
38. Seyed, T., Azazi, A., Chan, E., Wang, Y. & Maurer, F. SoD-Toolkit: A Toolkit for Interactively Prototyping and Developing Multi-Sensor, Multi-Device Environments. *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces, ITS 2015, Madeira, Portugal, 15-18 November 2015*. pp. 171–180 (2015), https://doi.org/10.1145/2817721.2817750
39. Siean, A., Pamparau, C., Sluÿters, A., Vatavu, R. & Vanderdonckt, J. Flexible gesture input with radars: systematic literature review and taxonomy of radar sensing integration in ambient intelligence environments. *J. Ambient Intell. Humaniz. Comput.*. **14**, 7967-7981 (2023), https://doi.org/10.1007/s12652-023-04606-9
40. Sluÿters, A., Lambot, S. & Vanderdonckt, J. Hand Gesture Recognition for an Off-the-Shelf Radar by Electromagnetic Modeling and Inversion. *Proceedings of the 27th International Conference on Intelligent User Interfaces, IUI 2022, Helsinki, Finland, 22-25 March 2022*. pp. 506-522 (2022), https://doi.org/10.1145/3490099.3511107

41. Sluÿters, A., Ousmer, M., Roselli, P. & Vanderdonckt, J. QuantumLeap, a Framework for Engineering Gestural User Interfaces based on the Leap Motion Controller. *Proc. ACM Hum. Comput. Interact.*. **6**, 161:1-161:47 (2022), https://doi.org/10.1145/3532211
42. Sluÿters, A., Lambot, S., Vanderdonckt, J. & Vatavu, R. RadarSense: Accurate Recognition of Mid-air Hand Gestures with Radar Sensing and Few Training Examples. *ACM Trans. Interact. Intell. Syst.*. **13** (2023,9), https://doi.org/10.1145/3589645
43. Sluÿters, A., Sellier, Q., Vanderdonckt, J., Parthiban, V. and & Maes, P. Consistent, Continuous, and Customizable Mid-Air Gesture Interaction for Browsing Multimedia Objects on Large Displays. *International Journal of Human–Computer Interaction*. **39**, 2492-2523 (2023), https://doi.org/10.1080/10447318.2022.2078464
44. Sluÿters, A., Lambot, S., Vanderdonckt, J. & Villarreal-Narvaez, S. Analysis of User-Defined Radar-Based Hand Gestures Sensed Through Multiple Materials. *IEEE Access*. **12** pp. 27895-27917 (2024), https://doi.org/10.1109/ACCESS.2024.3366667
45. Sousa, K., Filho, H., Vanderdonckt, J., Rogier, E. & Vandermeulen, J. User interface derivation from business processes: a model-driven approach for organizational engineering. *Proceedings of ACM Symposium on Applied Computing, SAC 2008, Fortaleza, Ceara, Brazil, 16-20 March 2008*. pp. 553-560 (2008), https://doi.org/10.1145/1363686.1363821
46. Taranta II, E., Samiei, A., Maghoumi, M., Khaloo, P., Pittman, C. & Laviola Jr., J. Jackknife: A Reliable Recognizer with Few Samples and Many Modalities. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI 2017, Denver, Colorado, USA, 6-11 May 2017*. pp. 5850–5861 (2017), https://doi.org/10.1145/3025453.3026002
47. Vanderdonckt, J., Roselli, P. & Pérez-Medina, J. !FTL, an Articulation-Invariant Stroke Gesture Recognizer with Controllable Position, Scale, and Rotation Invariances. *Proceedings of the ACM International Conference on Multimodal Interaction, ICMI 2018, Boulder, CO, USA, 16-20 October 2018*. pp. 125-134 (2018), https://doi.org/10.1145/3242969.3243032
48. Villanueva, E., Torres, I., Osaba, E., Canzoneri, S., Franchini, A. & Blasi, L. PIACERE Integrated Development Environment. *Proceedings of the 3rd Eclipse Security, AI, Architecture And Modelling Conference on Cloud to Edge Continuum, ESAAM '23, Ludwigsburg, Germany, 17 October 2023*. pp. 62-66 (2023), https://doi.org/10.1145/3624486.3624507
49. Villarreal-Narvaez, S., Şiean, A., Sluÿters, A., Vatavu, R. & Vanderdonckt, J. Informing Future Gesture Elicitation Studies for Interactive Applications that Use Radar Sensing. *Proceedings of the ACM International Conference on Advanced Visual Interfaces, AVI 2022, Frascati, Rome, Italy, 6-10 June 2022*. (2022), https://doi.org/10.1145/3531073.3534475
50. Villarreal-Narvaez, S., Sluÿters, A., Vanderdonckt, J. & Vatavu, R. Brave New GES World: A Systematic Literature Review of Gestures and Referents in Gesture Elicitation Studies. *ACM Comput. Surv.*. **56**, 128:1-128:55 (2024), https://doi.org/10.1145/3636458
51. Wang, S., Song, J., Lien, J., Poupyrev, I. & Hilliges, O. Interacting with Soli: Exploring Fine-Grained Dynamic Gesture Recognition in the Radio-Frequency Spectrum. *Proceedings of the 29th Annual Symposium on User Interface Software and Technology, UIST 2016, Tokyo, Japan, 16-19 October 2016*. pp. 851-860 (2016), https://doi.org/10.1145/2984511.2984565
52. Xia, Z., Oyekoya, O. & Tang, H. Effective Gesture-Based User Interfaces on Mobile Mixed Reality. *Proceedings of the ACM Symposium on Spatial User Interaction, SUI 2022, Online, CA, USA, 1-2 December 2022*. (2022), https://doi.org/10.1145/3565970.3568189
53. Zen, M. & Vanderdonckt, J. Towards an evaluation of graphical user interfaces aesthetics based on metrics. *Proceedings of the IEEE 8th International Conference On Research Challenges In Information Science, RCIS 2014, Marrakech, Morocco, 28-30 May 2014*. pp. 1-12 (2014), https://doi.org/10.1109/RCIS.2014.6861050
54. Zhai, S., Kristensson, P., Appert, C., Andersen, T. & Cao, X. Foundational Issues in Touch-Surface Stroke Gesture Design - An Integrative Review. *Found. Trends Hum. Comput. Interact.*. **5**, pp. 97-205 (2012), https://doi.org/10.1561/1100000012