Colibri: A toolkit for rapid prototyping of networking across realities

Sebastian Hubenschmid[1] * Daniel Immanuel Fink[1] * Johannes Zagermann[1] Jonathan Wieland[1]
Harald Reiterer[1] Tiare Feuchtner[1,2]

[1]University of Konstanz
[2]Aarhus University

Abstract

We present Colibri, an open source networking toolkit for data exchange, model synchronization, and voice communication to support rapid development of distributed cross reality research prototypes. Development of such prototypes often involves multiple heterogeneous components, which necessitates data exchange across a network. However, existing networking solutions are often unsuitable for research prototypes in terms of data privacy, logging capabilities, or latency requirements, as well as requiring significant development resources. In contrast, Colibri is specifically designed for networking in interactive research prototypes: Colibri facilitates the most common tasks for establishing communication between cross reality components with little to no code necessary. We describe the usage and implementation of Colibri and report on its application in three cross reality prototypes to demonstrate the toolkit's capabilities. Lastly, we discuss open challenges to better support the creation of cross reality prototypes.

1 Introduction

Over the past few years, there has been a substantial increase in research prototypes for cross reality (CR) environments. One driving force behind this growth is the availability of increasingly sophisticated toolkits: For example, collaborative mixed reality (MR) systems *"have only recently advanced to the point where researchers can focus deeply on the nuances of supporting collaboration, rather than needing to focus primarily on creating the enabling technology"* [9]. Although there has been a proliferation of toolkits in different areas such as visualization [8,11,26,29] or logging [15,23], networking has been mostly neglected and delegated to commercial solutions. Networking is an essential part in many interactive CR prototypes, for example to support collaboration across realities (e.g., multiple homogeneous [10] or heterogeneous [28] devices) or to connect complementary interfaces [31] (e.g., transitioning between desktop and MR [17]). In contrast to commercial applications, research prototypes have distinct requirements regarding the empirical reproducibility, data availability, latency, and privacy – ruling out externally hosted server software while simplifying development by neglecting edge cases (e.g., anti-cheat precautions) or artificial restrictions (e.g., reducing update rate to save on bandwidth).

To better address the distinct needs of CR research prototypes, we created Colibri (**co**mmunication **libr**ary), an open source networking toolkit for data exchange, model synchronization, and voice communication. In the following sections, we review related work, outline the usage and technical implementation of Colibri, showcase its capabilities based on three prior research projects, and discuss open challenges to further support researchers in creating CR applications.

*Contributed equally

2 Related Work

Toolkits have long been a driving force to reduce development barriers for research prototypes. For example, the *Studierstube* project [27] was one of the enablers of early AR prototypes. Similarly, toolkits such as *ubitrack* [24] and *proximity toolkit* [22] allowed for a proliferation of research prototypes using tracking and proxemic interaction, respectively. Other research-driven frameworks, such as Webstrates [19] and its variants [4,14,25], support developers in seamlessly synchronizing and sharing content across web-based devices. Especially in the field of InfoVis, toolkits such as IATK [8], DXR [29], u2vis [26], and RagRug [11] play an essential role to significantly reduce the effort required to create data visualizations. Recent toolkits such as MRAT [23] and ReLive [16] also include data capturing capabilities to record and analyze MR study data.

As discussed by Friston et al. [12], commercial networking solutions (e.g., Photon Fusion [2]) come with significant limitations, such as supporting only one platform, requiring external third-party services, making strong assumptions – usually in favor of game development requirements such as anti-cheat behavior – and lacking capabilities necessary for research, such as data logging. Beyond commercial solutions, the Ubiq system [12] is most closely related to our own work and addresses many of these issues. However, it focuses on the specific needs of large-scale social virtual reality (VR) systems: For example, Ubiq aims to balance a large number of clients with available throughput, thereby introducing additional latency. In contrast, Colibri aims to provide the ease of use of commercial networking solutions while also making use of the distinct advantages of research prototypes (e.g., ideal lab conditions with little to no latency) to facilitate CR development.

3 Colibri

With Colibri, we aim to provide a networking toolkit that is specifically tailored towards CR research prototypes, favoring a quick and easy setup without extensive programming and reduced complexity through simplifying assumptions (e.g., small-scale deployment in controlled labs), while prioritizing latency. Our toolkit can be integrated into Unity [3] or web applications and uses a web server to manage clients and distribute data. Colibri takes care of common networking tasks such as *data exchange*, *model synchronization*, *persistent data storage*, *voice transmission*, and *logging*.

The following sections describe the usage and implementation of Colibri. For brevity, we focus on its use in Unity projects, but similar methods are available for web clients. Please refer to our project page[1] for more detailed documentation and examples.

3.1 Data Exchange

In the most general case, Colibri facilitates sending data through pub-/sub communication [6]. Data can be published from anywhere in the executed code, as illustrated with the following simple example of sending an integer value (`myInt`) on a `"click"` channel:

```
Sync.Send("click", myInt);
```

[1]`https://github.com/hcigroupkonstanz/Colibri`

The sent data can then be received anywhere in the application by registering a listener as follows:

```
Sync.ReceiveInt("click", (myInt) => {
  /* Will be called whenever an integer on
     "click" channel is received */
});
```

This allows developers to send data of any type, including custom classes (i.e., via JSON serialization), across the network. Published messages are ephemeral and thus best suited for sending events, such as button clicks.

3.2 Model Synchronization

For more complex cases, such as persistently synchronizing data models between different clients, Colibri provides a model synchronization behavior similar to state-of-the-art solutions (e.g., Photon Fusion [2]). Consider, for example, a data model that represents (and is attached to) a cube:

```
public class CubeModel : SyncBehaviour {
    [Sync]
    public Vector3 Position {
      get => transform.position;
      set => transform.position = value;
    }
}
```

We extended Unity's `MonoBehaviour` with a `SyncBehaviour` that keeps track of any properties marked with the `[Sync]` attribute, thereby automatically synchronizing the model's marked properties with other connected clients. For each data model we define a manager component to keep track of all instances of the model:

```
public class CubeManager
  : SyncedBehaviourManager<CubeModel> { }
```

Here, we only need to define an empty manager that inherits `SyncedBehaviourManager` for the data model. We add it to the Unity scene, and provide it with a Unity *prefab* to allow for dynamic instantiation of objects: when an object is created on one client, Colibri's `SyncedBehaviourManager` will automatically instantiate this *prefab* on all connected clients. Models are stored persistently on the server and are automatically retrieved whenever a client connects. Since synchronizing the transform data (i.e., position, rotation, scale) of an object is a common use case, Colibri provides a `SyncTransform` script (and matching manager) out of the box to easily synchronize object transforms without requiring any additional code.

3.3 Persistent Data Storage

While the examples above mainly focus on dynamic data exchange, some actions (e.g., calibrating a room for a study) are usually performed less frequently and require more persistent data storage. For this, Colibri offers persistent data storage, allowing data to be permanently saved on the server and easily shared across all connected clients. For example, consider saving calibration data (e.g., origin and orientation of coordinate system) in a `[Serializable] Calibration` class. Once the calibration is complete, it can be uploaded by specifying a unique name and the data (e.g., instance of `Calibration`) that should be uploaded:

```
Store.Put("calibration", calibrationData);
```

Similarly, any client can then retrieve this data from the server by specifying the name and expected class:

```
await Store.Get<Calibration>("calibration");
```

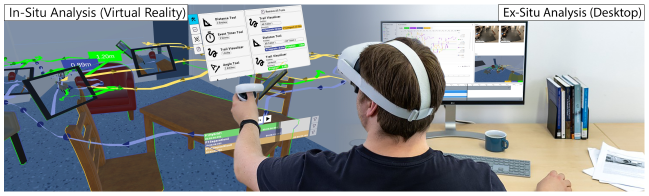
In-Situ Analysis (Virtual Reality)
Ex-Situ Analysis (Desktop)

Figure 1: RELIVE [15] is a CR application that combines an immersive analytics virtual reality view (left) with a synchronized visual analytics desktop view (right). The application state of RELIVE is kept in sync between both views using Colibri.

3.4 Voice Transmission

Similar to prior toolkits [12], Colibri also provides built-in voice transmission for remote collaboration scenarios. Voice transmission can be easily included in any project by adding a predefined *Voice Broadcast* (for audio recording) and *Voice Manager prefab* (for audio playback) to the Unity scene, without the need for any additional code. Each new client will then appear as a customizable entity, thus supporting spatial audio.

3.5 Logging

In our own experience, prototypes often break unexpectedly when deploying to different devices (e.g., Microsoft HoloLens, Android, iOS). In these cases, showing the logging output on the device itself is often not feasible or involves tedious debugging setups (e.g., connecting an iPhone via cable to Apple's Xcode IDE). To aid this troubleshooting process, Colibri can automatically send logging data, such as the console output, to the server and display it as a live data feed on the web interface.

3.6 Technical Implementation

Colibri is built on a client-server architecture, with client implementations for Unity and JavaScript. Our server is written in TypeScript using NodeJS and provides a web interface built with Angular. This allows for server-side broadcasting and recording (e.g., for *voice transmission*) as well as centralized data storage (e.g., for *persistent data storage*). In the future we want to investigate redirecting synchronized data to logging toolkits such as MRAT [23] or RELIVE [15], which could free up client resources.

We employ TCP for the Unity client and WebSockets for web clients for general data synchronization, which ensures that messages arrive safely and in order. In terms of data transmission, every client transmits differential updates as JSON packets, using functional-reactive programming frameworks such as UniRx and RxJS. In addition, we utilize UDP packets for voice transmission, while the persistent data storage uses HTTP requests to support large files.

4 CASE STUDIES

To demonstrate the flexibility and broad application range of Colibri, the following sections describe how we employed Colibri in our own projects, ranging from hybrid user interfaces for single users, to remote multi-user scenarios. While the selected works demonstrate the key features of Colibri, our toolkit has been used in many additional (as-of-yet) unpublished works and is therefore steadily improving. Please refer to the original publications for more details on the specific use cases described below. All included projects are available as open source project[2].

4.1 RELIVE

RELIVE [15] is a cross reality visual analytics framework that combines an immersive VR environment with a non-immersive desktop

[2]`https://github.com/hcigroupkonstanz`

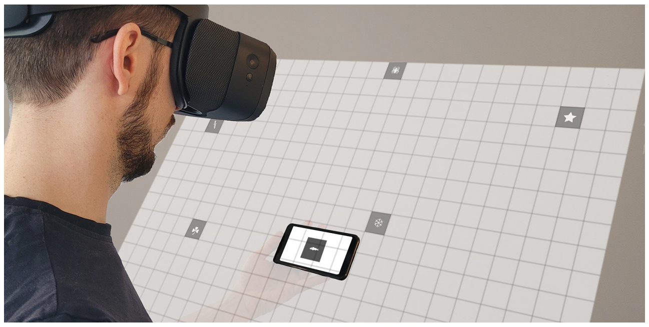
Figure 2: In “ARound the smartphone” [18] we use an AR HWD to virtually extend the physical screen space of a smartphone. In this project network latency was a priority, as the virtual and real screen had to be kept perfectly in sync.

environment to enable in-depth analysis of MR user study data (see Fig. 1). ReLive allows analysts to create visualizations of logged data, such as the movement trail of a study participant based on their recorded positions over time. This visualization can then be inspected both on a desktop in 2D (e.g., as top-down movement path) and in a VR environment in 3D (e.g., embedded within the 3D reconstruction of the original study environment).

Here, we used Colibri to automatically synchronize the entire application state (e.g., visualizations) using the *model synchronization*: For example, we created a data class to represent visualizations (e.g., name, type of visualization, color) and utilized the model synchronization function in Colibri to automatically keep the VR and desktop environments in sync. Since synchronized data models are persistent, refreshing the web page or restarting the VR client automatically retrieves previously created visualizations. /relive supports a variety of actions to aid data analysis, such as taking a screenshot of the VR scene by pressing a button in the desktop interface: For this, we registered a listener as described in *data exchange* (see Sect. 3.1), which was then triggered by sending a message from the web client on a predefined “`screenshot`” channel.

4.2 ARound the Smartphone

In the *“ARound the smartphone”* [18] project, we created a study prototype that uses an augmented reality (AR) head-worn display (HWD) to extend the physical screen of a smartphone with a surrounding virtual screen space (see Fig. 2). Using this setup, study participants were tasked to navigate a grid map with touch gestures, as known from state-of-the-art map applications. Since the visualized map virtually extended beyond the smartphone screen using AR, the map position had to be synchronized between both devices (e.g., from smartphone to AR HWD). We achieved this by attaching the Colibri `SyncTransform` script to the map object in Unity and adding the `SyncManager` *prefab* to the scene – thus keeping the location of both maps (i.e., on the smartphone and in the AR environment) in sync without having to write any code. As Colibri prioritizes low latency, both the smartphone screen and the virtual extension were kept (almost) perfectly in sync, despite the high refresh rate of current AR HWDs (i.e., 90+ Hz).

4.3 Re-locations

Re-locations [10] is a cross reality environment that explores the remapping of different physical workspace layouts in remote collaboration scenarios in AR, to facilitate a common reference frame and the use of deictics. For example, a pair of remote collaborators might work on a shared task using their respective desktop and whiteboard (i.e., two physical workspaces), which are likely in different relative locations within their rooms (see Fig. 3). They see each other as

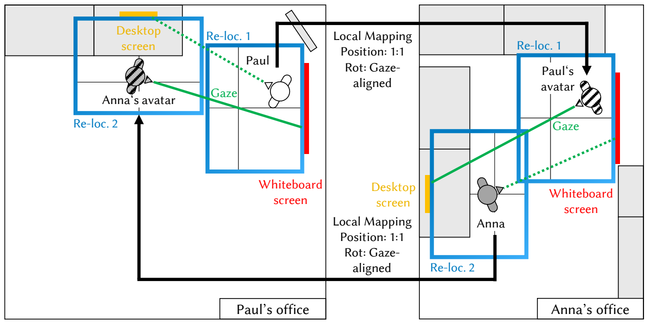

Figure 3: The Re-locations [10] concept semantically corrects gaze visualizations across different user-defined layout areas within physical workspaces during remote collaboration in augmented reality. For example, if Anna is standing close to her desktop while looking at the whiteboard screen (right) the 1:1 mapping of her rotation will be interrupted, and her avatar rotates towards the whiteboard in Paul's environment (left).

virtual avatars, allowing them to be co-present in each other's work spaces. To support a common spatial awareness and understanding, e.g., through semantically correct pointing gestures across these incongruous spaces (e.g., pointing at the whiteboard while sitting in front of the desktop PC), Re-locations automatically remaps the position and rotation of the remote user's avatar to make sense in the local user's workspace layout.

To correctly set up these physical workspaces in the real world, we calibrated each room once and saved the corresponding data using the *persistent data storage* in Colibri. Workspaces were then automatically retrieved by scanning an AR marker containing the ID of the room. In addition, the project uses the Colibri *voice transmission* capabilities to enable spatial audio between collaborators.

5 Discussion and Open Challenges

Although networking is an essential part of many cross reality applications, the development of such research prototypes spans many more areas that require further support. Since sophisticated toolkits can drastically reduce the technological barriers, we discuss four open challenges, which we do not yet find adequately supported by existing toolkits.

Alignment of Coordinate Systems Another common task when combining multiple systems on different points of the reality-virtuality continuum is the alignment of coordinate systems. Nowadays, many devices have powerful integrated tracking capabilities, making external tracking solutions (e.g., Optitrack) unnecessary. However, this also results in each device establishing their own coordinate system, which then needs to be aligned and synchronized with all others. In our experience, these solutions are usually specific to the hardware configurations of each project and often involve a combination of image markers, external tracking systems, or very specific workarounds (e.g., starting the application in the same room position). Here, a unified solution that automatically provides coordinate system alignment could greatly reduce the obstacles of combining multiple CR devices.

Transitional Interfaces Another key area for CR applications are transitional interfaces, which support ongoing interaction across various points on the reality-virtuality continuum. For example, Apple recently showcased one such transitioning technique between AR and VR through turning a knob on the side of their new Apple Vision Pro HWD [1]. Although other techniques have been explored (e.g., [5, 13]), these are usually not easily integrated within other research projects. Here, we see the potential for a library of transitioning methods that could easily be replicated.

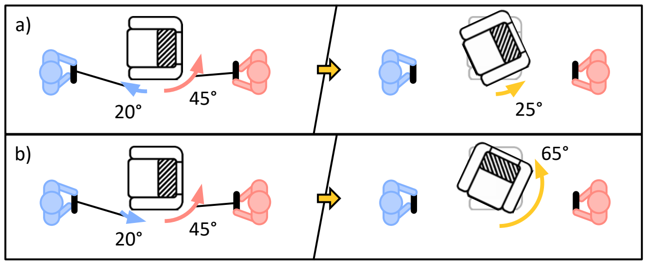

Figure 4: A composition merge policy allows two users to simultaneous manipulate an object, for example by summing up users' rotation inputs [30].

Asymmetric Device Capabilities CR applications can span across multiple heterogeneous devices, such as desktop computers, handheld devices (e.g., smartphones or tablets for AR), or head-worn VR/AR devices. Each device in this ecology has different capabilities or limitations and, at times, a distinct role to fulfill. Here, automatically detecting, communicating, and assigning roles based on capabilities could be beneficial. For example, in an asymmetric collaborative environment for immersive analytics, a desktop device can be useful for displaying visualizations in 2D, while other MR devices could be better suited for 3D visualizations. Here, we could specify data transformations (e.g., [20, 21]) to automatically convert the data, based on the specific capabilities or roles of the display device.

Merge Policies In many collaborative scenarios, multiple collaborators may manipulate an object simultaneously, such as two users moving or rotating an object at the same time (see Fig. 4). In such cases, a merge policy (e.g., averaging or summing up inputs) is necessary to properly incorporate the manipulations of both users (cf. [30]). Although Colibri already supports simultaneous manipulation of the different properties of an object (e.g., one user rotates, another translates) through its differential updates, our toolkit does not yet support simultaneous input.

6 Conclusion

Colibri is a Unity- and web-focused networking toolkit for researchers to quickly prototype cross reality applications. Our toolkit requires little to no code to apply and aids in common networking tasks, such as data exchange, model synchronization, and voice communication. It thus prioritizes low latency to suit the needs of research prototypes. This toolkit was developed and extended through the course of our own projects (i.e., [7, 16]) and is steadily improved through new projects in our research group. However, to adequately support the development of cross reality applications, further work is necessary, for example by communicating asymmetric device capabilities, automating coordinate system alignment, providing a set of transition techniques to quickly change between interfaces at different points on the reality-virtuality continuum, and supporting merge policies. While Colibri presents a first step towards supporting research prototypes, we strongly believe that the development of more sophisticated toolkits will be needed to pave the way for increased adoption of cross reality applications in other application scenarios.

Colibri is available as open source project on GitHub:
`https://github.com/hcigroupkonstanz/Colibri`

Acknowledgments

This research was funded by the *Deutsche Forschungsgemeinschaft* (DFG, German Research Foundation) – Project-ID 251654672 – TRR 161.

References

[1] Introducing Apple Vision Pro: Apple's first spatial computer. https://www.apple.com/newsroom/2023/06/introducing-apple-vision-pro/.

[2] Photon Fusion. https://www.photonengine.com/.

[3] Unity real-time development platform — 3d, 2d, vr and ar engine. https://unity.com/.

[4] S. K. Badam, A. Mathisen, R. Radle, C. N. Klokmose, and N. Elmqvist. Vistrates: A Component Model for Ubiquitous Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):586–596, Jan. 2019. doi: 10.1109/TVCG.2018.2865144

[5] M. Billinghurst, H. Kato, and I. Poupyrev. The MagicBook - moving seamlessly between reality and virtuality. *IEEE Computer Graphics and Applications*, 21(3):6–8, May 2001. doi: 10.1109/38.920621

[6] F. Buschmann, ed. *Pattern-Oriented Software Architecture: A System of Patterns*. Wiley, Chichester ; New York, 1996.

[7] S. Butscher, S. Hubenschmid, J. Müller, J. Fuchs, and H. Reiterer. Clusters, Trends, and Outliers: How Immersive Technologies Can Facilitate the Collaborative Analysis of Multidimensional Data. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*, pp. 1–12. ACM Press, New York, New York, USA, 2018. doi: 10.1145/3173574.3173664

[8] M. Cordeil, A. Cunningham, B. Bach, C. Hurter, B. H. Thomas, K. Marriott, and T. Dwyer. IATK: An Immersive Analytics Toolkit. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 200–209. IEEE, Osaka, Japan, Mar. 2019. doi: 10.1109/VR.2019.8797978

[9] B. Ens, J. Lanir, A. Tang, S. Bateman, G. Lee, T. Piumsomboon, and M. Billinghurst. Revisiting collaboration through mixed reality: The evolution of groupware. *International Journal of Human-Computer Studies*, 131:81–98, Nov. 2019. doi: 10.1016/j.ijhcs.2019.05.011

[10] D. I. Fink, J. Zagermann, H. Reiterer, and H.-C. Jetter. Re-locations: Augmenting Personal and Shared Workspaces to Support Remote Collaboration in Incongruent Spaces. *Proceedings of the ACM on Human-Computer Interaction*, 6(ISS):1–30, Nov. 2022. doi: 10.1145/3567709

[11] P. Fleck, A. S. Calepso, S. Hubenschmid, M. Sedlmair, and D. Schmalstieg. RagRug: A Toolkit for Situated Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 29(7):3281–3297, July 2023. doi: 10.1109/TVCG.2022.3157058

[12] S. J. Friston, B. J. Congdon, D. Swapp, L. Izzouzi, K. Brandstätter, D. Archer, O. Olkkonen, F. J. Thiel, and A. Steed. Ubiq: A System to Build Flexible Social Virtual Reality Experiences. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–11. ACM, Osaka Japan, Dec. 2021. doi: 10.1145/3489849.3489871

[13] R. Grasset, J. Looser, and M. Billinghurst. Transitional interface: Concept, issues and framework. In *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 231–232, Oct. 2006. doi: 10.1109/ISMAR.2006.297819

[14] T. Horak, A. Mathisen, C. N. Klokmose, R. Dachselt, and N. Elmqvist. Vistribute: Distributing Interactive Visualizations in Dynamic Multi-Device Setups. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*, pp. 1–13. ACM Press, Glasgow, Scotland Uk, 2019. doi: 10.1145/3290605.3300846

[15] S. Hubenschmid, J. Wieland, D. I. Fink, A. Batch, J. Zagermann, N. Elmqvist, and H. Reiterer. ReLive: Bridging In-Situ and Ex-Situ Visual Analytics for Analyzing Mixed Reality User Studies. In *CHI Conference on Human Factors in Computing Systems*, pp. 1–20. ACM, New Orleans LA USA, Apr. 2022. doi: 10.1145/3491102.3517550

[16] S. Hubenschmid, J. Zagermann, S. Butscher, and H. Reiterer. STREAM: Exploring the Combination of Spatially-Aware Tablets with Augmented Reality Head-Mounted Displays for Immersive Analytics. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, pp. 1–14. Association for Computing Machinery, New York, NY, USA, May 2021. doi: 10.1145/3411764.3445298

[17] S. Hubenschmid, J. Zagermann, D. Fink, J. Wieland, T. Feuchtner, and H. Reiterer. Towards asynchronous hybrid user interfaces for cross-reality interaction. In H.-C. Jetter, J.-H. Schröder, J. Gugenheimer, M. Billinghurst, C. Anthes, M. Khamis, and T. Feuchtner, eds.,

ISS'21 Workshop Proceedings: "Transitional Interfaces in Mixed and Cross-Reality: A New Frontier?", 2021. doi: 10.18148/kops/352-2-84mm0sggczq02

[18] S. Hubenschmid, J. Zagermann, D. Leicht, H. Reiterer, and T. Feuchtner. ARound the Smartphone: Investigating the Effects of Virtually-Extended Display Size on Spatial Memory. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–15. ACM, Hamburg Germany, Apr. 2023. doi: 10.1145/3544548.3581438

[19] C. N. Klokmose, J. R. Eagan, S. Baader, W. Mackay, and M. Beaudouin-Lafon. *Webstrates*: Shareable Dynamic Media. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pp. 280–290. ACM, Charlotte NC USA, Nov. 2015. doi: 10.1145/2807442.2807446

[20] B. Lee, M. Cordeil, A. Prouzeau, B. Jenny, and T. Dwyer. A Design Space For Data Visualisation Transformations Between 2D And 3D In Mixed-Reality Environments. In *CHI Conference on Human Factors in Computing Systems*, pp. 1–14. ACM, New Orleans LA USA, Apr. 2022. doi: 10.1145/3491102.3501859

[21] B. Lee, A. Satyanarayan, M. Cordeil, A. Prouzeau, B. Jenny, and T. Dwyer. Deimos: A Grammar of Dynamic Embodied Immersive Visualisation Morphs and Transitions. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–18. ACM, Hamburg Germany, Apr. 2023. doi: 10.1145/3544548.3580754

[22] N. Marquardt, R. Diaz-Marino, S. Boring, and S. Greenberg. The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology - UIST '11*, p. 658. ACM Press, Santa Barbara, California, USA, 2011. doi: 10.1145/2047196.2047238

[23] M. Nebeling, M. Speicher, X. Wang, S. Rajaram, B. D. Hall, Z. Xie, A. R. E. Raistrick, M. Aebersold, E. G. Happ, J. Wang, Y. Sun, L. Zhang, L. E. Ramsier, and R. Kulkarni. MRAT: The Mixed Reality Analytics Toolkit. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–12. ACM, Honolulu HI USA, Apr. 2020. doi: 10.1145/3313831.3376330

[24] J. Newman, M. Wagner, M. Bauer, A. MacWilliams, T. Pintaric, D. Beyer, D. Pustka, F. Strasser, D. Schmalstieg, and G. Klinker. Ubiquitous Tracking for Augmented Reality. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 192–201. IEEE, Arlington, VA, USA, 2004. doi: 10.1109/ISMAR.2004.62

[25] R. Rädle, M. Nouwens, K. Antonsen, J. R. Eagan, and C. N. Klokmose. Codestrates: Literate Computing with Webstrates. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pp. 715–725. ACM, Québec City QC Canada, Oct. 2017. doi: 10.1145/3126594.3126642

[26] P. Reipschläger, T. Flemisch, and R. Dachselt. Personal Augmented Reality for Information Visualization on Large Interactive Displays. *IEEE Transactions on Visualization and Computer Graphics*, Feb. 2021. doi: 10.1109/TVCG.2020.3030460

[27] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnação, M. Gervautz, and W. Purgathofer. The Studierstube Augmented Reality Project. *Presence: Teleoperators and Virtual Environments*, 11(1):33–54, Feb. 2002. doi: 10.1162/105474602317343640

[28] J.-H. Schröder, D. Schacht, N. Peper, A. M. Hamurculu, and H.-C. Jetter. Collaborating Across Realities: Analytical Lenses for Understanding Dyadic Collaboration in Transitional Interfaces. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–16. ACM, Hamburg Germany, Apr. 2023. doi: 10.1145/3544548.3580879

[29] R. Sicat, J. Li, J. Choi, M. Cordeil, W.-K. Jeong, B. Bach, and H. Pfister. DXR: A Toolkit for Building Immersive Data Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):715–725, Jan. 2019. doi: 10.1109/TVCG.2018.2865152

[30] J. Wieland, J. Zagermann, J. Muller, and H. Reiterer. Separation, Composition, or Hybrid? – Comparing Collaborative 3D Object Manipulation Techniques for Handheld Augmented Reality. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 403–412. IEEE, Bari, Italy, Oct. 2021. doi: 10.1109/ISMAR52148.2021.00057

[31] J. Zagermann, S. Hubenschmid, P. Balestrucci, T. Feuchtner, S. Mayer, M. O. Ernst, A. Schmidt, and H. Reiterer. Complementary interfaces for visual computing. *it - Information Technology*, 0(0), Sept. 2022. doi: 10.1515/itit-2022-0031