Traceable Asynchronous Workflows in Video Retrieval with vitrivr-VR

Florian Spiess[1][0000−0002−3396−1516], Silvan Heller[1][0000−0001−5386−330X], Luca Rossetto[2][0000−0002−5389−9465], Loris Sauter[1][0000−0001−8046−0362], Philipp Weber[1*][0000−0003−4311−2363], and Heiko Schuldt[1][0000−0001−9865−6371]

[1] Department of Mathematics and Computer Science
University of Basel, Basel, Switzerland
{firstname.lastname}@unibas.ch, *phil.weber@stud.unibas.ch
[2] Department of Informatics, University of Zurich, Zurich, Switzerland
rossetto@ifi.uzh.ch

Abstract. Virtual reality (VR) interfaces allow for entirely new modes of user interaction with systems and interfaces. Much like in physical workspaces, documents, tools, and interfaces can be used, put aside, and used again later. Such asynchronous workflows are a great advantage of virtual environments, as they enable users to perform multiple tasks in an interleaved manner. However, VR interfaces also face new challenges, such as text input without physical keyboards, and the analysis of such asynchronous workflows. In this paper we present the version of vitrivr-VR participating in the Video Browser Showdown (VBS) 2023. We describe the current state of our system, with a focus on improvements in text input methods and logging of asynchronous workflows.

Keywords: Video Browser Showdown · Virtual Reality · Interactive Video Retrieval · Content-based Retrieval.

1 Introduction

Multimedia retrieval can be viewed as a mostly linear workload: a user formulates their information need as a query and receives results accordingly. However, in many cases, particularly when the information need is fuzzy and unrefined, users will iterate on their original query, by refining it with insights gained from previous results. It may even occur, that newer results alter the user's understanding of previous results, prompting them to return to a previous query to reevaluate its results. Such asynchronous workflows are particularly useful for analytical workloads, where results from different queries may need to be compared and contrasted. While they provide a great advantage for many use cases, these convoluted workflows are much more challenging to analyze for usage patterns.

Text input is a requirement of many applications, but with the increasing popularity of cross-modal retrieval, it is of particular importance in multimedia retrieval. Despite several existing approaches, fast and reliable text input is still a challenge in virtual reality (VR).

vitrivr-VR is a VR multimedia retrieval and analytics research system leveraging VR to support efficient text input and enabling asynchronous analytical workflows. To evaluate the effectiveness of its approach, vitrivr-VR participates in multimedia retrieval campaigns, such as the Video Browser Showdown (VBS) [4], in which it has participated multiple times in the past [9,10].

In this paper, we describe the version of vitrivr-VR participating in the VBS 2023, with a focus on asynchronous workflows and our improved methods for text input. Section 2 describes the system and its components, Section 3 introduces our novel text input method, Section 4 relates our efforts to record and analyze asynchronous interactions, and Section 5 concludes.

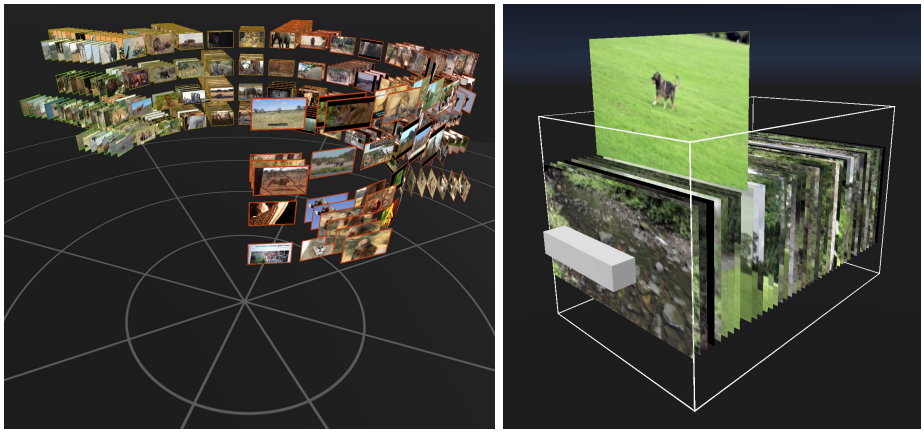
2 vitrivr-VR

vitrivr-VR is a an experimental VR multimedia retrieval and analytics system. The vitrivr-VR stack consists of three components: the *Cottontail DB* column store [3], the *Cineast* retrieval engine [8], and the *vitrivr-VR*[3] virtual reality user interface. The user interface of vitrivr-VR is developed in Unity with C# and is capable of running on any XR platform supporting OpenXR, but has been designed for and tested on the HTC Vive Pro and Valve Index. To enable multimedia retrieval, vitrivr-VR communicates with Cineast through a RESTful API provided with OpenAPI specifications. Cineast enables a variety of retrieval features, such as OCR and ASR search, as well as color and deep-learning based features. Cottontail DB provides Boolean, text and kNN vector space retrieval through a gRPC connection.

The main focus of vitrivr-VR is to explore and exploit the affordances of VR user interfaces. As such, vitrivr-VR provides a sophisticated yet simple interaction system, on which the three main phases of multimedia analytical workloads are built: *query formulation*, *result set exploration*, and *media item inspection*.

The interaction system of vitrivr-VR aims to be as expressive as possible, while maintaining simplicity and intuitiveness by resembling the interaction with real-world objects. To this end, the system supports two main kinds of interaction: *grabbing* objects and UI elements by utilizing the grab gesture on Index controllers or the grip button on other controllers, and *triggering* objects and UI elements by pressing the controller trigger while hovering over an object or UI element. Through only these two interactions, interface elements can be designed to allow complex interactions. In addition to these two primary interactions, the interaction system also supports pointer-style interactions with canvas-based UI elements. Building on this interaction system, vitrivr-VR provides interfaces for a multitude of input modalities such as text, concept, and pose based query terms. Previous iterations of the VBS have shown that being able to include temporal order in a query is an advantage. For this reason vitrivr-VR allows users to specify the temporal order of query terms by grabbing and dragging small representations of them into the desired order.

[3] https://github.com/vitrivr/vitrivr-vr



(a) (b)

Fig. 1. Screenshots of the vitrivr-VR user interface; temporal results view (1a) and multimedia drawer segment view (1b).

Once a query has been formulated and results have been returned, they are displayed through a cylindrical results display. Results displays for temporal queries show each result sequence as a stack that can be grabbed and pulled towards the user for easier viewing, as seen in Figure 1a. The cylindrical results displays can be rotated to reveal further results. Once a video segment of interest has been identified, it can be popped out of the results display and positioned freely and independently in virtual space. This video detail view allows the video associated with the selected clip to be played and explored. To gain an even better overview of the selected video, the detail view allows a multimedia drawer view to be created. This multimedia drawer, which can be seen in Figure 1b, displays the most representative frame of every video segment of a video in a temporally ordered horizontal stack, similar to files in a drawer. A handle at the front of the drawer allows it to be extended, increasing the gap between frames and allowing them to be viewed more clearly. By moving the controller through the drawer and hovering over a frame, it is moved above the drawer, allowing it to be viewed individually. By selecting a hovered frame, the associated video detail view is skipped to the respective video segment.

3 Text Input in Virtual Reality

With the current trend of multimedia retrieval towards machine learning enabled text-based query methods such as CLIP [7] and our own visual-text co-embedding [9], effective text input methods are becoming a necessity for multimedia retrieval systems. While this does not pose much of a problem for conventional desktop-based multimedia retrieval systems, fast and reliable text input in

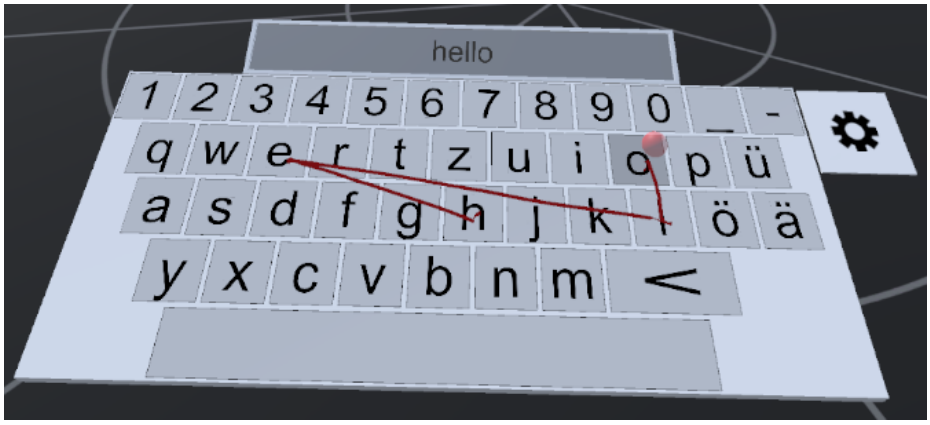


Fig. 2. The word-gesture keyboard used in vitrivr-VR, as a word is being input. The red sphere indicates the controller, the red line is the graph drawn by the user and the small display above the keyboard shows the current prediction.

VR is still a difficult challenge. Many different text input methods have already been developed for VR [1,2,11], with none being able to match the speed and precision of physical keyboards [5]. These different methods vary greatly in text input speed and flexibility of their input vocabulary, and as such are differently well suited for different tasks and use-cases. To take advantage of the strengths and overcome the weaknesses of these methods, vitrivr-VR uses a combination.

As a primary text input method, vitrivr-VR uses an on-device speech-to-text approach based on Mozilla DeepSpeech[4] and publicly available as a Unity Package Manager (UPM) package[5]. While speech-to-text methods are unable to match physical keyboards in precision and input flexibility, usually being limited to a predetermined set of words or phonemes, they allow fast, hands-free text input. The two main limitations of speech-to-text methods are the used vocabulary and misinterpretations of speech. Both of these limitations lead to incorrect or missing words in the transcribed text and require manual correction, often on the level of individual characters. Such corrections are usually difficult if not impossible using speech-to-text. For this purpose, vitrivr-VR offers a VR word-gesture keyboard as a secondary text-input method.

Previously, vitrivr-VR used a very simple virtual touch keyboard as secondary text input method. Although this method functioned well to type individual words or make small corrections in VR, the simplicity of the approach resulted in some limitations. The touch activation of the keyboard, while allowing intuitive pressing of the virtual keys without the use of controller buttons, resulted in occasional accidental inputs due to the user moving a controller through it outside of their field of view. This touch activation also caused occasional dou-

[4] https://github.com/mozilla/DeepSpeech
[5] https://github.com/Spiess/deep-speech-upm

ble inputs, due to the controller passing fully through a key and reactivating it when moving back. Another limitation of this simple approach is the very limited typing speed when writing entire words. To address these limitations we have replaced the simple touch keyboard with our VR word-gesture keyboard implementation shown in Figure 2. To avoid accidental input, our word-gesture keyboard requires the press of the controller trigger to register a button press. Our implementation is based on SHARK^2 [6] and allows the fast input of individual words through gestures on the keyboard. To input a word, its graph must be traced on the keyboard, starting from the key of the first character and passing through each character of the word in order using straight line paths between characters. Our word-gesture keyboard is available as an open-source UPM package on GitHub[6].

4 Interaction Logging for Asynchronous Workflows

vitrivr-VR enables users to jump back to results displays of previous queries and to keep selected video detail views in the virtual space as potential result candidates, or for future reference. These features, combined with the ability to place results in a corner of virtual space and only come back to them after doing something unrelated, enable highly asynchronous workflows and facilitate analytical usage through the comparison of query result sets and items.

While these asynchronous workflows are in many regards a strength of vitrivr-VR, they make analysis of interaction data much more difficult. The VBS has often encouraged, and sometimes required, that in addition to task solution submissions participating systems log interaction data and raw query results. While this is enough data to allow submitted results to be traced back to their origin query for most conventional systems that only support linear workflows, doing the same in vitrivr-VR is often very difficult, and in certain cases impossible. By exploring beyond the exact returned set of results through interaction methods like the multimedia drawer, and keeping them in the virtual space over the course of several queries, it quickly becomes ambiguous which query and interactions led to any specific submission.

To capture the flow of data, we improved the interaction logging of vitrivr-VR with identifying information allowing the path of interactions resulting in a submission to be uniquely identified. Through the improved data logging, we are not only able to determine in greater detail which query and interactions led to a submission, but also allows us to analyze the different paths that can lead to a submission.

5 Conclusion

In this paper we present the vitrivr-VR system, in the state in which we intend to participate in the VBS 2023. We describe the system with a focus on changes to text input and logging of asynchronous interactive workflows.

[6] https://github.com/Philipp1202/WGKeyboard

Acknowledgements

This work was partly supported by the Swiss National Science Foundation through projects "Participatory Knowledge Practices in Analog and Digital Image Archives" (contract no. 193788) and "MediaGraph" (contract no. 202125).

References

1. Boletsis, C., Kongsvik, S.: Controller-based Text-input Techniques for Virtual Reality: An Empirical Comparison. International Journal of Virtual Reality **19**(3), 2–15 (2019). https://doi.org/10.20870/IJVR.2019.19.3.2917
2. Chen, S., Wang, J., Guerra, S., Mittal, N., Prakkamakul, S.: Exploring Word-gesture Text Entry Techniques in Virtual Reality. In: CHI Conference on Human Factors in Computing Systems. ACM (2019). https://doi.org/10.1145/3290607.3312762
3. Gasser, R., Rossetto, L., Heller, S., Schuldt, H.: Cottontail DB: An Open Source Database System for Multimedia Retrieval and Analysis. In: International Conference on Multimedia. pp. 4465–4468. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3394171.3414538
4. Heller, S., Gsteiger, V., Bailer, W., Gurrin, C., Jónsson, B.Þ., Lokoč, J., Leibetseder, A., Mejzlík, F., Peška, L., Rossetto, L., Schall, K., Schoeffmann, K., Schuldt, H., Spiess, F., Tran, L.D., Vadicamo, L., Veselý, P., Vrochidis, S., Wu, J.: Interactive video retrieval evaluation at a distance: Comparing sixteen interactive video search systems in a remote setting at the 10th Video Browser Showdown. International Journal of Multimedia Information Retrieval **11**(1), 1–18 (2022). https://doi.org/10.1007/s13735-021-00225-2
5. Jin Ryong Kim, Tan, H.Z.: A study of touch typing performance with keyclick feedback. In: IEEE Haptics Symposium. pp. 227–233 (2014). https://doi.org/10.1109/HAPTICS.2014.6775459
6. Kristensson, P.O., Zhai, S.: SHARK^2: A large vocabulary shorthand writing system for pen-based computers. In: Symposium on User Interface Software and Technology. pp. 43–52. ACM (2004). https://doi.org/10.1145/1029632.1029640
7. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning Transferable Visual Models From Natural Language Supervision. arXiv (2021). https://doi.org/10.48550/ARXIV.2103.00020
8. Rossetto, L.: Multi-Modal Video Retrieval. Ph.D. thesis, University of Basel (2018). https://doi.org/10.5451/unibas-006859522
9. Spiess, F., Gasser, R., Heller, S., Parian-Scherb, M., Rossetto, L., Sauter, L., Schuldt, H.: Multi-modal Video Retrieval in Virtual Reality with vitrivr-VR. In: MultiMedia Modeling. pp. 499–504. Springer (2022). https://doi.org/10.1007/978-3-030-98355-0_45
10. Spiess, F., Gasser, R., Heller, S., Rossetto, L., Sauter, L., Schuldt, H.: Competitive Interactive Video Retrieval in Virtual Reality with vitrivr-VR. In: MultiMedia Modeling. pp. 441–447. Springer (2021). https://doi.org/10.1007/978-3-030-67835-7_42
11. Yu, C., Gu, Y., Yang, Z., Yi, X., Luo, H., Shi, Y.: Tap, Dwell or Gesture? Exploring Head-Based Text Entry Techniques for HMDs. In: CHI Conference on Human Factors in Computing Systems. pp. 4479–4488. ACM (2017). https://doi.org/10.1145/3025453.3025964