Multimedia Tools and Applications
https://doi.org/10.1007/s11042-023-15082-w

1228: ADVANCES IN LIFELOG RETRIEVAL AT THE LIFELOG SEARCH CHALLENGE 2021

A tale of two interfaces: vitrivr at the lifelog search challenge

Silvan Heller[1] · **Florian Spiess**[1] · **Heiko Schuldt**[1]

Received: 29 April 2022 / Revised: 1 February 2023 / Accepted: 2 March 2023

Abstract

The past decades have seen an exponential growth in the amount of data which is produced by individuals. Smartphones which capture images, videos and sensor data have become commonplace, and wearables for fitness and health are growing in popularity. Lifelog retrieval systems aim to aid users in finding and exploring their personal history. We present two systems for lifelog retrieval: vitrivr and vitrivr-VR, which share a common retrieval model and backend for multi-modal multimedia retrieval. They differ in the user interface component, where vitrivr relies on a traditional desktop-based user interface and vitrivr-VR has a Virtual Reality user interface. Their effectiveness is evaluated at the Lifelog Search Challenge 2021, which offers an opportunity for interactive retrieval systems to compete with a focus on textual descriptions of past events. Our results show that the conventional user interface outperformed the VR user interface. However, the format of the evaluation campaign does not provide enough data for a thorough assessment and thus to make robust statements about the difference between the systems. Thus, we conclude by making suggestions for future interactive evaluation campaigns which would enable further insights.

Keywords Lifelog retrieval · Lifelog search challenge · Interactive retrieval system evaluation · Virtual reality

S. Heller and F. Spiess have contributed equally to this work.

✉ Silvan Heller
silvan.heller@unibas.ch

Florian Spiess
florian.spiess@unibas.ch

Heiko Schuldt
heiko.schuldt@unibas.ch

[1] Department of Mathematics and Computer Science, University of Basel, Basel, Switzerland

1 Introduction

With the increasing availability, affordability and quality of smart, wearable devices, such as smartwatches, fitness trackers, and body cams, the ability of individuals to continuously record their lives in the form of images, videos, audio, and biometric data has increased greatly within the last two decades. Whether to ensure no social-media-worthy situation goes unrecorded, to analyze personal fitness, or simply to keep a digital record of ones experiences, the practice of recording streams of such data, called lifelogging, has become more commonplace with this increased affordability and availability of recording devices. Depending on the frequency and scope of data collection during lifelogging, collections generated in this way, so-called lifelogs, can reach exceptionally large volumes very quickly, making manual management impractical. As such, with the increasing size of lifelogs, it becomes increasingly difficult to analyze and gain insight or retrieve a specific piece of information without efficient computational methods, especially due to the multi-modal nature of lifelog data.

Lifelogs traditionally contain a variety of data with implicit and explicit relations, ranging from videos, to images and structured data such as GPS or heart rate. Lifelog retrieval systems should thus ideally support multiple types of data, and query interfaces supporting different modalities lend themselves well for this use case.

One way to evaluate contributions in the area of multimedia retrieval systems is through interactive retrieval competitions, where different user-system combinations compete with each other. The Lifelog Search Challenge, "modelled on the successful Video Browser Showdown (VBS) [13]" [8], offers such a platform. While there are many different configurations for such evaluations [18], LSC has so far focused on Known-Item Search Tasks, where users are provided textual hints and use their retrieval system to find a target lifelog moment. Thanks to advances in tools [28, 33], these evaluations have also been held online in the past years [13, 27].

Recent evaluation campaigns have also shown that in addition to traditional desktop user interfaces, novel user input methods such as Virtual Reality (VR) can provide new and intuitive retrieval interfaces, while offering competitive performance [9, 13].

In this paper, we present two systems which have participated at LSC in past years: vitrivr[1] [31] with its conventional desktop user interface, vitrivr-ng and its Virtual Reality counterpart, vitrivr-VR.[2] Both systems use the same retrieval engine [30] and database [4], but offer different query and interaction methods. vitrivr-VR builds upon the vitrivr system, which has been used in a variety of contexts such as cultural heritage [20, 22, 34], retrieval for speech transcription [39], and video retrieval [11].

Our contribution is threefold: First, we describe in detail how the shared backend of vitrivr and vitrivr-VR handles the lifelog data used at LSC, which features are used and how the retrieval model works. Second, we describe the user interfaces vitrivr-ng and vitrivr-VR for lifelog retrieval, and the implications on the retrieval approach. Third, we compare and analyze the performance of the two interfaces at LSC, providing insights into the differences between the traditional desktop-based system and the VR system for lifelog retrieval. The fact that both systems use a common retrieval engine allows for an interesting comparison. All components of vitrivr and vitrivr-VR are fully open-source.

[1] https://vitrivr.org

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

The rest of this article is structured as follows: In Section 2, we describe the data and retrieval model used by both systems. Section 3 compares the architecture of the two systems, introducing common components and highlighting key differences. The user interfaces are described in Section 4, Section 5 analyses the results of the participation at LSC 2021, and Section 6 concludes.

2 Data and retrieval model

In 2021, vitrivr participated for the third time [12, 14, 26] with its traditional desktop interface vitrivr-ng, and vitrivr-VR for the first time [38] in the LSC. Between 2019 and 2021, our conceptual approach to lifelog retrieval has evolved, and this section describes our model for the 2021 iteration in detail.

This section contains our data model, the internal representation of the dataset, in Section 2.1, which is followed by an in-depth look at the retrieval model for multimodal lifelog retrieval in Section 2.2. The content draws from previous LSC participations [12, 14, 26, 38] and existing work on the conceptual underpinnings of vitrivr [5–7, 15, 23]. It contains some simplifications compared to the actual implementation due to the focus on lifelog retrieval.

2.1 Data model

Fundamentally, we are provided with two different types of data for the LSC: The images captured by the lifeloggers and the metadata accompanying each image[3]. In our data model, the smallest unit of retrieval is called a *segment*, and segments are grouped into *objects*. This distinction comes from the focus of vitrivr on multimedia, and enables us to easily distinguish different result aggregations such as ranking segments individually, or sorting by object. In the context of lifelog retrieval, each individual image is considered a *segment*, and each day is an *object* consisting of all segments within that day. Figure 1 shows an example of two objects (days), each consisting of multiple segments (images).

2.1.1 Multimedia data

As our retrieval model is centered on *multimedia* retrieval, there are different types of segments and objects depending on the type of multimedia. For example, in video retrieval the original videos are represented as objects, on which we perform shot segmentation to generate partial video sequences (*shots*) which are represented as segments. At the LSC we are provided with individual images with the associated timestamp of capture, which we use as our smallest unit of retrieval, and thus consider each image a segment. Different summarizations are possible, such as by week, month, or semantic events. We have chosen to select a day as our abstraction level, which is illustrated in Fig. 1.

We assign to each object o, i.e., each day, a unique identifier and store the path $path$ where all images for that day are located.[4] No content is stored for an object, as the actual

[3]Provided annotations by external APIs, e.g., OCR data and textual descriptions can be considered as metadata in this context.

[4]If our data model would only consider the LSC dataset, we would not necessarily need the object abstraction, but as the model and systems are built for general purpose multimedia retrieval, we need to select an object abstraction.

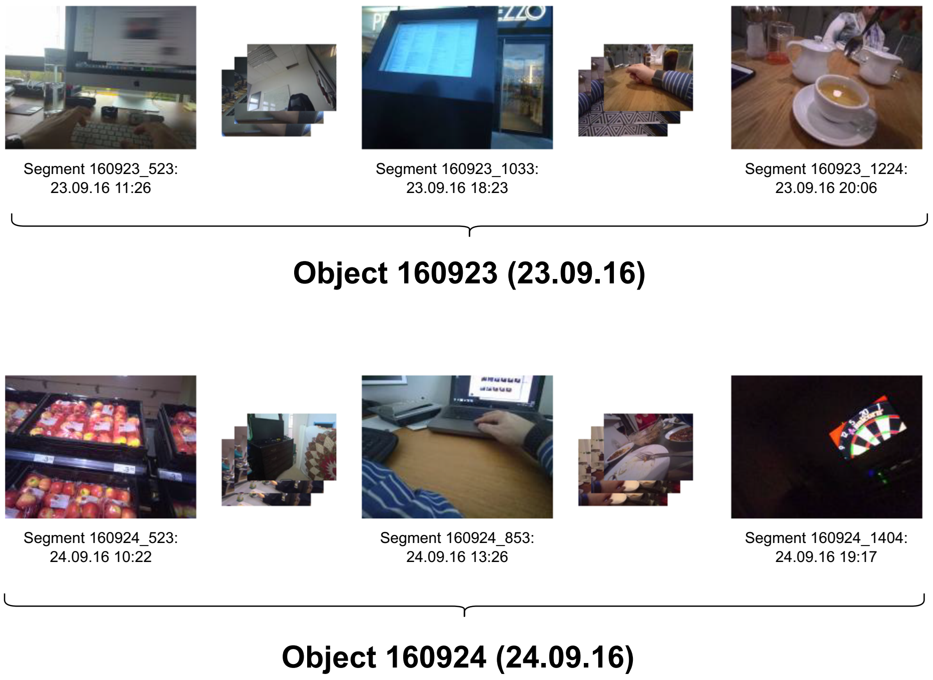


Fig. 1 The data model groups segments (individual images) into objects (days). The figure shows selected segments from two consecutive days which are grouped into two different objects. A segment is our smallest unit of retrieval

content (images) is stored per segment. All segments for a given object are stored in a single folder, which is stored as $path$.

$$o := \langle objectId, path \rangle$$

Since each segment s is part of an object, we generate a unique identifier per segment and store it alongside the objectId, the sequence number (i.e., its index within a day), and its timestamp. The actual image is stored in the file system and fetched on demand. For example, the segment with the tea cup in Fig. 1 has the sequence number 1224, is taken at 20:06 and belongs to the object 160923 (all images from the 23^{rd} of September 2016)

$$s := \langle segmentId, objectId, sequenceNum, timestamp, path \rangle$$

2.1.2 Metadata

We separate provided metadata into two categories: metadata which is used for Boolean retrieval, and metadata which is used for content-based retrieval (e.g., text).

Metadata for boolean retrieval The metadata model looks the same for objects and segments. Given the unique identifier id of either one of the two, a metadata tuple m is defined as follows:

$$m := \langle id, domain, key, value \rangle \text{ with } id \in \{segmentId, objectId\}$$

The most commonly used *domain* values are *technical* e.g., for resolution, *provided* for metadata which is provided by external sources, but this also allows to add e.g., *exif* as a domain for metadata which comes with an image. An example tuple storing the timezone for our example segment from Fig. 1 could thus be ⟨160923_1224, provided, timezone, Europe/Dublin⟩.

Metadata for content-based retrieval Provided annotations such as OCR and textual descriptions are used during retrieval along with other features which we extract ourselves, which is discussed in the next section.

2.2 Retrieval model

At its core, our retrieval model is based on a *late fusion* of different features, which each consider different aspects of a query. Each feature is responsible for returning the best matching segments for a query, and the scores are fused together in a second step. More formally, each feature produces a list of *scored segments* $\hat{s} = \langle segmentid, score\rangle$ containing a segment identifier and the score for said segment (scores are bound, $0 \leq score \leq 1$).

In the following, we will first introduce the features which are available, then how they can be queried, and how different query modalities can be combined to generate a single ranked list. Afterwards, extensions which were made to enable queries with temporal context are briefly described.

2.2.1 Features for lifelog images

Our retrieval engine implements a variety of features for different types of media [6, 23]. In this section we briefly list the features used for lifelog retrieval.

Visual-Text Co-Embedding: Recent evaluations have shown that features transforming text and images into a joint embedding space are very competitive in interactive video retrieval [13, 19], and the popularity of such embeddings for systems participating at LSC [1] show that they are also considered essential for state-of-the-art systems supporting lifelog retrieval. We use a very simple yet effective visual-text co-embedding based on pre-trained visual and textual feature encoders and embedding networks trained by us [37]. Figure 2 shows a conceptual overview of this feature. The pre-trained feature encoders used are the InceptionResNetV2 [40] for visual features and the Universal

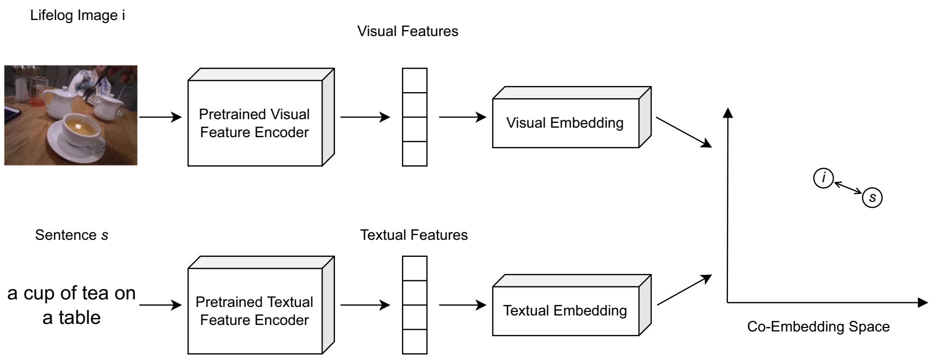


Fig. 2 Conceptual overview of our visual-text co-embedding method (adapted from [37])

Sentence Encoder [3] for textual features. Our models were trained on a mixture of captioned image and video data from Flickr30k [45], Microsoft COCO [17], MSR-VTT [44], TextCaps [36], TGIF [16], and VaTeX [43]. For regularization, the output vectors are normalized to the 256 dimensional hypersphere. During retrieval, the similarity between a query vector qv and a known segment vector sv is calculated as the Euclidean distance between the two vectors transformed to a similarity score with a linear correspondence function with a factor of 2:

$$sc_{\text{vtce}}(qv_i, sv_i) = \max\left(0, 1 - \frac{\sqrt{\sum_{i=1}^{256}(qv_i - sv_i)^2}}{2}\right)$$

Boolean Metadata Retrieval: We focus on two types of metadata queries: *range* for e.g., hour and *option* for e.g., day or timezone. Range queries get transformed to simple *BETWEEN* request to the database, and for options we support various Boolean operators such as $=, \leq, or\ \geq$.

Concept Classification / Tag Retrieval: In contrast to the visual-text co-embedding feature, which takes free text input, this feature assumes a pre-defined set of concepts (or *tags*) have been classified. In the past, we have used 3DConvNets [42] or external APIs [32]. For LSC 2021, we have used the provided annotations and their score which is used by our tag retrieval feature. More formally, given a set of user-provided tags $T = \langle t_1, t_2, \ldots, t_k\rangle$, we retrieve for every tag all matching segments. Given a relevance score per segment-tag combination $rs(s, t)$, the score of a segment is computed by summing all scores and normalizing:

$$sc_{\text{tag}}(s, T) = \frac{\sum_{i=1}^{k} rs(s, t_i)}{k}$$

Geospatial NNS: We support simple queries in the form "near this point", meaning the retrieval engine takes as input a single point of the coordinate system as a query $cq = \langle lat_{cq}, lon_{cq}\rangle$. This is then compared to the coordinates of each segment, $cs = \langle lat_{cs}, lon_{cs}\rangle$, and compared using the Haversine distance hav. Transformation to a similarity score is done using hyperbolic correspondence with a configurable value $divisor$ which determines the distance at which the score should be 0.5[5]

$$sc_{\text{geo}}(cq, cs) = \max\left(0, 1 - \frac{1}{1 + \frac{\text{hav}(cq, cs)}{divisor}}\right)$$

OCR: Over the years, we have experimented with different OCR features such as a combination of EAST [46] and CRNN [35] in 2019 [32], but have not used OCR in 2020 and 2021. For 2022, we will use HyText [41].

2.2.2 Query model

Our retrieval model allows combining different query modalities such as text and Boolean retrieval (and thus features) in different ways. We show an example query in Fig. 3. At the highest level, users can specify temporal sequences using multiple *query containers*, such as first seeing "a full plate of food" and then "a cup of tea on a table". Each query container

[5]The divisor can be configured based on the dataset and user preference, and was set to 100'000 for LSC 2021

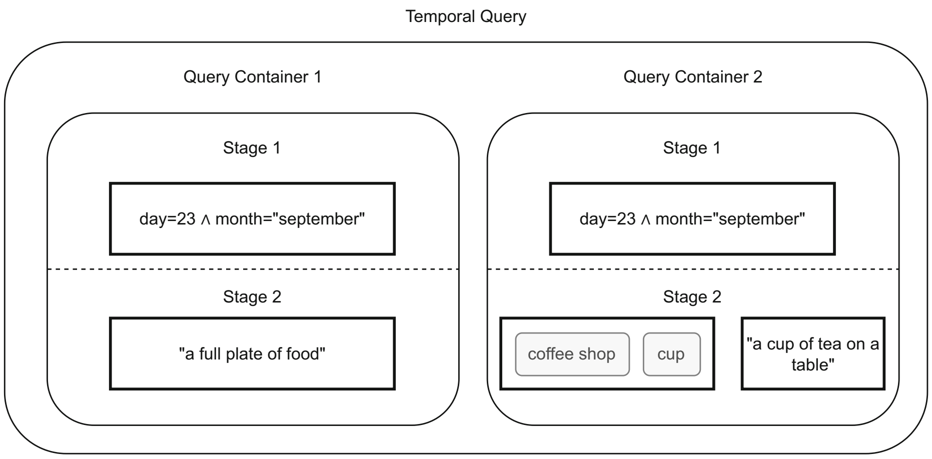


Fig. 3 The query model allows users to specify a sequence of queries. In this example, a user is looking for images from the 23rd of September, where there is first a full plate of food and then a cup of tea on a table. The second query container also has the tag modality

represents an information need for a single segment. We enable users to restrict the search space of a feature by the output of a previous one, e.g., by searching only within the images of the 23rd of September as done in both containers. We call this a *staged query* [15]. The simplest combination of modalities having them on the same level with equal importance, as done i.e., in Stage 2 of the query in Fig. 4.

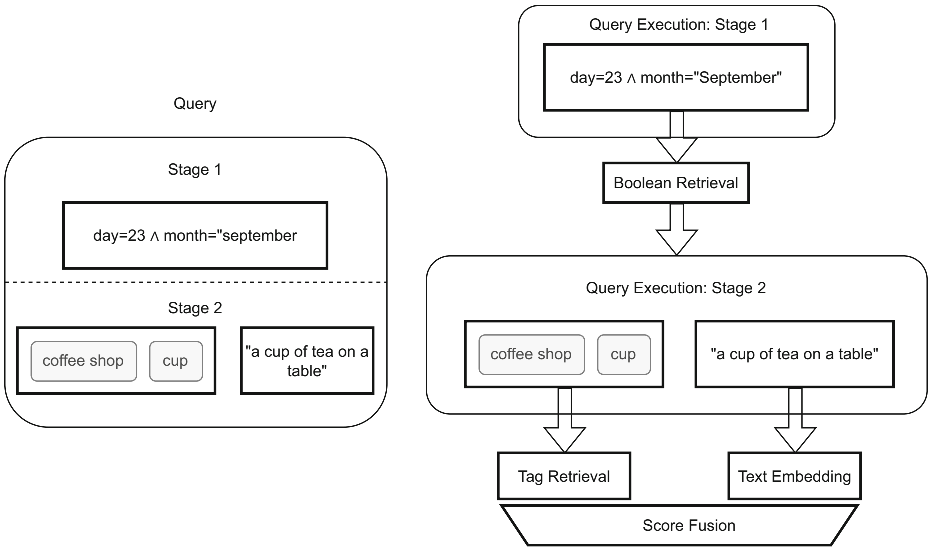


Fig. 4 Example of a staged query where the search space for the content-based features (tag and visual-text co-embedding) is limited to elements on the 23rd of September. The results of the tag and text feature are merged in the front-end to generate the final result

2.2.3 Query execution model

We will now discuss how the results of the individual features are combined given the different options from the query model, moving from simple to complex fusion. We show an example of staged query execution in Fig. 4.

Feature fusion If multiple modalities are present in one stage, we offer a configurable choice between max-pooling and average-pooling, i.e., given a list of similarity scores $(sc_1, sc_2, \ldots, sc_j)$ for a segment, either $score = \max(sc_1, sc_2, \ldots, sc_j)$ or $score = \text{avg}(sc_1, sc_2, \ldots, sc_j)$. For LSC 2021, vitrivr and vitrivr-VR have both used average-pooling. In Fig. 4, the results of the tag and visual-text co-embedding feature are fused together. Features are fused in the front-end which allows users to weigh features differently without re-executing a query.

Staged queries For lifelog retrieval queries, there are often hard binary constraints provided, such as the day of the week or a timeframe (e.g., evening). In those scenarios, it makes sense to limit the search space for similarity-based queries in advance. Our database layer supports this functionality by taking a list of allowed ids as an optional query parameter. In the example of Fig. 4, the user specifies the Boolean query for day and month as a first stage. Only the ids for this feature are returned, the features in the second stage start retrieving, as they receive the list of allowed ids in addition to the user-specified query. This necessarily takes place in the backend, but the results are still returned per feature, to allow fusion in the front-end.

Temporal queries in lifelog retrieval Starting in 2020, we have experimented with different ways to allow users to specify the temporal context of their queries, and different algorithms and architectures for these queries. This has been identified as an important component for interactive multimedia retrieval system in the analysis of interactive retrieval competitions, e.g., in VBS 2020 “The results reveal that the top two systems mostly relied on temporal queries before a correct frame was identified” [19] and 2021 “[...] almost all top performing systems [...] enable specification of temporal context in queries” [13]. Our current approach allows users to specify a *sequence of queries*, each describing a desired element. An individual query in this sequence is called *Query Container*.

The results of these individual queries are then fused together to rank objects higher which fit the specified query sequence. Our approach for video retrieval is described in detail in [10], and is adopted for lifelog retrieval with minor modifications.

Our current model has several drawbacks when applied to lifelog retrieval, such as the fact that object-level constraints such as the date have to be specified per query container. Additionally, we have yet to perform a thorough investigation into the performance of our algorithms in the context of lifelog retrieval.

3 Architecture

As mentioned previously, both vitrivr and vitrivr-VR share a common retrieval engine called Cineast [29, 30], and rely on a specialized database for multimedia and Boolean retrieval, Cottontail DB [4]. Figure 5 shows a high-level view of the architecture of the two systems.

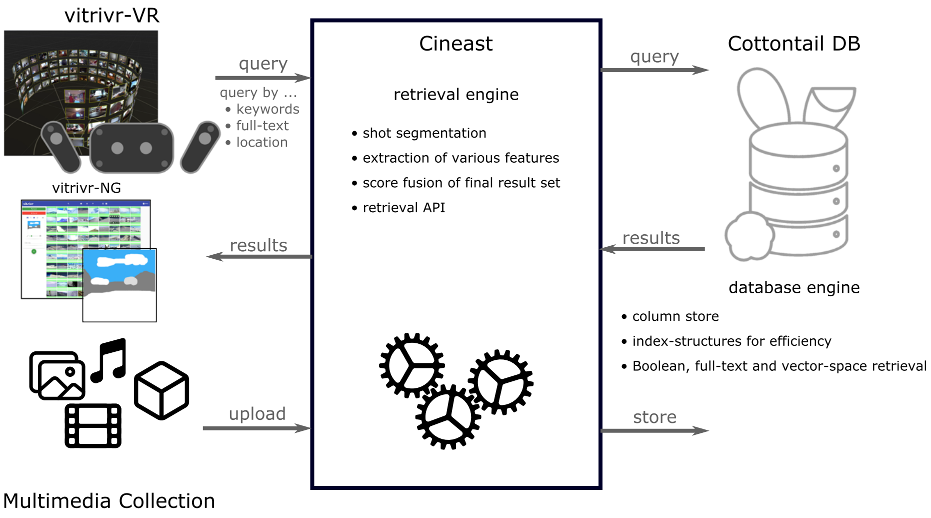


Fig. 5 Architecture of both systems, with the shared components of the retrieval engine and the database. Adopted from [37]

In this section, we will briefly discuss how the different components are related to each other the separation of concerns between the layers. This builds the foundation for Section 4, where we describe the user interfaces in detail.

Database: Cottontail DB supports vector-based similarity search, text retrieval and simple Boolean retrieval. Since it does not offer joins, metadata and segment/object information lookup is done by the retrieval engine. As discussed in previous section, all functionality is used, e.g., visual-text co-embedding uses vector-based similarity search and Boolean retrieval is used for metadata.

Retrieval Engine: Cineast is responsible for feature extraction ahead of the competition and provides a GRPC, Websocket and RESTful API for query execution.

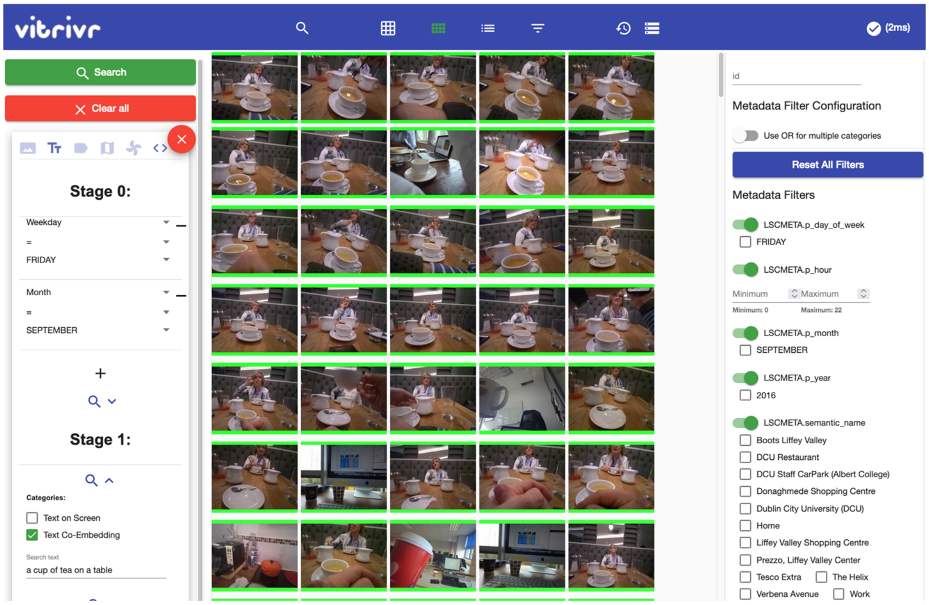
Due to technical reasons, the systems use different API methods. vitrivr uses the Websocket API of Cineast, whereas vitrivr-VR uses the RESTful API, but the underlying retrieval model is the same.

vitrivr has a web-based user interface called vitrivr-ng which is implemented in Typescript[6] and uses Angular[7]. First introduced in [6], it also has served as the basis for LSC participations by other teams [24, 25]. vitrivr-VR is implemented in Unity and uses OpenXR.

4 A tale of two interfaces

Having discussed the conceptual retrieval model which is shared between the two systems and their common architectural components, we now turn to their main defining difference: the user interface. Both user interfaces allow query formulation with different modalities and result presentation, with the main differences being that vitrivr-VR does not support late filters nor staged or temporal queries.

[6]https://www.typescriptlang.org/
[7]https://angular.io/



Stage 0:

Weekday
=
FRIDAY

Month
=
SEPTEMBER

Stage 1:

Categories:
- [ ] Text on Screen
- [x] Text Co-Embedding

Search text
a cup of tea on a table

id

Metadata Filter Configuration

Use OR for multiple categories

Reset All Filters

Metadata Filters

LSCMETA.p_day_of_week
- [ ] FRIDAY

LSCMETA.p_hour
Minimum Maximum
Minimum: 0 Maximum: 22

LSCMETA.p_month
- [ ] SEPTEMBER

LSCMETA.p_year
- [ ] 2016

LSCMETA.semantic_name
- [ ] Boots Liffey Valley
- [ ] DCU Restaurant
- [ ] DCU Staff CarPark (Albert College)
- [ ] Donaghmede Shopping Centre
- [ ] Dublin City University (DCU)
- [ ] Home
- [ ] Liffey Valley Shopping Centre
- [ ] Prezzo, Liffey Valley Center
- [ ] Tesco Extra
- [ ] The Helix
- [ ] Verbena Avenue
- [ ] Work

Fig. 6 Full overview of the vitrivr-ng user interface. On the left side, users can formulate queries. The middle part shows results, and on the right side, filters can be applied

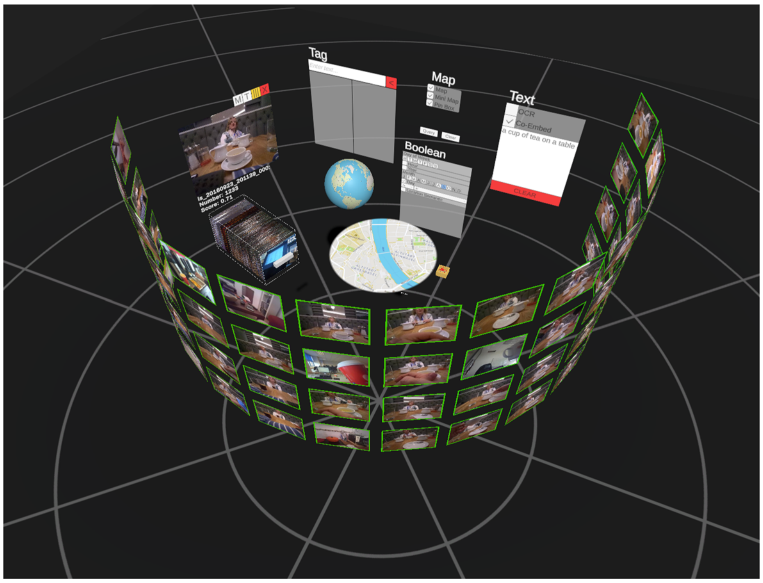
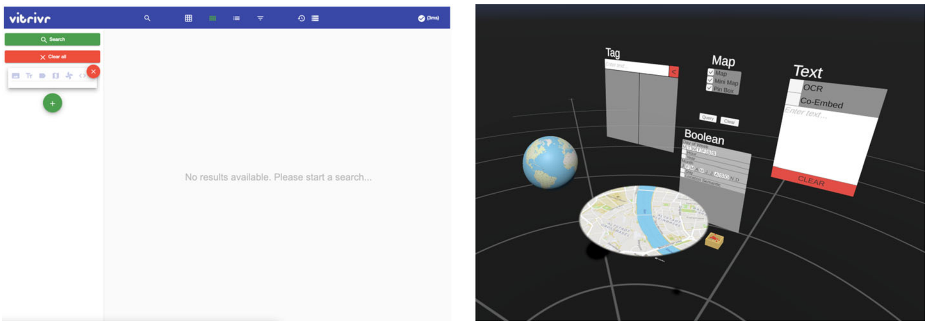


Fig. 7 Overview of vitrivr-VR showing the modular query formulation UI surrounded by the cylindrical results display and one result shown in detail view with the temporally neighboring images shown in the form of a drawer of images below it. Due to the inherent nature of made-for-VR content, it is not possible to clearly depict all aspects of vitrivr-VR in a single 2D representation



(a) vitrivr-ng (b) vitrivr-VR

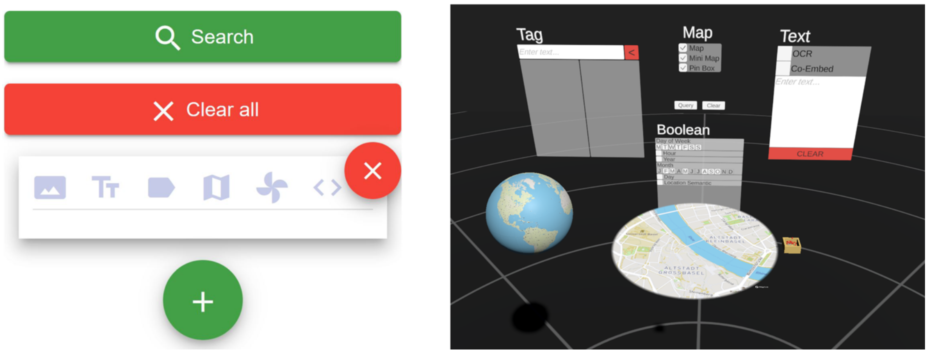
Fig. 8 Side-by-side comparison of the entrypoints of the two systems

In this section, we will cover each user interaction aspect in one of the following subsections. While sketch queries are supported by both systems, they were not used in the LSC context and therefore are not discussed here.

In Fig. 6 and 7, we show the two user interfaces in action, with a query already formulated and results visible. In vitrivr-ng, query formulation happens on the left side of the screen, and results are displayed in the center. Different result views can be toggled in the header. vitrivr-VR takes a more free-form and modular approach to the user interface in an attempt to make use of additional freedom in VR. The query formulation interface is separated into panels by modality. Each of these interfaces can be grabbed by the user and rearranged in virtual space. In vitrivr-VR, query results are displayed in a horizontally scrollable display that wraps cylindrically around the user. Figure 8 shows a comparison of the entrypoints of the two systems.

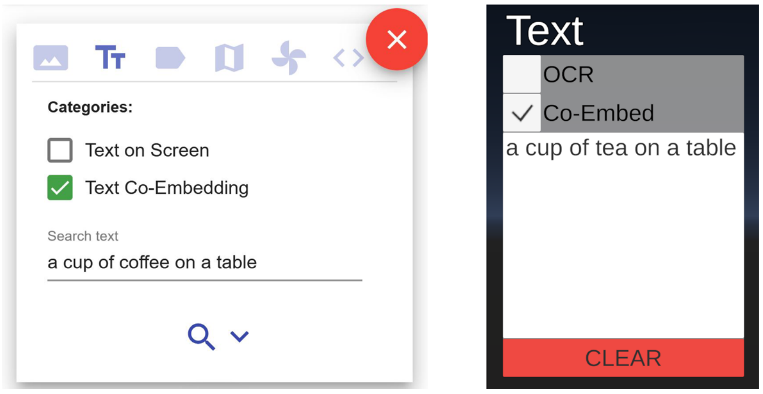
4.1 Query formulation

In Fig. 9, we show a comparison of the query formulation views. In vitrivr-ng, all modalities can be toggled. Additionally, new query containers can be added by clicking on the green plus button, which allows adding temporal context. This enables users to specify temporal



(a) vitrivr-ng (b) vitrivr-VR

Fig. 9 Side-by-side comparison of the query formulation view



(a) vitrivr-ng (b) vitrivr-VR

Fig. 10 Side-by-side comparison of the textual query formulation modality

sequences in their query. When multiple modalities are used within the same query container, they can be individually pushed to a later stage by clicking on the downward arrow next to the magnifying glass visible in Fig. 10a.

In the current version of vitrivr-VR, staged and temporal queries are not yet available, making query formulation less expressive, but also much simpler. All available query modalities are available as individual interfaces in virtual space and are automatically used for subsequent queries as soon as the user inputs data. The text term input, for example, is used for queries as long as at least one text category has been selected and the user has input text in the text field.

As highlighted in previous sections, our model for lifelog retrieval allows a seamless combination of different modalities. The query formulation for those modalities will be described in the following.

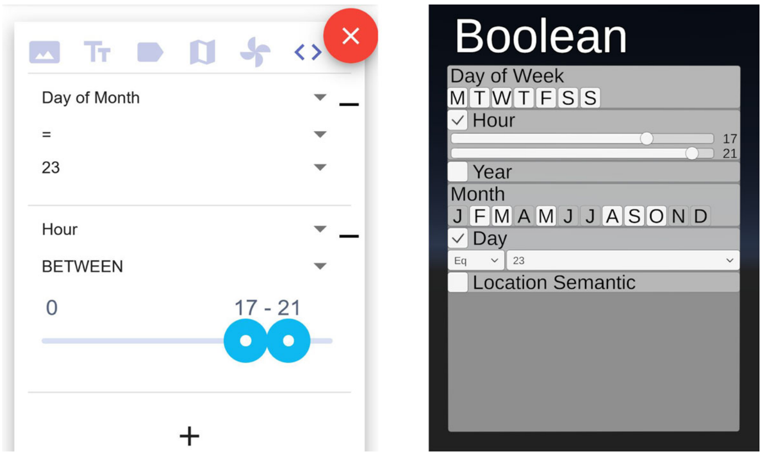
4.1.1 Textual queries

Both interfaces allow textual queries for OCR or visual-text co-embeddings, and a comparison is shown in Fig. 10. vitrivr-ng offers traditional text input, where boxes can be checked depending on the desired feature (e.g., OCR, textual embedding). vitrivr-VR offers a very similar interface in VR, where different feature categories can be toggled via a set of checkboxes attached to a text field. Text input in vitrivr-VR is facilitated through speech-to-text and a virtual keyboard. The speech-to-text solution, based on DeepSpeech[8], is particularly useful to quickly enter long scene descriptions for the visual-text co-embedding, while the virtual keyboard allows the input of hard-to-pronounce words and precise corrections.

4.1.2 Boolean queries

vitrivr-ng mainly supports two types of Boolean queries, shown in Fig. 11a: simple dropdowns, either with a provided list of options or dynamically generated based on distinct

[8]https://github.com/mozilla/DeepSpeech



(a) vitrivr-ng (b) vitrivr-VR

Fig. 11 Side-by-side comparison of the Boolean query formulation view

column values, or range queries for e.g., hour of the day. Additionally, the same query modality offers text retrieval e.g., for segment ids as a convenience feature.

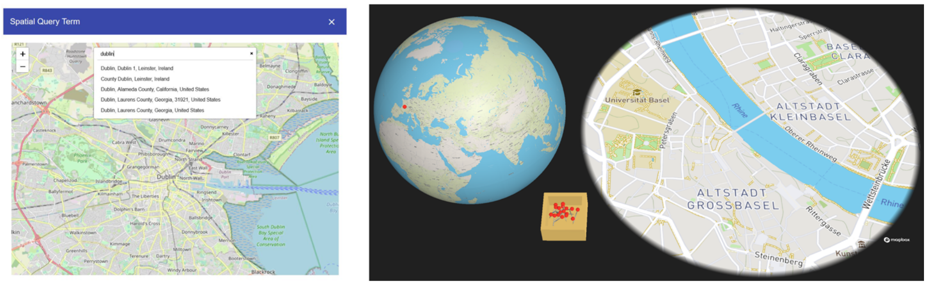
vitrivr-VR supports the same two types of Boolean queries through sliders for ranges and drop downs for single selections, but also supports multiple-response style selection e.g., to select the days of the week. This multiple-response style selection results in a Boolean expression specifying a segment's membership in the selected set of options. As seen in the month selection in Fig. 11b, multiple-response style selections indicate if data fitting a certain value even exist by disabling the options that do not represent any data.

4.1.3 Geographical queries

For information needs with a spatial context, vitrivr-ng supports the simple use case of putting a pin on the map and searching for segments in proximity. This is implemented using Leaflet[9] and OpenStreetMap[10], and the leaflet-geosearch package[11] is used for location lookup independent of the dataset, i.e., searching for "Dublin". Figure 12 shows the query formulation view of both systems.

In vitrivr-VR the geospatial query interface consist of three parts: a globe-like *mini-map*, a flat *detail-map* and a *pin-box*. Geospatial queries are formulated by grabbing a pin from the pin-box in virtual space and sticking it into the detail-map. To specify locations outside of the detail-map's default view, it can be dragged with one hand to pan the map and zoomed through a pinching gesture using both hands. To quickly change the location displayed by the detail-map without having to zoom the map out and in again, a pin can be placed on the globe mini-map, which immediately sets the detail-map to focus on the corresponding location. All parts of the geospatial query interface can be grabbed and placed anywhere in virtual space and even disabled, allowing users to customize the query interface to their individual needs.

[9]https://leafletjs.com/
[10]https://www.openstreetmap.org
[11]https://github.com/smeijer/leaflet-geosearch


Spatial Query Term
dublin
Dublin, Dublin 1, Leinster, Ireland
County Dublin, Leinster, Ireland
Dublin, Alameda County, California, United States
Dublin, Laurens County, Georgia, 31921, United States
Dublin, Laurens County, Georgia, United States
Universität Basel
ALTSTADT KLEINBASEL
ALTSTADT GROSSBASEL
Rhine

(a) vitrivr-ng (b) vitrivr-VR

Fig. 12 Side-by-side comparison of the geospatial query formulation view

4.2 Result presentation

Having formulated the queries, the retrieval engine returns results per segment and feature[12], allowing the user interface to determine the result view and final ranking.

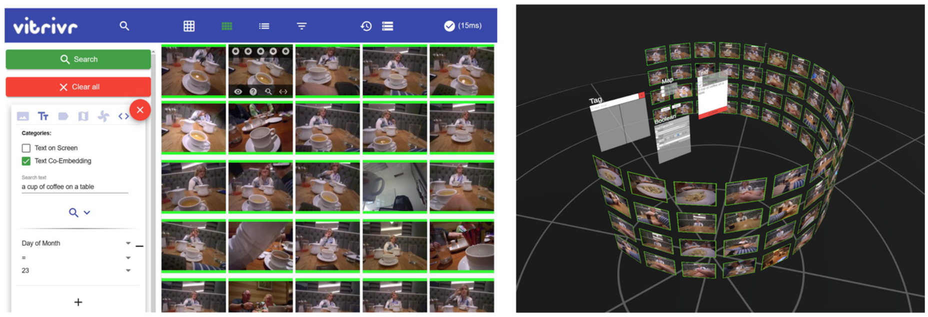
Both interfaces allow two different result presentation views: A ranked segment list, shown in Section 4.2.1, and views which groups segments together and aggregates their score, shown in Section 4.2.2. The UI for inspecting segments is shown afterwards in Section 4.2.3.

The result presentation of vitrivr-ng and vitrivr-VR differs in the details of presentation, but there are also some bigger functional differences between the two interfaces. While the result displays offered by vitrivr-ng are more interactive and allow additional configuration through late filtering, only the result set of the most recent query can be viewed. vitrivr-VR, by comparison, does not support late filtering, but preserves the results from previous queries, allowing users to return to previous result set including their scrolling position within the results. Furthermore, by allowing results to be pulled out of a results display and placed anywhere in virtual space, vitrivr-VR allows specific results to be kept independently from the query results from which they were taken, and to be used as reference or for comparison while viewing results from a different query.

4.2.1 Ranked segment list

The default view for both interfaces orders individual segments (that is, individual images in our context) by their fused score. Shown in Fig. 13, this is a classic 2D-grid in vitrivr-ng and a horizontally scrolling 2D-grid that wraps cylindrically around the user in vitrivr-VR. While additional results can be viewed in vitrivr-ng via vertical scrolling, in vitrivr-VR results scrolled beyond 360 degrees around the user are hidden from the display and replaced with new results in decreasing order of similarity score. This interaction is performed through regular scrolling in vitrivr-ng, while it is performed through the primary touchpad or joystick in vitrivr-VR.

[12]Features are grouped by the backend into so-called *categories*, and technically results are returned per category. That implementation detail is not particularly relevant, as often categories only contain a single feature. Categories can also be thought of as hierarchical features.



(a) vitrivr-ng (b) vitrivr-VR

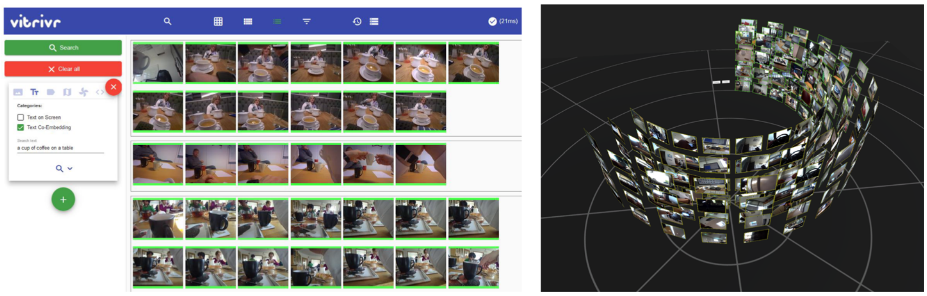
Fig. 13 Ranked segment list result display. Images from different days and events are scored individually, and ranked by score

4.2.2 Segment aggregation views

Both interfaces offer a result view which is based on object scores, which is a score for a given day in the LSC context. The two interfaces differ both in the way these object scores are calculated and how they are displayed. In vitrivr-ng, objects are scored through average-pooling of the segment scores, i.e., given all scored segments $\hat{S}_o = (\hat{s}_o^1, \hat{s}_o^2, \ldots, \hat{s}_o^n)$ for an object o, the score is determined as follows:

$$score_o = \frac{\sum_{i=1}^{n} \text{score}\left(\hat{s}_o^i\right)}{n}$$

vitrivr-VR uses the maximum score of all segments to rank objects in its segment aggregation results view. In this results view, each object (in the case of the LSC dataset, each day) is assigned a position in the cylindrical grid, ordered by the maximum score of the segments it contains. A fixed number of segments of this object is then shown in this position, one behind the other, ordered by segment score. These views are shown in Fig. 14.



(a) vitrivr-ng (b) vitrivr-VR

Fig. 14 Object-based result display in vitrivr-VR in a cylindrical grid (query formulation UI disabled for clarity) and in vitrivr-ng as a list, with object (days) separated by double grey lines

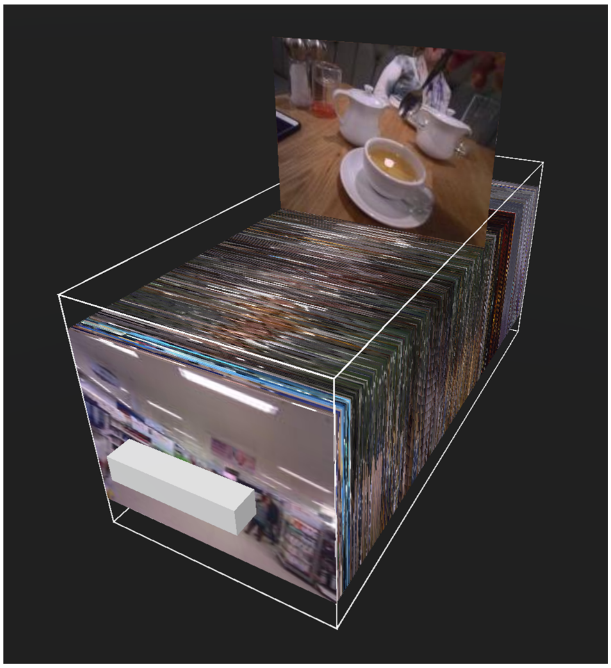


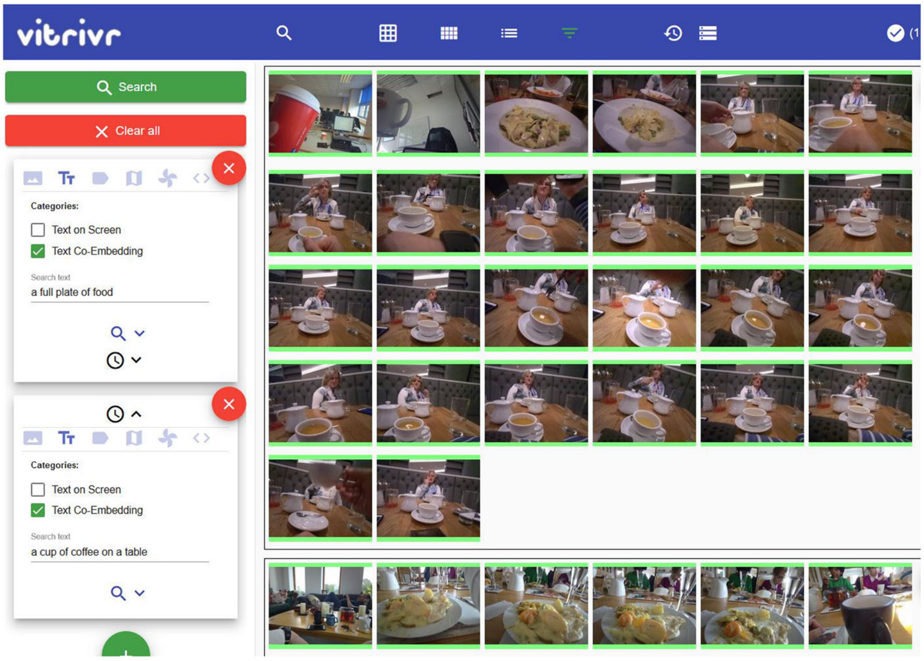
Fig. 15 The drawer view of the segments of a day in chronological order

Both vitrivr-ng and vitrivr-VR support different additional segment aggregation views. vitrivr-VR provides an additional drawer view, showing the segments of an object neighboring the selected segment in chronological order inside a virtual box, which can be moved by the user. This view is shown in Fig. 15. By moving their hand into this virtual box and hovering over a segment image, users can take a closer look at individual segments, which are then shown above the box. By quickly moving their hand through this virtual box, users can 'riffle' through the segments, producing an effect similar to that of a flip-book. The virtual box containing the segment images can be moved around. Moreover, users can also elongate the box to simplify segment selection, by grabbing and pulling the handle, similar to the action of pulling out a drawer.

In addition to its regular segment aggregation view, vitrivr-ng offers a temporal scoring view, which computes sequences of segments which match the user-specified query sequence as discussed previously. In our context, this means that for a given day, there can be multiple sequences each with their individual score. The result view shown in Fig. 16 looks similar to that for objects, which makes sense given that it can also be looked at as a different way of aggregating segment scores.

4.2.3 Segment inspection

If a user is interested in closely inspecting a segment to see whether it matches their information need, both user interfaces offer the ability to look at accompanying metadata, and vitrivr-ngalso shows extracted features. We show a side-by-side comparison of the views in Fig. 17.



Categories:

- [ ] Text on Screen
- [x] Text Co-Embedding

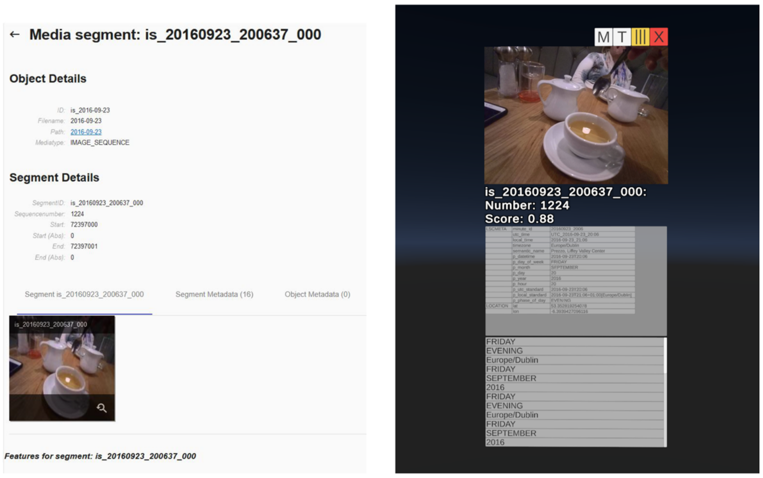
Search text
a full plate of food

Categories:

- [ ] Text on Screen
- [x] Text Co-Embedding

Search text
a cup of coffee on a table

Fig. 16 vitrivr-ng also allows specifying temporal context, and groups segments together which match the order of the query


← Media segment: is_20160923_200637_000

Object Details

ID:	is_2016-09-23
Filename:	2016-09-23
Path:	2016-09-23
Mediatype:	IMAGE_SEQUENCE

Segment Details

SegmentID:	is_20160923_200637_000
Sequencenumber:	1224
Start:	72397000
Start (Abs):	0
End:	72397001
End (Abs):	0

Features for segment: is_20160923_200637_000

is_20160923_200637_000:
Number: 1224
Score: 0.88

LSCMETA	minute_id	20160923_2006
	utc_time	UTC_2016-09-23_20:06
	local_time	2016-09-23_21:06
	timezone	Europe/Dublin
	semantic_name	Prezzo, Liffey Valley Center
	p_datetime	2016-09-23T20:06
	p_day_of_week	FRIDAY
	p_month	SEPTEMBER
	p_day	20
	p_year	2016
	p_hour	20
	p_utc_standard	2016-09-23T20:06
	p_local_standard	2016-09-23T21:06+01:00[Europe/Dublin]
	p_phase_of_day	EVENING
LOCATION	lat	53.352819254078
	lon	-6.3939427096116

FRIDAY
EVENING
Europe/Dublin
FRIDAY
SEPTEMBER
2016
FRIDAY
EVENING
Europe/Dublin
FRIDAY
SEPTEMBER
2016

(a) vitrivr-ng (b) vitrivr-VR

Fig. 17 Segment inspection which allows looking at metadata

4.2.4 Filtering of result sets

Given that in LSC, new hints appear over time, it is useful to have front-end filters which avoid reformulation and execution of a query. vitrivr-ng has added this functionality specifically for LSC. Two types of late filters are supported: checkboxes which are dynamically generated based on the available metadata, and range filters for e.g., time of day.

5 LSC 2021 result analysis

Our analysis focused on the results of LSC 2021, both due to the scope of the special issue and the fact that this was the first year vitrivr-VR participated, which allows for a comparison of the two participants. We use logs provided by the competition server DRES [28], and post-process them into an SQLite database which makes for easier analysis. The final database was provided to all teams. Our analysis is based on the methodology of the VBS 2021 analysis [13]. We first compare the relative performance of the two systems through analysis of the solved tasks, then analyze and discuss indicators of browsing efficiency, and finally provide recommendations for improving the data collection with the goal of enabling more in-depth analyses in the future. The data and results show that limited robust conclusions can be drawn, which is why in Section 5.3 we discuss recommendations for future iterations of LSC and similar interactive evaluations.

5.1 Solved tasks

Table 1 shows an overview of correctly solved tasks by the two teams. Even though both systems shared the same features and retrieval model, the operator of vitrivr solved three tasks that the operator of vitrivr-VR did not solve, but vitrivr-VR also solved one tasks which was not solved by vitrivr. Due to the similarities between system capabilities, these differences in solved tasks can be most likely attributed to operator behaviour, such as query formulation and browsing strategies. While the low number of samples we can analyze is partly because both systems together were only able to solve nine of the twenty-four tasks, LSC could benefit from having multiple independent operators for each system. Such an increase in sample size was already shown to increase the statistical robustness of interactive evaluations [27].

Table 1 Tasks which were solved by at least one of the two systems, together with the time in seconds until correct submission

Task	vitrivr	vitrivr-VR
01	168	**97**
04	**36**	60
06	**101**	282
07	**260**	–
08	**166**	–
13	**194**	–
15	94	**47**
24	294	**278**
23	–	**53**

The faster system has its time highlighted in bold

Due to the low number of data samples, it is difficult to make any conclusive statistical statements, however the data we do have indicates, that there are no clear system specific patterns with regards to task solve time. Neither system-operator combination clearly outperforms the other with regards to the time to solve a task, with only a very slight tendency of vitrivr solving more tasks than vitrivr-VR.

5.2 Browsing efficiency

Our main tool to investigate the difference between the two systems is the result logs that both systems submitted to DRES. This allows us to compare the browsing efficiency between the two systems in two key aspects: We can compare the time it takes between the appearance of a correct item in the result view until the operator submits it, and if there were items in the result views that would have been correct, but were not submitted.

To compare the two systems based on how fast result sets can be browsed and correct results can be found, we can look at two different metrics: The time delta between the *first* time the correct result appears in the results and submission time, and between the *last* time the correct result appears before submission and submission time. The first option gives us insights into cases where the correct result would have been already in the results, but the operator chose to reformulate the query, while the second option really tells us about the time it took to browse the final result set before submission. Both cases are shown in Fig. 18.

As can be seen from Table 1, vitrivr solved eight tasks, while vitrivr-VR only solved six, however, both plots in Fig. 18 only contain four data points for vitrivr-VR. What may appear like an error on first glance is actually another indicator of the deficiencies of the current data logging format. In the two cases of missing data, the correctly submitted segment never appeared as a direct result of any query formulated through vitrivr-VR and was most likely found as a result of the many result exploration options. In particular, what most likely led to the correct submission in these cases, was the use of the drawer view originating from a different segment of the same day as the correctly submitted segment. This theory is corroborated by the log of the vitrivr-VR interaction events, which show that the use of the drawer view was the last logged interaction before submission.

In Fig. 19, we show all cases where a correct item did appear in the result list, regardless of whether it was submitted or not. It is clearly visible in Fig. 19 that for both systems,

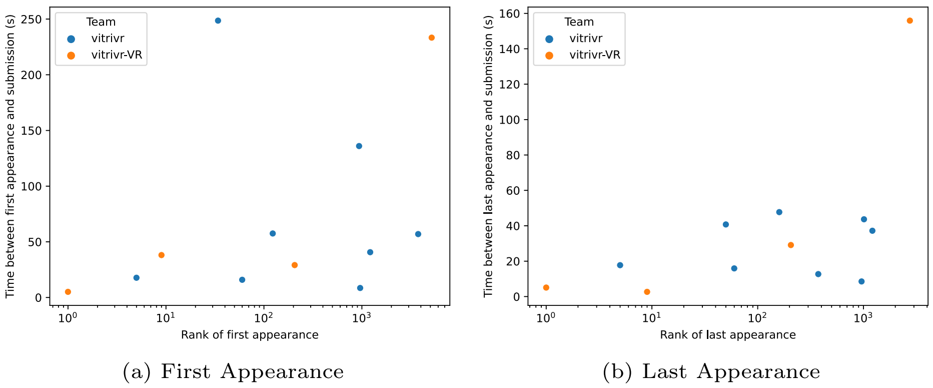
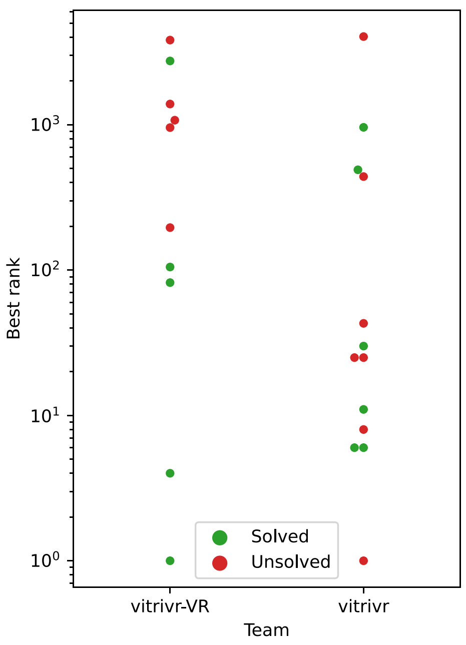


Fig. 18 Time between first or last result log appearance and correct submission compared to the rank of first or last appearance

Fig. 19 Best rank of any correct item per task and team. Red dots are operator browsing misses, i.e., the correct item would have been in the results, but did not get submitted



there were operator errors, where a correct item would have been in the results, but was not submitted. While it is somewhat expected that for images which have a poor rank, an operator might not browse that far, there are a significant amount of browsing misses for vitrivr between ranks 1 and 100, which indicates operator errors as these are items that were most likely seen during browsing. Due to the low number of data points it is difficult to make statistically robust statements, but it appears that the queries of the vitrivr operator led to results with a higher precision than those of the vitrivr-VR operator, but that correct results may be easier to spot in vitrivr-VR than in vitrivr.

5.3 Recommendations for data collection

Based on our experience and results, we highlight possible areas of improvement for the current methodology of LSC, both conceptually and for the current infrastructure.

Multiple independent participants As previously mentioned, the current format for the LSC means that we only have data from one system instance available. Having more instances of the same system participating independently would allow us to perform different analyses, such as comparing inter-operator and inter-system differences as done in [27], and increase confidence and robustness in the results while using the same methodologies as done here.

One operator per system instance The fact that the rules allow for multiple participants to be using the instance collaboratively, as long as there is only one operator means that some systems could have five people searching on the same screen and brainstorming for queries, while others could be using a single operator. This puts VR systems at a disadvantage, since they are more challenging to be used cooperatively, but also makes results analysis more challenging since there is an additional variable to be considered.

Data format The current data format provided by the evaluation server only allows data analysis beyond that already provided as statistics calculated by the server after significant data integration effort. As of the version used at LSC,[13] the evaluation server provides (i) a JSON file containing the most important information about an interactive retrieval competition event, such as the competing teams, the tasks that were used during the competition, and the submissions per task, and (ii) a CSV file containing a number of precomputed statistics resolved by team and task, such as the time until a correct submission was made.

While the evaluation server supports the logging of query results and interaction events, which was encouraged but not required at the LSC'21, this data is not contained within the standard set of analysis data provided by the server. This information is only accessible in the form of a complete log of all server events containing query results logs, interaction logs, submission logs, and task start and end events.

As this data, with the exception of submission logs, is only resolved by timestamp and session ID, a number of reconciliatory steps are required to connect it with the rest of the data and allow analysis to be performed. Two additional files are needed from the evaluation server maintainers: 1. the audit log file containing session logins resolved by username and timestamp, and 2. a file manually compiled by the evaluation server maintainers containing the mapping between usernames and user IDs. To import the data into a common schema session IDs must be resolved to usernames by matching login and logout time windows, usernames must be resolved to user IDs, query result and interaction logs must be resolved to tasks and users by matching task time windows and session IDs, and items in query result logs, which may be in any format, must be converted to the same format and matched with task targets. This reconciliation effort is additionally complicated through the mixed use of unlabeled server and client timestamps.

Based on these insights on the analysis data format, we make the following recommendations for future data collection: 1. specify and enforce a unified format both for data collected from participating systems, such as query result items, and for data logged by the server, such as timestamps, 2. link data appropriately on server side to ensure a consistent interpretation of analysis data across all individual team analyses, 3. provide all data in an appropriate format, such as CSV files or a relational database, and 4. document data sufficiently to prevent ambiguities and misinterpretations during analysis.

Collected data At the current time, the data collected from different retrieval systems is sufficient only for very limited analyses. To be able to thoroughly analyze the performance of different systems and identify the highest performing user strategies, more data must be collected to reveal the user interactions and data pathways that led to successful submissions. With the current data collection strategy, which only specifies the collection of query result, basic interaction, and submission data, it is often impossible to determine which query led to a submission and what interactions were involved.

We make the following recommendations for data collection to enable more in-depth analysis in the future: 1. specify a unified format for the different interaction methods that can lead to a submission to allow the quantitative analysis of this data without post-hoc reconciliation or interpretation of logs, and 2. expand and formalize the data collection to ensure the necessary data required to trace the submission back to the first relevant interaction is collected for all systems

[13] v1.0.1, https://github.com/dres-dev/DRES/releases/tag/1.0.1

We are aware that these recommendations require significant research effort to adequately implement, but we believe that this is could greatly improve the depth of analysis possible for interactive retrieval competition data.

6 Conclusion and future work

In this paper, we have compared and contrasted the retrieval and interaction approaches of two participating systems at LSC, vitrivr and vitrivr-VR. In addition to a careful description of the conceptual underpinnings of the systems, the different user interfaces have been described, with a special focus on the affordances of VR.

Our analysis of the result logs shows that, while the two systems perform similarly, with vitrivr performing slightly better, both systems have different strengths and weaknesses. The analysis clearly shows the need for improved data collection, in terms of the kind of data, the quantity, and the format. Multiple independent users per system, collection of data clearly linking submissions to operator interactions and queries, and the specification of a clear, formally defined schema for interaction and results logging would enable a vastly more robust analysis and would allow insights into the performance of systems and operators that are currently impossible to obtain.

LSC offers a great platform to gain insights through a holistic evaluation of end-to-end systems with users posing interactive queries in real-time. Participating with a VR system and a conventional retrieval system allows us to compare the opportunities and challenges of VR in a fair manner, as the shared retrieval backend eliminates the retrieval model from the comparison.

For future iterations of LSC, we plan to continue on our journey towards a general-purpose multimedia retrieval engine. Newer releases of the retrieval engine include, among others, a newer text-embedding model [37], new state-of-the-art visual-text co-embedding features such as CLIP [21], and a new OCR feature module [41].

One interesting avenue of research would be lifelog summarization, similar to approaches in video retrieval [2]. Especially in lifelogs, there is a lot of overlap between consecutive images, which means visually grouping similar shots or exploring different ways to ensure users can focus on browsing meaningfully different images and events.

Acknowledgements This work was partly supported by the Swiss National Science Foundation (project "Participatory Knowledge Practices in Analog and Digital Image Archives", contract no. CRSII5_193788).

The authors would like to thank all contributors to the 2019, 2020 and 2021 LSC participations of vitrivr and vitrivr-VR for their work which has enabled this paper. Additionally, we would like to thank Ralph Gasser, Ashery Mbilinyi, Loris Sauter, and Marco Vogt for their constructive input during writing.

Funding Open access funding provided by University of Basel

Statements and Declarations The code for the systems described in this paper is available open source at https://github.com/vitrivr/.

The logs analysed in this paper are available from corresponding authors or LSC organizers on reasonable request.

The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Ang W-H, Yen A-Z, Chu T-T, Huang H-H, Chen H-H (2021) LifeConcept: an interactive approach for multimodal lifelog retrieval through concept recommendation. In: 4th annual on lifelog search challenge. Association for Computing Machinery, New York, pp 47–51. https://doi.org/10.1145/3463948.3469070
2. Barnes C, Goldman DB, Shechtman E, Finkelstein A. (2010) Video tapestries with continuous temporal zoom. ACM Trans Graph 29(4):89–1899. https://doi.org/10.1145/1778765.1778826
3. Cer D, Yang Y, Kong S-Y, Hua N, Limtiaco N, John RS, Constant N, Guajardo-Cespedes M, Yuan S, Tar C, Sung Y-H, Strope B, Kurzweil R (2018) Universal sentence encoder. arXiv:1803.11175
4. Gasser R, Rossetto L, Heller S, Schuldt H (2020) Cottontail DB: an open source database system for multimedia retrieval and analysis. In: Chen CW, Cucchiara R, Hua X-S, Qi G-J, Ricci E, Zhang Z, Zimmermann R (eds) International conference on multimedia (MM). Association for Computing Machinery, New York, pp 4465–4468. https://doi.org/10.1145/3394171.3414538
5. Gasser R, Rossetto L, Schuldt H (2019) Multimodal multimedia retrieval with vitrivr. In: International conference on multimedia retrieval (ICMR). Association for Computing Machinery, New York, pp 391–394. https://doi.org/10.1145/3323873.3326921
6. Gasser R, Rossetto L, Schuldt H (2019) Towards an all-purpose content-based multimedia information retrieval system. arXiv:1902.03878
7. Giangreco I (2018) Database support for large-scale multimedia retrieval. Thesis, University of Basel. https://doi.org/10.5451/unibas-006827345
8. Gurrin C, Schoeffmann K, Joho H, Dang-Nguyen D-T, Riegler M, Piras L (2018) Proceedings of the 2018 ACM workshop, on the lifelog search challenge. Association for Computing Machinery, New York
9. Gurrin C, Schoeffmann K, Joho H, Leibetseder A, Zhou L, Duane A, Dang-Nguyen D-T, Riegler M, Piras L, Tran M-T, Lokoč J, Hürst W (2019) [Invited papers] comparing approaches to interactive lifelog search at the lifelog search challenge (LSC2018). ITE Transactions on Media Technology and Applications 7(2):46–59. https://doi.org/10.3169/mta.7.46
10. Heller S, Arnold R, Gasser R, Gsteiger V, Parian-Scherb M, Rossetto L, Sauter L, Spiess F, Schuldt H (2022) Multi-modal interactive video retrieval with temporal queries. In: Jónsson BÞ, Gurrin C, Tran M-T, Dang-Nguyen D-T, Hu AM-C, Huynh Thi Thanh B, Huet B (eds) MultiMedia modeling. Springer International Publishing, Cham, pp 493–498. https://doi.org/10.1007/978-3-030-98355-0_44
11. Heller S, Gasser R, Illi C, Pasquinelli M, Sauter L, Spiess F, Schuldt H (2021) Towards explainable interactive multi-modal video retrieval with vitrivr. In: Lokoč J, Skopal T, Schoeffmann K, Mezaris V, Li X, Vrochidis S, Patras I (eds) MultiMedia modeling. Springer International Publishing, Cham, pp 435–440. https://doi.org/10.1007/978-3-030-67835-7_41
12. Heller S, Gasser R, Parian-Scherb M, Popovic S, Rossetto L, Sauter L, Spiess F, Schuldt H (2021) Interactive multimodal lifelog retrieval with vitrivr at LSC 2021. In: Gurrin C, Schoeffmann K, Jónsson BÞ, Dang-Nguyen D-T, Lokoc J, Tran M-T, Hürst W, Rossetto L, Healy G (eds) Workshop on lifelog search challenge. Association for Computing Machinery, New York, pp 35–39. https://doi.org/10.1145/3463948.3469062
13. Heller S, Gsteiger V, Bailer W, Gurrin C, Jónsson BÞ, Lokoč J, Leibetseder A, Mejzlí k F, Peška L, Rossetto L, Schall K, Schoeffmann K, Schuldt H, Spiess F, Tran L-D, Vadicamo L, Veselý P, Vrochidis S, Wu J (2022) Interactive video retrieval evaluation at a distance: Comparing sixteen interactive video search systems in a remote setting at the 10th Video Browser Showdown. Int J Multimed Inf Retri 11(1):1–18. https://doi.org/10.1007/s13735-021-00225-2
14. Heller S, Parian MA, Gasser R, Sauter L, Schuldt H (2020) Interactive lifelog retrieval with vitrivr. In: Gurrin C, Schöffmann K, Jónsson BÞ, Dang-Nguyen D-T, Lokoc J, Tran M-T, Hürst W (eds) Third annual workshop on lifelog search challenge. Association for Computing Machinery, New York, pp 1–6. https://doi.org/10.1145/3379172.3391715
15. Heller S, Sauter L, Schuldt H, Rossetto L (2020) Multi-stage queries and temporal scoring in vitrivr. In: IEEE international conference on multimedia expo workshops (ICMEW). IEEE, New Jersey, pp 1–5. https://doi.org/10.1109/ICMEW46912.2020.9105954
16. Li Y, Song Y, Cao L, Tetreault J, Goldberg L, Jaimes A, Luo J (2016) TGIF: a new dataset and benchmark on animated gif description. In: IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 4641–4650. https://doi.org/10.1109/CVPR.2016.502

17. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft COCO: common objects in context. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (eds) Computer Vision – ECCV 2014, vol 8693. Springer International Publishing, Cham, pp 740–755. https://doi.org/10.1007/978-3-319-10602-1_48
18. Lokoč J, Bailer W, Barthel KU, Gurrin C, Heller S, Jónsson BÞ, Peška L, Rossetto L, Schoeffmann K, Vadicamo L, Vrochidis S, Wu J (2022) A task category space for user-centric comparative multimedia search evaluations. In: Jónsson BÞ, Gurrin C, Tran M-T, Dang-Nguyen D-T, Hu AM-C, Huynh Thi Thanh B, Huet B (eds) MultiMedia modeling. Springer International Publishing, Cham, pp 193–204. https://doi.org/10.1007/978-3-030-98358-1_16
19. Lokoč J, Veselý P, Mejzlík F, Kovalčík G, Souček T, Rossetto L, Schoeffmann K, Bailer W, Gurrin C, Sauter L, Song J, Vrochidis S, Wu J, Jónsson BÞ (2021) Is the reign of interactive search eternal? Findings from the video browser showdown 2020. ACM Trans Multimed Comput Commun Appl 17(3):91–19126. https://doi.org/10.1145/3445031
20. Peterhans S, Sauter L, Spiess F, Schuldt H (2022) Automatic generation of coherent image galleries in virtual reality. In: Linking theory and practice of digital libraries. Springer International Publishing, Cham, pp 282–288. https://doi.org/10.1007/978-3-031-16802-4_23
21. Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, Krueger G, Sutskever I (2021) Learning transferable visual models from natural language supervision. arXiv:2103.00020
22. Rettig L, Shabani S, Sauter L, Cudré-Mauroux P, Sokhn M, Schuldt H (2021) City-stories: combining entity linking multimedia retrieval, and crowdsourcing to make historical data accessible. In: Brambilla M, Chbeir R, Frasincar F, Manolescu I (eds) Web engineering. Springer International Publishing, Cham, pp 521–524. https://doi.org/10.1007/978-3-030-74296-6_43
23. Rossetto L (2018) Multi-modal video retrieval. Thesis, University of Basel. https://doi.org/10.5451/unibas-006859522
24. Rossetto L, Baumgartner M, Ashena N, Ruosch F, Pernischová R, Bernstein A (2020) LifeGraph: a knowledge graph for lifelogs. In: Proceedings of the third annual workshop on lifelog search challenge. Association for Computing Machinery, New York, pp 13–17
25. Rossetto L, Baumgartner M, Gasser R, Heitz L, Wang R, Bernstein A (2021) Exploring graph-querying approaches in lifegraph. In: Workshop on lifelog search challenge. Association for Computing Machinery, New York, pp 7–10. https://doi.org/10.1145/3463948.3469068
26. Rossetto L, Gasser R, Heller S, Parian MA, Schuldt H (2019) Retrieval of structured and unstructured data with vitrivr. In: Gurrin C, Schöffmann K, Joho H, Dang-Nguyen D-T, Riegler M, Piras L (eds) Workshop on lifelog search challenge. Association for Computing Machinery, New York, pp 27–31. https://doi.org/10.1145/3326460.3329160
27. Rossetto L, Gasser R, Heller S, Parian-Scherb M, Sauter L, Spiess F, Schuldt H, Peška L, Souček T, Kratochvíl M, Mejzlík F, Veselý P, Lokoč J (2021) On the user-centric comparative remote evaluation of interactive video search systems. IEEE MultiMedia 28(4):18–28. https://doi.org/10.1109/MMUL.2021.3066779
28. Rossetto L, Gasser R, Sauter L, Bernstein A, Schuldt H (2021) A system for interactive multimedia retrieval evaluations. In: Lokoč J, Skopal T, Schoeffmann K, Mezaris V, Li X, Vrochidis S, Patras I (eds) MultiMedia modeling. Springer International Publishing, Cham, pp 385–390. https://doi.org/10.1007/978-3-030-67835-7_33
29. Rossetto L, Giangreco I, Heller S, Tanase C, Schuldt H (2016) Searching in video collections using sketches and sample images - the cineast system. In: Tian Q, Sebe N, Qi G-J, Huet B, Hong R, Liu X (eds) MultiMedia modeling, vol 9517. Springer International Publishing, Cham, pp 336–341. https://doi.org/10.1007/978-3-319-27674-8_30
30. Rossetto L, Giangreco I, Schuldt H (2014) Cineast: a multi-feature sketch-based video retrieval engine. In: IEEE international symposium on multimedia (ISM). IEEE, New Jersey, pp 18–23. https://doi.org/10.1109/ISM.2014.38
31. Rossetto L, Giangreco I, Tanase C, Schuldt H (2016) Vitrivr: a flexible retrieval stack supporting multiple query modes for searching in multimedia collections. In: International conference on multimedia (MM). Association for Computing Machinery, New York, pp 1183–1186. https://doi.org/10.1145/2964284.2973797
32. Rossetto L, Parian MA, Gasser R, Giangreco I, Heller S, Schuldt H (2019) Deep learning-based concept detection in vitrivr. In: Kompatsiaris I, Huet B, Mezaris V, Gurrin C, Cheng W-H, Vrochidis S (eds) MultiMedia modeling, vol 11296. Springer International Publishing, Cham, pp 616–621. https://doi.org/10.1007/978-3-030-05716-9_55
33. Sauter L, Gasser R, Bernstein A, Schuldt H, Rossetto L (2022) An asynchronous scheme for the distributed evaluation of interactive multimedia retrieval. In: International workshop on

interactive multimedia retrieval. Association for Computing Machinery, New York, pp 33–39. https://doi.org/10.1145/3552467.3554797

34. Sauter L, Rossetto L, Schuldt H (2018) Exploring cultural heritage in augmented reality with GoFind! In: IEEE international conference on artificial intelligence and virtual reality (AIVR). IEEE, New Jersey, pp 187–188. https://doi.org/10.1109/AIVR.2018.00041
35. Shi B, Bai X, Yao C. (2017) An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Trans Pattern Anal Mach Intell 39(11):2298–2304. https://doi.org/10.1109/TPAMI.2016.2646371
36. Sidorov O, Hu R, Rohrbach M, Singh A (2020) TextCaps: a dataset for image captioning with reading comprehension. In: Vedaldi A, Bischof H, Brox T, Frahm J-M (eds) Computer vision – ECCV 2020, vol 12347. Springer International Publishing, Cham, pp 742–758. https://doi.org/10.1007/978-3-030-58536-5_44
37. Spiess F, Gasser R, Heller S, Parian-Scherb M, Rossetto L, Sauter L, Schuldt H (2022) Multi-modal video retrieval in virtual reality with vitrivr-VR. In: Jónsson BÞ, Gurrin C, Tran M-T, Dang-Nguyen D-T, Hu AM-C, Huynh Thi Thanh B, Huet B (eds) MultiMedia modeling. Springer International Publishing, Cham, pp 499–504. https://doi.org/10.1007/978-3-030-98355-0_45
38. Spiess F, Gasser R, Heller S, Rossetto L, Sauter L, van Zanten M, Schuldt H (2021) Exploring intuitive lifelog retrieval and interaction modes in virtual reality with vitrivr-vr. In: Gurrin C, Schoeffmann K, Jónsson BÞ, Dang-Nguyen D-T, Lokoc J, Tran M-T, Hürst W, Rossetto L, Healy G (eds) Workshop on lifelog search challenge. Association for Computing Machinery, New York, pp 17–22. https://doi.org/10.1145/3463948.3469061
39. Spolaôr N, Lee HD, Takaki WSR, Ensina LA, Parmezan ARS, Oliva JT, Coy CSR, Wu FC (2021) A video indexing and retrieval computational prototype based on transcribed speech. Multimed Tools Appl 80(25):33971–34017. https://doi.org/10.1007/s11042-021-11401-1
40. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA (2017) Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-first AAAI conference, on artificial intelligence
41. Theus A, Rossetto L, Bernstein A (2022) HyText – a scene-text extraction method for video retrieval. In: Jónsson BÞ, Gurrin C, Tran M-T, Dang-Nguyen D-T, Hu AM-C, Huynh Thi Thanh B, Huet B (eds) MultiMedia modeling. Springer International Publishing, Cham, pp 182–193. https://doi.org/10.1007/978-3-030-98355-0_16
42. Tran D, Bourdev L, Fergus R, Torresani L, Paluri M (2015) Learning spatiotemporal features with 3d convolutional networks. In: IEEE international conference on computer vision, pp 4489–4497
43. Wang X, Wu J, Chen J, Li L, Wang Y-F, Wang WY (2019) VaTeX: a large-scale high-quality multilingual dataset for video-and-language research. In: IEEE/CVF international conference on computer vision (ICCV). IEEE, pp 4580–4590. https://doi.org/10.1109/ICCV.2019.00468
44. Xu J, Mei T, Yao T, Rui Y (2016) MSR-VTT: a large video description dataset for bridging video and language. In: IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 5288–5296. https://doi.org/10.1109/CVPR.2016.571
45. Young P, Lai A, Hodosh M, Hockenmaier J (2014) From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. Trans Assoc Comput Linguist 2:67–78. https://doi.org/10.1162/tacl_a_00166
46. Zhou X, Yao C, Wen H, Wang Y, Zhou S, He W, Liang J (2017) EAST: an efficient and accurate scene text detector. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5551–5560