Unified Multimedia Segmentation – A Comprehensive Model for URI-based Media Segment Representation

Jan Willi
University of Zurich, Switzerland

Abraham Bernstein
University of Zurich, Switzerland

Luca Rossetto[1]
University of Zurich, Switzerland
Dublin City University, Ireland

Abstract

In multimedia annotation, referencing specific segments of a document is often desired due to its richness and multimodality, but no universal representation for such references exists. This significantly hampers the usage of multimedia content in knowledge graphs, as it is modeled as one large atomic information container. Unstructured data – such as text, audio, images, and video – can commonly be decomposed into its constituent parts, as such documents rarely contain only one semantic concept. Hence, it is reasonable to assume that these advances will make it possible to decompose these previous atomic components into logical segments. To be processable by the knowledge graph stack, however, one needs to break the atomic nature of multimedia content, providing a mechanism to address media segments.

This paper proposes a Unified Segmentation Model capable of depicting arbitrary segmentations on any media document type. The work begins with a formal analysis of multimedia and segmentation, exploring segmentation operations and how to describe them. Building on this analysis, it then develops a practical scheme for expressing segmentation in Uniform Resource Identifiers (URIs). Given that this approach makes segments of multimedia content referencable, it breaks their atomic nature and makes them first-class citizens within knowledge graphs. The proposed model is implemented as a proof of concept in the MediaGraph Store, a multimedia knowledge graph storage and querying engine.

2012 ACM Subject Classification Information systems → Document representation; Information systems → Web data description languages; Computing methodologies → Knowledge representation and reasoning; Information systems → Multimedia information systems; Information systems → Multimedia and multimodal retrieval

Keywords and phrases Multimodal Knowledge Graphs, Multimedia Segmentation, Multimedia Representation

Digital Object Identifier 10.4230/TGDK.2.3.1

Supplementary Material The source code for MeGraS is published on Github under an MIT license. The test suite is published on Github under a Creative Commons Attribution license.
Software: `https://github.com/lucaro/MeGraS` [63]
archived at `swh:1:dir:d2dfb84bd390d27ad96190f973dfa484b9e68bc0`
Audiovisual: `https://github.com/lucaro/Unified-Media-Segmentation-Test-Suite` [62]

Funding This work was partly supported by the Swiss National Science Foundation through project "MediaGraph" (contract no. 202125).

Received 2024-03-03 **Accepted** 2024-09-30 **Published** 2024-12-18

[1] Corresponding Author

1 Introduction

Be it social media platforms, streaming services, news outlets, or video games, multimedia content is ubiquitous and takes up a big part of today's devices' bandwidth. Multimedia content, in comparison to pure textual content, carries vast amounts of knowledge whose capture is far from trivial. This is due to both the multimodality of multimedia as well as the interconnectedness of the individual pieces of knowledge captured by a media object. A video clip, for instance, not only includes factual information about elements of the scene, like objects, humans, or dialog, but also contains relations between these elements and implicitly conveys sentiments and emotions. To capture these knowledge elements and their relations, two ingredients are needed: First, individual elements of a multimedia object, henceforth called segments, need to be unambiguously referenceable. Such segments can be multi-dimensional and of different modalities, such as temporal, spatial, or some combination thereof. Second, a structural representation of interconnected knowledge must be able to capture semantic annotations of segments and relations between them. For knowledge that can be easily represented in textual form, this is frequently done using knowledge graphs, which are graph structures where nodes contain facts and edges establish relations. Multimedia content, however, exceeds text in richness and is thus unsuitable to be expressed in traditional knowledge graphs.

In literature, this problem has been approached from different perspectives. The rise of multimedia content on the Web led to numerous file standards, like SVG or MPEG-7, which describe metadata of multimedia and can annotate specific regions of it. Following that, efforts were made to formalize these standards using Semantic Web technologies to improve the processing of annotations by computers. This led to a number of proposed ontologies, such as the COMM Ontology [2]. At the same time, another stream of research focused on leveraging Uniform Resource Identifiers (URIs), a core concept of the Semantic Web, to uniquely describe not only complete media objects but also segments of it. Such URIs can then be used in Resource Description Framework (RDF) triples to annotate (or other knowledge graph representations cf. [33]) and relate multimedia segments. A notable mention of this is the Media Fragments URI 1.0 specification [75], which establishes a syntax for accessing fragments of a media object. Similar concepts can also be found in other sectors, like the International Image Interoperability Framework (IIIF) [71], an initiative for standards for image delivery in cultural heritage institutions, which features a URI scheme to access full or partial images. In all of this literature, work was focused on specific use cases and applications with only a limited number of segmentation and media document types. The only exception is the General Fragment Model by [23], which attempts to generalize the notion of media fragments in an ontological way. It is, however, a primarily abstract model, and no practical implementation exists.

In this paper, we present an RDF-compatible approach for the unified representation of multimedia segmentations with the aim of making multimedia first-class citizens in knowledge graphs. The goal of this paper is to tackle segmentation and its representation equally in a more holistic way. To that end, a Unified Segmentation Model is defined, which aims to be universal enough to describe all possible segments of all possible multimedia documents and yet be practical enough to be implemented into a real-world application.

Hence, the contributions of this paper are three-fold:

1. We discuss the properties of media segmentation, put them onto a solid formal foundation, independent of specific media types, and introduce our Unified Segmentation Model based on that foundation.
2. We present an RDF-compatible URI-based approach for the unified representation of arbitrary segments of multimedia content that leverages the insights of the Unified Segmentation Model.
3. We illustrate the applicability of the approach in a prototypical implementation.

These contributions provide the foundation to break the atomicity of multimedia content in a principled manner allowing multimedia *segments* to become first-class citizens in the knowledge graph domain.

The remainder of this paper is structured as follows: Section 2 discusses how literature currently annotates the created segments to close the gap between data and knowledge. In Section 3, the segmentation model is formally established and reasoned, and resulting properties, characteristics, and classes are discussed. Sections 4 and 5 then bridge the gap to the concrete implementation in Section 6 by translating the theoretical principles into applicable URI schemes and discussing their use in knowledge graph applications, respectively. A discussion follows in Section 7, after which Section 8 concludes the paper.

2 Segment Annotation

In multimedia content analysis, semantic knowledge is crucial for efficient manipulation and retrieval of media [9]. In many cases, though, semantic annotations may not target the whole media document but refer to specific segments. Therefore, to effectively annotate multimedia content, annotations need to be able to refer to any possible segment of that content. To this end, literature and practice have proposed and established various file types, standards, recommendations, and frameworks. The many existing approaches can be distinguished into two classes. Media documents can be annotated with segment information in metadata documents. Later, with the rise of the Semantic Web, work was put into ensuring the interoperability of applications that produce such multimedia annotations. Segments can be captured in the Uniform Resource Identifier (URI) that specifies the resource. Apart from *segment*, the term *fragment* is often found in literature. In the following, the two are used interchangeably.

This section discusses these two approaches, followed by a foray into the more generic General Fragment Model and a discussion of the insights gathered by reviewing these related works.

2.1 Metadata-based Annotation

This first category consists of approaches that store segment information in a file separate from the original multimedia resource. While these typically XML-based files allow for conveying extensive segments, an indirection is required. The semantic description of a segment refers to the definition of the segment, and not the segment itself [74, 52]. In other words, segmentation is defined by a part of a description document, which in turn represents the multimedia document.

2.1.1 Synchronized Multimedia Integration Language

The Synchronized Multimedia Integration Language (SMIL) is an XML-based language to create interactive multimedia presentations [34]. Users can include multimedia, like audio, video, and text, and define timings, transitions, and animations. For that, SMIL supports both temporal and spatial fragments. For the former, the user can specify the beginning and end of the desired snippet of a longer video or audio object. For the latter, SMIL allows cropping rectangles from images or videos. Additionally, transitions of the rectangle can be defined, which enables the creation of a primitive spatio-temporal fragment.

2.1.2 Scalable Vector Graphics

The Scalable Vector Graphics (SVG) format, also based on XML, is used to describe two-dimensional vector and raster images [40]. Besides rectangles, arbitrary paths can be specified, allowing for any fragment shapes. Additionally, with `svgView`, fragments can be established directly in the URI of an SVG object, although only rectangular regions are supported. Temporal segments are not naturally supported, but SVG allows adding HTML or SMIL as foreign objects, which themselves could contain content with a temporal component.

2.1.3 MPEG-7

The MPEG-7 standard is created by the Moving Picture Experts Group (MPEG) for describing multimedia content of many different types [37]. Unlike previous MPEG standards, which focus on storing content, MPEG-7 provides means to store information about the content. MPEG-7 defines four building blocks [3, 65, 41]:

- *Descriptors* describe syntax and semantics of a feature that characterizes the content.
- *Description schemes* describe the structure of relationships between components (descriptors or description schemes).
- The *description definition language* allows creating descriptors and description schemes as well as extension and modification of existing description schemes.
- *Syntax tools* are used for binarization, synchronization, transport, and storage of descriptions.

The `Segment` description schema can be used to annotate temporal, spatial, and spatio-temporal fragments using XML-based metadata [65]. Temporal fragments are specified using starting time and duration. Spatial regions are created from polygon coordinates, allowing for arbitrary shapes. It is also possible to construct a segment from multiple components. On top of that, time frames and regions can be combined into spatio-temporal fragments.

2.1.4 Ontological Segment Description

[7] proposed the Semantic Web as an extension to the World Wide Web. Its goal is to share data, resulting in a Web of Data on the web of documents of the traditional web. The Semantic Web architecture makes use of various standards. XML, for example, can be used to create structure in webpages by annotating sections with tags, but it says nothing about the meaning of those structures [10]. For that, the Resource Description Framework (RDF) is used [13]. In the form of triples (i.e., `(subject, predicate, object)` tuples), assertions between things and properties are made, and a knowledge graph is built. The elements of the triples are either literals or URIs, with the latter pointing to further information about a resource. This Linked Data principle enables information linking and ensures unique definitions of concepts [4]. RDF alone does not prevent different systems from using different terms for the same concept. For computers to understand and reuse information across systems, shared vocabularies and ontologies, which define common concepts and relationships, are needed. A vocabulary is usually a simple description scheme specified in RDF Schema (RDFS) [29]. More complex semantics are usually modeled in the Web Ontology Language (OWL) [17]. Lastly, the SPARQL query language [68] for RDF provides an SQL-like syntax to extract data.

At the same time, multimedia content on the Web has increased, and newly created media formats, as described above, allow metadata annotations. These are, however, not easily understandable by computers, as the different standards lack formal semantics [70]. To address this issue, Semantic Web technologies are required, and a number of ontologies for single media and multimedia as well as several different ontologies to annotate various types of multimedia documents have been proposed.

For image description, for instance, [49] proposes a vocabulary to describe regions of various shapes, such as polygons, rectangles, or circles, and link them to images together with additional information. Numerous ontologies based on MPEG-7 have been proposed to describe the semantics of general multimedia. The first attempt is made by [35], who models parts of MPEG-7 using the ABC core ontology by [45] as a foundation. [76] builds on this and extends it to an ontology called DS-MIRF. [36] takes a slightly different approach in their audio-visual core ontology and focuses more on adding genre or themes to segments. [9] proposes a framework that uses the core ontology DOLCE [25] with two multimedia ontologies, namely Multimedia Structure Ontology and Visual Descriptor Ontology. [26] is the first to produce a full MPEG-7 ontology, called Rhizomik, by automatically mapping XML Schema to OWL. [2] proposes COMM, a core ontology for multimedia annotation. Unlike previous ontologies, it does not directly translate MPEG-7, but the authors re-engineer MPEG-7 to capture the intended semantics of the standard fully. This ontology covers the most used parts of the specification.

Beyond MPEG-7, the Ontology for Media Resources 1.0 defines a core vocabulary to describe media resources and a mapping to a set of media formats that support metadata [67]. Its goal is to unify metadata properties across commonly used multimedia formats. The Multimedia Metadata Ontology (M3O) by [64] also looks beyond MPEG-7 and provides a generic modeling framework that can be integrated with media formats like SMIL or SVG.

2.2 URI-based Annotation

With the rising presence of multimedia on the Web, there were efforts to encode segment information in URIs. [5], later superseded by [6], defines the generic URI syntax as a sequence of components: `<scheme>:<authority>/<path>?<query>#<fragment>`

The mandatory `scheme` defines the context of the URI, like the protocol or notation used. The `authority` indicates the place of governance of the remainder of the URI, such as a domain or an IP address. The `path` then identifies the desired resource, followed by the `query` and `fragment`, identified by question marks and number signs, respectively, to target components within that resource. While both query and fragment can be used to refer to segments of the resource, they come with different consequences. Queries in the form of key-value pairs are sent to the server and interpreted, after which the server returns a new resource, which has no relation to the original one [80]. Fragments keep the reference to the original by identifying a secondary resource, which is a subset or view of the primary one. Conversely, URI fragments are not transmitted to a server but are interpreted only client-side after the complete original resource is transmitted [30].

To formalize the description of segments as part of a URI, several standards have been proposed, which the following describes in closer detail.

2.2.1 Hypertext Markup Language

In the Hypertext Markup Language (HTML), URI fragments are used to reference anchor elements and allow jumping to specific content on the same or another page. Anchors are created using either the `name` or `id` attribute. Such a fragment URI, for example, looks like this: `http://example.com#section-1`. To resolve it, the fragment name is compared to all element IDs of the document tree, and the first matching element is used [32]. To solve the problem of changing IDs and broken references, text fragments can be used instead, which directly reference textual content [12]. Being a relatively recent specification, it is only implemented in Chromium-based browsers at the time of writing.

2.2.2 XPointer

In XML, the XPointer Framework defines ways of addressing specific components through URI fragments [28]. Besides the shorthand form for referencing specific element names, two schemes allow advanced references. The `element()` scheme can be used to traverse the XML tree and target specific children [27] (e.g., `example.xml#element(/1/4)`). The `xpointer()` scheme extends the functionality even more and allows operations on ranges and strings, allowing highly specific addressing of segments [18] (e.g., `example.xml#xpointer(string-range(//title, 'text', 4, 5))`).

2.2.3 Text

[78] proposes the usage of URI fragment identifiers for plain text files. Its proposal, later partly captured in [79], includes the `char` and `line` schemes, used either as single positions or ranges. For example, the segment of lines 11 to 20 is identified as: `http://example.com/text.txt#line=10,20`.

Additionally, the `length` and `md5` schemes are provided for integrity checks. In a similar fashion, [31] defines URI fragment schemes to select specific rows, columns, and cells in comma-separated values (CSV) documents. For example, a segment of cells that starts at the second row and second column and ends at the eighth row and sixth column is targeted like this:
`http://example.com/data.csv#cell=2,2-8,6`

2.2.4 MPEG-21

MPEG-21 is another standard developed by the MPEG, focusing on creating, delivering, and consuming multimedia [38]. The key components of the MPEG-21 framework include *digital items* created by and exchanged between *users*. Digital items are virtual containers for *resources*, which include both multimedia objects as well as metadata [11]. One such example is a music album, which, besides the actual audio tracks, may include lyrics, copyright information, cover art, and metadata about the songs or the artist.

Most relevant for this paper, part 17 of the specification defines URI-based fragment identifiers [39]. With four different schemes, which can also be combined, specific parts of resources can be addressed:

- `ffp()` identifies items and tracks by their name or ID.
- `offset()` is used to indicate the start and, optionally, the length of a byte segment of the resource.
- `mp()` is applicable to audio-visual resources and can be used to specify temporal, spatial, or spatio-temporal segments in numerous ways.
- `mask()` applies a binary mask to a video resource.

The following example applies both a temporal and a spatial segmentation to a video:
`video.mp4\#mp(~time('ntp','0:00:05','0:00:15')*mp(~region(rect(100,100,200,200)))`

While these schemes provide powerful means to address segments without indirections, [52] deems them over-designed and ambiguous due to their complexity. For instance, both `ffp()` and `mp()` can be used to segment a track. Additionally, only MPEG formats are supported.

2.2.5 Temporal URI

Temporal URIs have been proposed as part of the Continuous Media Markup Language (CMML) and the file format Annodex, which together enable hyperlinking and annotating continuous media, like audio or video [57, 58]. A user can express one or multiple temporal segments by specifying start and end times: `video.mp4?t=15/25`. The specification allows both URI fragments and URI queries to be used depending on the needs.

2.2.6 Media Fragments URI 1.0

All file standards described above, although providing means to specify temporal and spatial segments, have certain problems associated, like indirections, complex expressions, or too narrow scopes. Therefore, [73] concludes that a standardized way to address segments is needed, which should follow the *cool URIs* practices published by [16]. This allows media content to become a first-class citizen on the Web. Based on this proposition, the Media Fragments URI 1.0 W3C specification was created [75, 52]. URI fragments are chosen over URI queries, as fragments keep the relation to the original. That way, a media fragment is of the same file type as the parent resource. This setup ensures that media fragments follow the Linked Data principles [30]. The specification includes four fragment dimensions, which, due to being logically independent, can be combined freely:

- Temporal fragments are set with start and end time, where different time formats are supported: `video.mp4#t=5,15`
- Spatial fragments are limited to rectangles, specified as pixel coordinates or in percentages. The rectangle is distinctly defined by `xywh`, i.e., the position of the top left corner pixel and width and height of the bounding box: `image.jpg#xywh=200,200,100,50`
- One or multiple tracks can be segmented from media objects with multiple tracks (e.g., video, audio, subtitle track) by specifying track names: `video.ogg#track=audio`
- The id dimension is used to select a named fragment, such as a chapter from an audiobook: `audio.mp3#id=chapter-1`

In major browsers, however, only temporal fragments are natively supported.[2] Using the other dimensions requires custom software. As far as we were able to ascertain, there are also no other applications, such as dedicated media players, that natively support media fragments. In addition to that, existing Web infrastructure can only use byte ranges to refer to parts of a resource, like a temporal or named fragment. To resolve and retrieve media fragments over HTTP, different strategies are proposed in the specification and in literature [19, 22, 21, 48, 80].

- *No mapping*: As defined in [5], URI fragment identifiers are only interpreted by the user agent after the full resource is downloaded from the server. This approach, which can be found in browser implementations, only requires the user agent to be able to interpret and render media fragments. On the downside, this wastes bandwidth, as potentially large parts of the resource are discarded upon retrieval.
- *User-side mapping*: The user agent maps fragments to byte ranges. This is possible when the user agent has already downloaded parts of the resources, like headers or an index, and knows about the mapping. Then, a partial content request for this specific range can be sent to the server.
- *Server-side mapping*: The user agent requests the fragment in a special header, and the server performs the mapping. While special extensions must be installed on the server, this solution is the most efficient for bandwidth and latency. [19] implements a prototype in NinSuna, a platform for multimedia content selection and adaptation [20].
- *Third-party mapping*: A Media Fragments Translation Service (MFTS) [48] performs the mapping to byte ranges, either as a service for the user agent or as a proxy between a user agent and server. This approach requires minimal changes to the existing infrastructure.

While proper implementation of the full specification hence requires different client and server considerations, there exist various media management and retrieval systems that rely on media fragments, like the works of [42], [56], or [66]. The last, for example, implements a prototype system that can first segment videos of parliament speakers using video and audio segmentation techniques and second present the segments together with additional annotations on parliamentary websites. Beyond that, [47] uses media fragments for adaptive streaming over HTTP.

[2] `https://bugs.chromium.org/p/chromium/issues/detail?id=94368`
`https://bugzilla.mozilla.org/show_bug.cgi?id=648595`
`https://trac.webkit.org/changeset/104197/webkit`

2.2.7 Media Fragments URI Extensions

The media fragments URI specification exhibits a number of limitations, such as no spatio-temporal fragmentation or spatial fragments being limited to rectangles. To address these limitations, extensions to the specification were proposed by [43]. Those extensions provide more options for spatial fragments, like circles, ellipses, or polygons. They also introduce static and animated shape transformations, which can be applied to spatial fragments. These additions enable more precise fragment specifications, as well as spatio-temporal fragments. The authors also provide a CSS extension to style fragments directly. On GitHub, more similar extensions can be found.[3]

While media fragments can be used in Semantic Web technologies, multimedia-specific queries are not supported. Hence, [44] presents SPARQL-MM, an extension that provides spatio-temporal filter and aggregation functions to SPARQL. This allows, for example, querying a video snippet that contains two specific persons, as long as the persons are identified by a fragment URI. [55] follows a similar idea and provides support for temporal SPARQL queries with media fragments. They introduce query logic that can compare temporal fragments to, for instance, query events that happened before a certain event.

2.2.8 International Image Interoperability Framework

The International Image Interoperability Framework (IIIF), as described by [71], is both a set of standards for image delivery and a community of cultural heritage institutions, like museums, libraries, and archives. Its goal is to allow the sharing of digital images using Linked Open Data principles. It provides application programming interfaces (APIs) for interoperable access to images and metadata stored in different repositories. At its core, IIIF has two APIs: The *Presentation API* provides metadata in the form of a JSON manifest. It describes a IIIF item (e.g., a book, newspaper, or artwork) as a collection of references to resources (e.g., scans of individual pages) and defines its structure and layout. Additionally, it can contain annotations, where media segments are typically referenced following the Media Fragments URI specification. Various IIIF clients exist that can load and display these manifests. Images are queried through the *Image API* with standard HTTP requests. A specific image and desired characteristics are specified in a URI following the format: `<image API>/<identifier>/<region>/<size>/<rotation>/<quality>.<format>`. The `identifier` references the image, `region` defines a rectangular segment, either as pixel coordinates, percentages, or “full”. `Size` and `rotation` are used to scale and rotate the region, respectively, while `quality` allows turning the image grayscale or black and white. Lastly, `format` specifies the image file format. The client application uses the region specifier to only fetch the image section currently in view. An advantage of this is that the browser can work with standard HTTP requests and does not need to distinguish between statically served and dynamically generated images. Thus, unlike when relying on the Media Fragments URI specification, the browser requires no additional logic to retrieve partial images. This fact makes it convenient to use IIIF image URIs in the general Semantic Web context independently of the presentation API [51].

With IIIF, the image server does all the work. It prepares the requested image segment from the original according to the request URI and sends it back over HTTP. The requested images are generated “on the fly” from high-quality images and transformed into the requested format. There exist several image server implementations online[4] and in literature [61].

[3] `https://github.com/tomayac/dynamic-media-fragments`
`https://github.com/oaubert/mediafragment-prototype`

[4] `https://iiif.io/get-started/image-servers/`

It has been recognized that IIIF is limited to images and should be extended to support other media formats, such as audio and video. To that end, [15] coins the acronym IxIF to represent all non-image resources and provides an interim implementation; an audio and video (A/V) working group has been set up subsequently. The initial timetable was to extend the presentation API and then move on to content APIs. For the latter, [59] looks into an IIIF-based video framework and conducts some experiments on encoding and decoding speeds of videos, which is crucial for real-time delivery. Later, the working group decided to only focus on the presentation API, as it was deemed much more important. Nonetheless, there exists one account of an IIIF-like API for A/V content [50]. This API defines the request syntax `<A/V API>/<identifier>/<region>/<size>/<rotation>/<temporal_region>/<quality>.<format>`, where the added `temporal_region` parameter determines the time interval and controls playback speed.

2.2.9 EPUB Canonical Fragment Identifier

The EPUB Canonical Fragment Identifier (EPUB CFI) specification defines ways to access specific content inside an electronic publication (EPUB) document [72]. Such a reference code can, for example, look like this: `book.epub#epubcfi(/6/4[chap01ref]!/4[body01]/10[para05]/3:10)`. Slashes are used to navigate the document tree, where even and odd indices describe child elements and chunks of character data, respectively (`/6` therefore denotes the third child element, while `/3` refers to the second chunk of data). Exclamation marks perform an indirection step by accessing another document referenced in the current element. Square brackets assert the ID of the current element, and colons are used for character offsets. While the standard is generally used to extract XML or text elements, it also features identifiers for temporal, spatial, and spatio-temporal offsets. A fragment identifier in this format tends to become complicated quickly and only applies to the EPUB format, thus further limiting its applicability as a fragment representation.

2.2.10 OData

The Open Data Protocol (OData)[5] is a REST-based protocol for defining CRUD operations. It enables the encoding of filtering operations and function calls into HTTP URLs. An example for retrieving a filtered list of documents authored in or after 2024 that contain the term "Painting" in their title could look like this:
`/OData/Documents?\$filter=AuthorDate geq 2024-01-01 and contains(Name, "Painting")`.
While OData is primarily used for structured data encoded in JSON, it has also found uses in hypermedia applications.

2.3 General Fragment Model

A more generalized approach to segment representation is presented in [23, 24]. They argue that all fragment identification schemes described so far are too narrow and basic. For this reason, they propose a General Fragment Model (GFM), an ontological model that can describe fragments of generalized media types and query them.

The model has the following three main concepts. First, *information artifacts* are objects with codified content, such as images, text, or audio files. Second, *indexers* are functions that map tokens to fragments of an information artifact. Lastly, an *anchor* is a particular token applied to an indexer and denotes a fragment of the information artifact. The set of possible tokens for a particular indexer is defined as a *token set*. Figure 1 shows the relations of these concepts.

[5] `https://www.odata.org/`

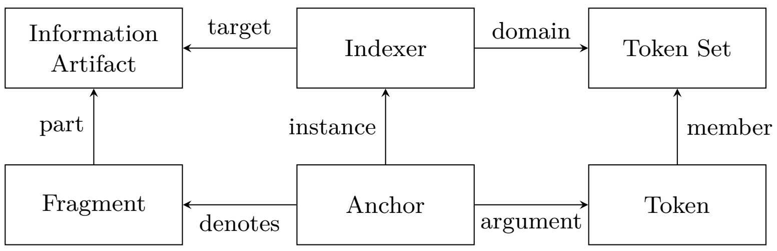

Figure 1 Conceptual model of the General Fragment Model adapted from [23].

A simple example is an image information artifact and a `rect` indexer function that extracts rectangular sections from the image. A possible anchor is then `rect(img, [0,0,100,100])`, which defines a fragment of size 100×100 in the top left corner of an image called `img`. Instead of `img`, another anchor can be passed, allowing for the arbitrary composition of anchors, like extracting a rectangular fragment but only the red color channel: `channel(rect(img, [0,0,100,100]), "red")`.

The authors further investigate classes of indexers and present a taxonomy of non-disjoint categories.

- *Identity indexers* map all segments to themselves.
- *Binary indexers* enumerate all bits of information of the information artifact. This indexer exists for every information artifact.
- *Tabular indexers* are further distinguished into *vector indexers* and *dictionary indexers*. Vector indexers create a vector space for the information artifact, such that the indexer domain is a totally-ordered set. This constructs a partial order of information parts. The *binary indexer* is a special type of this. Dictionary indexers map non-ordered input symbols to parts of the information artifact.
- *Spatio-temporal indexers* allow the selection of n-dimensional regions.
- *Query indexers* are formulae that reference media segments according to some criteria. The authors compare them to prepared statements in query languages like SQL.

The model is implemented in the knowledge representation framework *Hyperknowledge*, which focuses on multimedia content. It uses the construct of anchors to link fragments to graph nodes. For example, a spatial anchor on an image node represents a section of that image. By keeping the notions of indexer function and tokens very general, the GFM can be implemented in different forms supporting many use cases.

2.4 Interpretations and Implications

In summary, many ways exist to represent segmentation, either as a metadata annotation external to the medium or placed in a URI path, query, or fragment. One can observe some commonalities and tendencies in this multitude of proposals, further motivating this paper.

The number of proposed standards is remarkable, especially considering the relatively short period at the beginning of this century, in which most emerged. It indicates that many researchers saw a need for segment annotation at roughly the same time, but no single proposal could outdo the others. This, in turn, hints that each solution has shortcomings, be it only a very specific use case, restricted media support, or limited segment options. Conversely, universal and diverse approaches, such as MPEG-7 and later MPEG-21, quickly tend to be too complex and ambiguous. This is arguably also true for multimedia ontologies, where many either focus on specific media types or try to model MPEG-7. In many cases, the proposed standards are not widespread but occupy a rather niche space without much technological support.

Table 1 Overview of segment annotation mechanisms found in the literature. ✓* indicates that support for a functionality requires an extension beyond the original definition.

	SMIL	SVG	MPEG-7	XPointer	MPEG-21	Media Fragments URI	IIIF	EPub	GFM
External Metadata	✓	✓	✓	–	–	–	–	–	✓
Server-side	–	–	–	–	–	partial	✓	–	–
Spatial	some	✓	✓	✓	✓	✓*	some	✓	✓
Temporal	✓	–	✓	–	✓	✓	✓*	–	✓
Spatio-temporal	some	–	✓	–	✓	✓*	✓*	–	✓
Filter	–	some	–	–	–	–	some	–	✓
Text	–	–	–	✓	–	✓	–	✓	✓
Image	✓	✓	✓	–	✓	✓	✓	–	✓
Audio	✓	–	✓	–	✓	✓	✓*	–	✓
Video	✓	–	✓	–	✓	✓	✓*	–	✓
3D	–	–	–	–	–	–	–	–	✓
Type-agnostic	✓	✓	–	–	–	–	–	–	✓
Format-agnostic	✓	✓	–	✓	–	✓	✓	–	✓
Implementation Available	✓	✓	✓	✓	–	partial	✓	✓	–

The first somewhat common approach is the Media Fragments URI 1.0 specification, whose authors identified the above-mentioned issues. The result is a standard describing temporal, spatial, track, and id segments in a URI. This is, however, not without problems. Most notable for this paper is that while different segment types are supported, their flexibility is still lacking, as, for example, spatial segments can only be rectangles. Although there are proposals for extensions, none of them seem to have been picked up, which appears to be a general theme again. A considerable body of literature exists about the specification, but real applications are relatively rare, and browser support is still partially or fully lacking.

Interestingly, the standard with the most real-world relevance appears to be the IIIF image API, which is the only one coming from a different origin and motivation. Nonetheless, it offers a way to describe an image segment with a distinct URI, making it suitable for use outside of IIIF as an identifier of a resource in a knowledge context. Again, the segment flexibility is minimal, with only rectangles supported, and extensions were looked into but, so far, came to nothing.

Table 1 provides a comparison of the concrete segmentation annotation mechanisms discussed in this section. It shows that there is a broad range of capabilities, both in theory and in practice, across the various schemes that can be found in the literature. Most of the schemes focus on one specific type of media, most commonly text or images, or on a specific set of use cases.

Generalizations towards arbitrary media or segmentations are underrepresented in literature, with the General Fragment Model being the only account. It shares a similar derivation and motivation with this paper but then focuses more on an ontological structure. While it can describe virtually any segmentation, it offers few deductions of properties toward possible implementations. There appears to exist an implementation of numerous types of segmentations following the General Fragment Model, but these seem to be built case by case instead of in a general manner, as the model suggests. Unfortunately, the implementation is mostly closed-source and provides few insights into the applied mechanics.

We aim to close this gap by proposing the Unified Segmentation Model for media documents. The model combines the expressiveness of the various methods discussed above into a simple and unified representation. Furthermore, it is specified in such a way that it does not rely on the client to implement the segmentation scheme. This makes it much more easily interoperable with existing standards and technologies. The following discusses the model and how it can be used in the context of multimodal knowledge graphs.

3 Unified Segmentation Model

In this section, the formal part of the Unified Segmentation Model is designed and presented. It aims to abstract various media segmentations and provide a generalized segmentation operation independent of the media document type. First, the formal model is developed, and properties of different types of segmentations are discussed. The second part investigates approaches for describing segmentation rules. This sets the stage for concrete types relevant to the implementation.

3.1 Multimedia Object

To arrive at a model of segmentation, the first step is to generalize multimedia documents regardless of their type. As established previously, any multimedia document consists of n dimensions, where each can be temporal or spatial. An audio track, for example, has one temporal dimension, an image has two spatial dimensions, while a video comprises one temporal and two spatial dimensions. These n dimensions span an n-dimensional space in which a concrete multimedia object can be seen as a bounded, finite subset of the space. Usually, this subspace is not continuously defined but consists of a finite amount of discrete data points. In a raster image, these are the pixels, and the subspace is bounded by the width and height of the image, whereas a video is a temporal enumeration of images additionally bounded by a duration.

The data point coordinates only determine the extent of the media object. To fully represent a specific piece of multimedia, additional description information for each data point can be required, again consisting of one or multiple dimensions. For a digital audio track, each sampling point has an amplitude. For an image, each pixel needs a color value, for instance, a mixture of red, green, and blue colors. Alternatively, the descriptions can also be empty, as is the case with a 3D model, provided that no materials or surfaces are defined.

Formally, any arbitrary multimedia object M can thus be described as

$$M \subset P \times D, \tag{1}$$

where P and D indicate the space of possible data points and descriptions, respectively. They are both defined as n-ary cartesian products of a number of dimensions

$$P = dim_1 \times \cdots \times dim_n \qquad \text{and} \qquad D = dim_1 \times \cdots \times dim_m, \tag{2}$$

where each dimension dim is an arbitrary number set, such as $\mathbb{N}$ or $\mathbb{R}$. The number set depends on the application and can have constraints. An RGB color channel, for instance, is limited to 256 elements, while each ASCII character is one of 128. Figure 2 gives a visualization of this definition for two- and three-dimensional data points. Formulated differently, the media object M defines a binary relation with domain P and codomain D. Such a binary relation fulfills the functional (right-unique) and total property, i.e.

$$\forall p \in P,\ \forall d_1 \in D,\ \forall d_2 \in D : ((p, d_1) \in M \wedge (p, d_2) \in M) \Rightarrow d_1 = d_2 \tag{3}$$

$$\forall p \in P,\ \exists d \in D : (p, d) \in M \tag{4}$$

In other words, this means that data points are unique, while their descriptions are not, and every data point has a valid description. From this, it follows that the relation is a function $P \rightarrow D$ but is neither injective nor surjective.

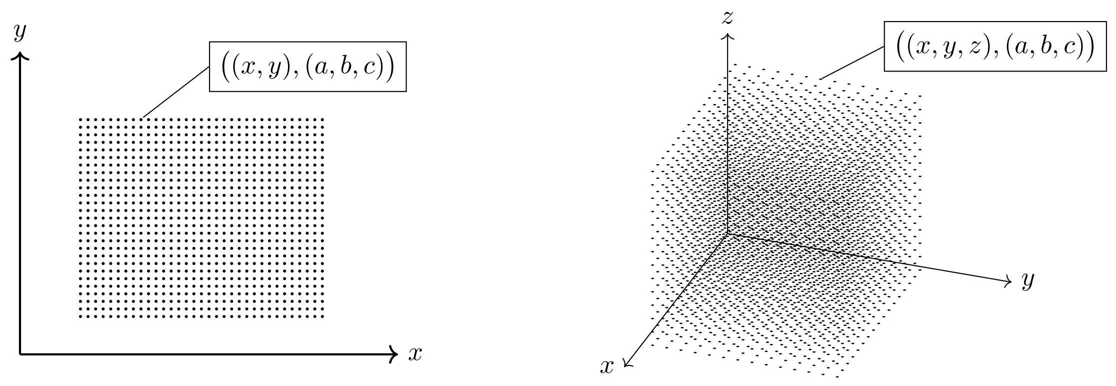

Figure 2 Schematic representation of a media object consisting of ordered pairs of data points and descriptions. Each data point has n dimensions (here x, y or x, y, z) and each description has m dimensions (here a, b, c).

3.2 Segmentation Operation

Segmentation is defined as the process of extracting a part from a whole. Applying this to the formal definition of a multimedia object, a segmentation function can be defined as

$$s : M \rightarrow M', \quad M' \subseteq M. \tag{5}$$

It takes as input a media object M, which is an arbitrary subset of the space $P \times D$, and yields another media object M' of the same space. To uphold the definition of segmentation, the result must be a subset of the original media object. This very general function s can have all sorts of implementations depending on the segmentation and media object at hand but can be grouped into three variants – filters, reductions, and transformations – as well as post-processing operations, which we discuss in the following.

3.2.1 Filter Segmentation

Most segmentation tasks revolve around wanting to retain a subset of the data points of the initial media object. The filter operation is thus generally defined as

$$s_f(M) = \{m \mid m \in M \wedge f(m)\}, \tag{6}$$

where the filter function

$$f : P \times D \rightarrow \{true, false\} \tag{7}$$

is an *indicator function* that decides for each data point in the space whether it should be kept or not. It can use any properties of the data point itself or its description to make the decision. Examples of such filter segmentations are the selection of a rectangular cutout of an image, a scene from a movie, an excerpt of an audio recording, or one page from a PDF document.

The filter segmentation is not limited to a single filter function but can consist of an arbitrary number of sub-filters $f_1, \dots, f_k$, which are combined using Boolean algebra operations. For only conjunctions, the filter operation is defined as

$$s_f(M) = \{m \mid m \in M \wedge f_1(m) \wedge \dots \wedge f_k(m)\}. \tag{8}$$

For arbitrary Boolean operations, data points are part of the segment if the Boolean combination of indicator functions yields 1. This allows the mixture of any filter criteria, for instance, only keeping image pixels of certain colors, which are located in a certain area of the image and do not have large illumination changes compared to neighboring pixels. From Equation 8, it follows that

$$\begin{aligned} &\{m \mid m \in M \wedge f_1(m) \wedge \dots \wedge f_k(m)\} \\ \Leftrightarrow\ &\{m \mid m \in M \wedge f_1(m)\} \cap \dots \cap \{m \mid m \in M \wedge f_k(m)\} \\ \Leftrightarrow\ &s_{f_1}(M) \cap \dots \cap s_{f_k}(M), \end{aligned} \tag{9}$$

using the definition of set intersection. Equally, by the definition of set union, it follows that

$$\{m \mid m \in M \wedge (f_1(m) \vee \dots \vee f_k(m))\} \ \Leftrightarrow\ s_{f_1}(M) \cup \dots \cup s_{f_k}(M). \tag{10}$$

This shows that the conjunction and disjunction of sub-filters are equivalent to the intersection and union of segments, respectively, where each segment is created using one particular sub-filter. The same relationship can also be shown for arbitrary combinations of operators.

As the segmentation result is another media object, multiple filter segmentations can be applied consecutively, i.e.,

$$s_{f_k}\left(s_{f_{k-1}}\left(\dots\left(s_{f_2}\left(s_{f_1}(M)\right)\right)\right)\right) = (s_{f_k} \circ \dots \circ s_{f_1})(M). \tag{11}$$

Thereby, the multimedia object is increasingly filtered by every segmentation. This corresponds to applying all filter operations in the same segmentation operation, which can be written as a conjunction of filters. As derived above, this is equivalent to the intersection of the segments that result from applying the filter segmentations to the original media object. It follows that nested filter segmentations are commutative:

$$(s_{f_k} \circ \dots \circ s_{f_1})(M) = s_{f_1}(M) \cap \ldots \cap s_{f_k}(M) = (s_{f_1} \circ \dots \circ s_{f_k})(M) \tag{12}$$

As an illustration, consider a temporal video segment, which is afterward cropped to a square. Its result is equivalent to first cropping the whole video and then cutting it to the desired length. The segmentations can also operate on the same dimensions, such as in an image with overlapping segment areas, where the order of segmentation does not matter for the outcome.

3.2.2 Reduction Segmentation

Alternatively, a user might want to keep all data points but segment the description of each point. This reduction operation

$$s_r(M) = \{(p, r(d)) \mid (p, d) \in M\} \tag{13}$$

applies a reduction function

$$r : D \to D', \quad \dim(D') \leq \dim(D) \tag{14}$$

to the description of each data point. Like the filter function, this dimensionality reduction function is not allowed to modify the individual data points but can only select which dimensions to keep or discard. A possible application could be extracting one color channel of an image.

The reduction function can be seen as a special filter function, which operates on only the codomain of the media object. However, it does not filter elements of the set but the description dimensions of each element. In other words, the scope of each data point description is changed. Hence, set theory operations are not applicable in a mathematical sense. However, one can interpret the description of a data point as an m-dimensional set instead of an m-tuple of values. Thereby, the reduction function can be described as a filter function on each set of values, enabling the combination by set operations. Then, the reduction operation is commutative. See Section 4.2 for a practical example.

Filter and reduction operations can also be combined but require caution, as commutativity is not always given. In some cases, the filter operation might use properties affected by a reduction operation, for example, a filter operation based on color and a reduction operation that drops specific color channels. In other cases, segmentations are commutative, like selecting one color channel and cropping the image. A solution to this uncertainty is adapting the filter operation to only use data point information (i.e., P) without changing its semantics. For instance, the filtering based on color can also be represented as filtering of certain point coordinates that happen to have said color. With this trick, filter and reduction functions operate on disjoint pieces of information, which ensures commutativity.

3.2.3 Transformation

Besides operations that directly affect ordered pairs of the set, utility operations need to be considered. One issue is the fact that a certain media object can have different representations, where each one is defined in a different space. To ensure that any segmentation can be applied, the media object might need to be transformed in preparation. Such a transformation function

$$t : M \to M', \quad M \subset P \times D, \quad M' \subset P' \times D' \tag{15}$$

transforms a media object M from one space to a media object M' in another space, such that M is semantically equivalent to M'. The two media objects can have different sets of dimensions. An example of that is the Fourier transformation of an audio signal from time and amplitude dimensions to frequency and magnitude dimensions. Any such transformation must be reversible, such that the inverse transformation can be applied to the segmented media object to restore the format of the original media object. Note, however, that such transformations can be lossy.

3.2.4 Postprocessing

Postprocessing operations can be applied to the result of a segmentation operation. Strictly speaking, these are not part of the segmentation definition, as they modify the media object instead of just selecting a fraction of it. However, for real-world applications, postprocessing can be vital for usability, as it allows speeding up a song, rotating an image, performing anti-aliasing to smoothen edges, or removing a constant dimension. The last allows turning a single frame of a video, which is so far just a very short video, into an image by removing the temporal dimension. As these operations do not fulfill the segmentation definition, they are not discussed further in this context.

3.3 Compound Media Object

So far, a media object is considered a subset of a single n-dimensional space. However, there are cases where the media object comprises a collection of individual media. Such a compound media object C can be defined as

$$C = \{M_i \mid i = 1, \dots, m\}, \tag{16}$$

where each M_i corresponds to a media object as defined in Equation 1 and can potentially have different dimensionalities and structures. A prime example of a compound media object is a movie, which consists of a video track, one or multiple audio tracks, and possibly subtitles.

The segmentation operations established before are also applicable to compound media objects by applying the operation to each individual sub-object, i.e.,

$$s(C) = \{s(M_i) \mid M_i \in C\}. \tag{17}$$

In practice, this can result in any number of sub-objects being affected by the segmentation, and the others are left unchanged. If the movie is cropped, only the video track is modified, but if a scene is extracted, all sub-objects are temporally segmented. A special case of compound media object segmentation is removing all data points from one or more sub-objects. This corresponds to a subsetting operation and can, for example, be used to extract a movie's audio. While this is technically a filter operation that leaves certain media objects empty, it resembles more a reduction operation, where certain elements are dropped.

3.4 Segmentation Definedness

So far, the filter and reduction operations retain a subset of data points and description dimensions, respectively. An issue with this generalization is that the promise of the segmentation definition, extracting a part from a whole, is not necessarily given. With the ultimate goal of real-world applicability, some considerations need to be made.

The filter operation may select multiple non-overlapping clusters of data points. For example, two rectangles from an image with space between them. To keep the semantics of the media object and thus remain a valid segmentation, the segmented data points need to keep their relative position. Therefore, the space between the rectangles has to remain, and the two segment parts cannot be shrunk together. To this end, all data points inside the bounding box of the segmentation need to be retained. Only data points beyond the bounds can be safely discarded. To still be able to represent any segmentation, the description must be set to null if the data point is not part of the desired segment. For that, the notion of neutral elements o and neutral descriptions $(o_1, \ldots, o_m)$ is introduced. A neutral element is a special value of that specific dimension, which indicates the absence of that dimension. Neutral descriptions are m-tuples of neutral elements, which fully void the description part of a data point without changing its signature. In the example of the two rectangles in an image, all pixels between the rectangles are neutralized by setting them to transparent color values. Regions outside of the rectangles can be cropped safely.

The reduction operation suffers from a similar issue. If the dimensionality of the data point description is changed by discarding undesired dimensions, the media object can potentially lose its definedness. For instance, an image in RGBA color space can get rid of its alpha channel and still be a valid RGB image, but an image with only two color channels is invalid or requires special file formats for storage. As a solution, the reduction function needs to replace dimensions not part of the segmentation with a neutral element o_j of the specific dimension. For RGB, the undesired color channel would be set to zero for each data point, such that this color no longer contributes to the image.

4 Applied Segmentation

To ultimately be able to implement the Unified Segmentation Model in a real application, another level of abstraction needs to be discussed. First, the classes of operations outlined above need to be refined into more distinct and meaningful subclasses covering all realistically desired segmentations.

Second, thought must go into the serialization of segmentation. In other words, schemes must be developed on how the segmentation instructions can be represented textually. The goal is to fit this description into a URI to ensure both usability and compactness without sacrificing expressiveness. All segmentation URIs described in this chapter have the form

```
<reference to multimedia object>/<segmentation type>/<segmentation description>
```

First, the base of the URI has to reference the media object on which the segmentation is performed. For convenience, this part is omitted from here on. Then, the URI needs to inform about the specific segmentation type via a keyword, followed by the actual description of the segmentation of this type. The keyword is necessary to distinguish between similar segmentations, which cannot be inferred from the description alone. This allows proper segmentation processing and ensures uniqueness across types. Additionally, it needs to be discussed how to ensure that a particular segmentation has only one unambiguous URI representation, such that each segmentation is uniquely definable.

4.1 Filter Segmentation

For filter segmentation, we can distinguish between several different types of filter functions. The following lists their properties and presents URI schemes for the representation of each of them.

4.1.1 Segment Boundary

Segments can be described in terms of their boundary, as segment data points are usually coherent. Due to the limited amount of information needed, this approach is well suitable to be conveyed in a URI. This is supported by the observation that all URI-based segment annotation approaches presented in Section 2 describe segments in terms of boundaries. Depending on the dimensionality of the boundary, different considerations are necessary.

One-dimensional Segmentation

For segmentation in only one dimension, things remain relatively simple. However, flexibility is needed to handle different measures and units because many multimedia documents can be segmented along a single dimension. For instance, a video can be segmented with timestamps or frame numbers, while a PDF or text document is only segmented along integer page or character numbers, respectively. In all cases, a segment is serializable as a list of intervals in ascending order. This is showcased in Figures 3 and 4 with a 30 seconds audio clip and a 44-character text sentence, respectively. As an alternative to absolute numbers, the same segment can be equivalently defined using relative numbers in the range $[0, 1]$. The output can be shrunk to the bounds of the segment, cutting off the irrelevant parts at the beginning and end. Inside the bounds, parts not in the segment cannot be removed but must be nullified to adhere to the segmentation definition. In audio, this means muting non-segment parts, a video could turn those parts black or transparent, and text documents could replace the respective characters with whitespaces.

Sometimes, the media object is not represented in the correct space to apply the segmentation directly. In such a case, a transformation needs to be applied first. A prime example of this is segmenting an audio track by frequency, which requires a Fourier transformation from time and amplitude space to frequency and magnitude space. Then, one-dimensional segmentation can be performed as usual before the result is transformed back to its original space, here using an inverse Fourier transformation. It is to mention that while the formal definition required an explicit statement of this transformation and its inverse, this is no longer necessary but implicitly

```
/time/<list of ranges>

/time/5-12,15-22,26-28

/time/0.2-0.4,0.5-0.8,0.86-0.93
```

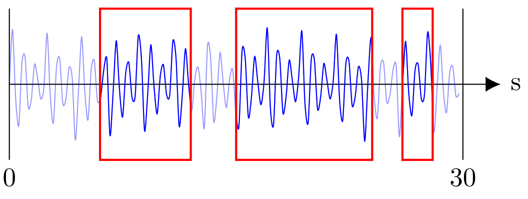

Figure 3 An example segmentation of an audio snippet by time in seconds. The absolute and relative URI representations on the left correspond to the visualization on the right, with the segmented parts shown in red.

```
/character/<list of ranges>

/character/4-12,15-22,26-28

/character/0.091-0.273,0.341-0.5,0.591-0.636
```

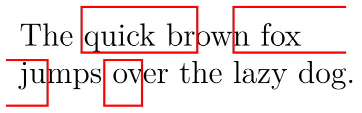

Figure 4 An example segmentation of a plain text by characters. The absolute and relative URI representations on the left correspond to the visualization on the right, with the segmented parts shown in red.

derived from the specified segmentation type. An example of a frequency segmentation is shown in Figure 5, where a frequency range is selected. This particular segmentation thus corresponds to applying both a high- and low-pass filter.

```
/frequency/<list of ranges>

/frequency/1000-4000
```

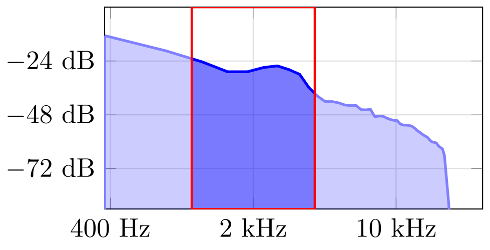

Figure 5 An example segmentation of an audio snippet by frequency between 1000 Hz and 4000 Hz. A Fourier transform operation precedes the segmentation to translate the time space to a frequency space.

Two-dimensional Segmentation

Segmentations along two simultaneous dimensions need to describe areas of a plane spanned by the two dimensions. This type of segmentation can be applied to various media, such as images, videos, or PDF documents. In all cases, the media object can be viewed as a single or stack of two-dimensional planes, and the segmentation punches out a specific shape or area of each plane. A suitable analogy would be that of a cookie cutter.

Rectangle

In the simplest case, a single rectangular shape is desired. Assuming that the rectangle is not tilted, it is sufficient to define the minimum and maximum x and y coordinates, either as absolute values (e.g., pixels) or relative numbers in range $[0, 1]$. An example is presented in Figure 6, where Big Buck Bunny is segmented from a screenshot of Blender Foundation's eponymous short film [8].

```
/rect/<xmin>,<xmax>,<ymin>,<ymax>

/rect/165,340,35,315

/rect/0.236,0.486,0.088,0.785
```

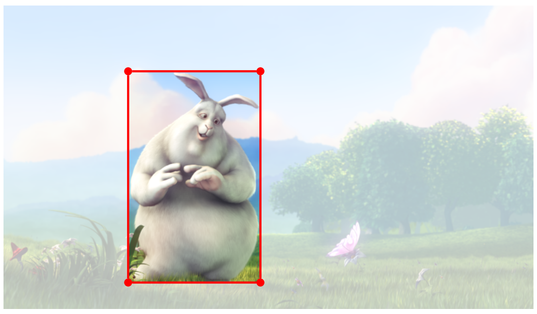

Figure 6 Rectangular segmentation of a 700 × 400 screenshot from Big Buck Bunny [8]. The URIs of absolute and relative point coordinates on the left correspond to the visualization on the right.

Polygon and Spline

If the segment area is still one closed shape but more complex, polygonal or spline structures can be used when the edges are straight or rounded, respectively. Such forms are describable by an enumeration of points on the plane. If a polygon is desired, these points are connected by lines between consecutive points. In the case of closed splines, the point requirements depend on the type of spline. For instance, cubic Bézier splines require each Bézier curve to have four control points, whereas basis splines need additional care to ensure a closed shape [69]. In any case, the points can be serialized as shown in Figures 7 and 8 as absolute or relative coordinates. To ensure the unambiguity of the URI, one can establish a rule in which winding direction to specify points and with which point to start, such as starting at the left lowermost coordinate and proceeding clockwise. The output image can be cropped to the rectangular bounding box of the segmentation. Pixels inside the bounds that are not part of the segment need to be changed to transparent. The model does not explicitly define how to handle pixels on the boundary.

```
/polygon/<point1>,<point2>,<point3>,...

/polygon/(170,35),(165,180),(225,320)
     ,(340,275),(335,65),(300,35)

/polygon/(0.243,0.088),(0.236,0.45)
     ,(0.321,0.8),(0.486,0.688)
     ,(0.479,0.163),(0.429,0.088)
```

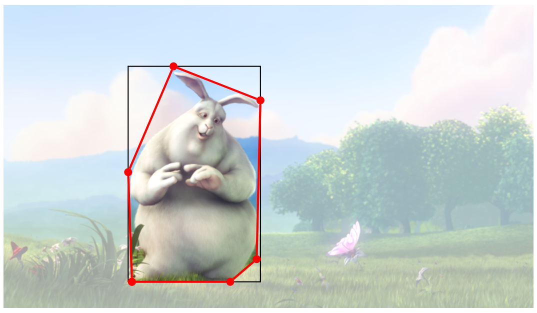

Figure 7 Polygonal segmentation of a 700 × 400 screenshot from Big Buck Bunny [8]. The URIs of absolute and relative point coordinates on the left correspond to the visualization on the right. The polygon is shown in red, and the resulting image is outlined in black, where pixels outside of the polygon are turned transparent.

They can either be assigned proportionally using an alpha component or completely based on the majority of their area (as is the case in our implementation, see Section 6),

SVG Path

As an alternative, or if the segment comprises disjoint segments, the concept of SVG's path property can be used to outline the segment area. Its specification is made to represent a shape as a description string, serialization is thus built-in. It features instructions to define linear and

```
/bezier/<start>,<control1>,<control2>,<
    end>,<control1>,...

/bezier/(190,30),(130,40),(150,300)
    ,(225,320),(400,360),(450,-20)
    ,(190,30)

/bezier/(0.271,0.075),(0.186,0.1)
    ,(0.214,0.75),(0.321,0.8)
    ,(0.571,0.9),(0.643,-0.05)
    ,(0.271,0.075)
```

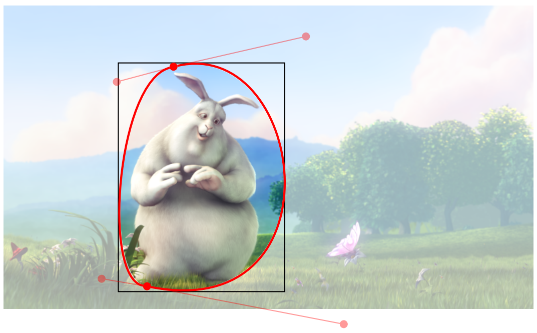

Figure 8 Bézier spline segmentation of a 700 × 400 screenshot from Big Buck Bunny [8]. The URIs of absolute and relative point coordinates on the left correspond to the visualization on the right. The shape is drawn in red, with semi-transparent Bézier control points, and the resulting image is outlined in black, where pixels outside of the shape are turned transparent.

curved line segments and an operator to move the cursor to draw disconnected shapes [40]. All of them are used in the example shown in Figure 9, where the Bézier spline segment from Figure 8 is extended with a disjoint triangular segment. On the downside, unique serialization is not as straightforward, especially with composite shapes. Nonetheless, one can still come up with some ordering, especially since the move operation allows a lot of flexibility.

```
/path/<svg path>

/path/M 190,30 C 130,40 150,300 225,300 C
     400,360 400,-20 190,30 M 410,60 L
    470,130 L 480,70 Z
```

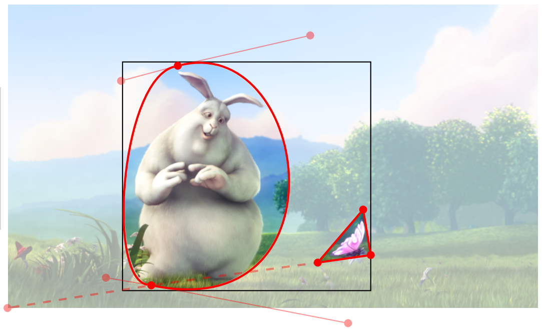

Figure 9 SVG path segmentation of a 700 × 400 screenshot from Big Buck Bunny [8]. The URI on the left corresponds to the visualization on the right. The segment is drawn in red, with the Bézier control points given as semi-transparent lines, and the cursor move command M is displayed as a dashed line. The resulting image is shown in black, where pixels outside of the boundaries are turned transparent.

Three-dimensional Segmentation

Three-dimensional segmentations need the most care, as their dimensional complexity can lead to computational and representational limitations. The cookie-cutter analogy no longer holds here, as the third dimension is also constrained in some way. Instead, one can think of these segmentations as taking the cuboid of the media object and carving it into, possibly multiple, arbitrary shapes.

Rotoscoping

If the segmentation is simple enough, a shortcut can be made by diminishing the importance of one dimension, therefore turning the problem into a 2.5-dimensional segmentation. In what is known in the film industry as rotoscoping, a 3D segment is described as a series of keyframes.

Each keyframe defines a two-dimensional segment at a point in the third dimension. For points not directly defined, the shape can be linearly interpolated from the nearest neighbors. To, for instance, segment a car passing through a video scene at a constant speed, it might be sufficient to segment only the first and last frame of the scene in question and interpolate everything in between. This approach, therefore, allows for smooth three-dimensional segmentations with minimal instructions. This is beneficial for serialization, as seen in the example in Figure 10. Nonetheless, this approach can also be used for complex segmentations, for example, by defining an image mask for each video frame.

```
/rotoscope/<timestamp>,<type>,<description>;
    ...;<timestamp>,<type>,<description>

/rotoscope/0,rect,0,100,0,100;
    400,rect,150,300,100,200
```

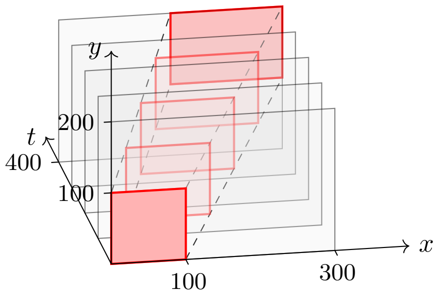

Figure 10 Rotoscope segmentation, where a series of 2D shapes determines the 3D segment. The URI of two rectangles at different points in time on the left implicitly defines shapes for time points in between, as visualized on the right.

3D Mesh

If the segmentation becomes too complex, it can alternatively be described as a three-dimensional mesh. Applying this mesh to a media object largely depends on the media type. In the case of a video, the 3D mesh must be sliced using the time coordinate to yield a 2D segment for each video frame. If the media object is itself a 3D object, the intersection must be computed between it and the segment, for example, using Constructive Solid Geometry algorithms [60, 46].

While this approach allows virtually any three-dimensional segmentation, the URI serialization can become complex, as the complete mesh definition must be embedded into it. For that, it makes sense to draw inspiration from simple mesh file formats, such as Wavefront OBJ, Polygon File Format (PLY), or StereoLithography File Format (STL) [54, 77, 1]. In the first, for example, vertices are specified with a `v`, followed by the coordinates, and faces with `f` and a list of vertex indices. The unit cube, for example, is then defined as displayed in Figure 11.

```
v 0 0 0
v 1 0 0
v 1 1 0
v 0 1 0
v 0 0 1
v 1 0 1
v 1 1 1
v 0 1 1
```

```
f 1 2 3 4
f 5 6 7 8
f 1 2 6 5
f 2 3 7 6
f 3 4 8 7
f 1 4 8 5
```

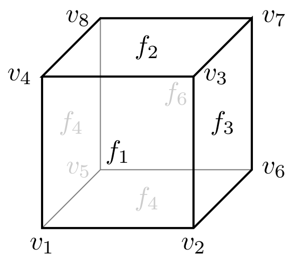

Figure 11 Unit cube in the Wavefront OBJ 3D mesh format. On the left, vertices are described by their coordinates and faces by their vertex indices, and the resulting hexahedron mesh is shown on the right.

Instead of listing the entities by line in a file, they can be encoded into a URI in a comma-separated sequence. One issue with these file formats and, hence, the URI serialization is that the order of vertices and faces can be arbitrary. The same object can thus have many different serializations. To solve this ambiguity, faces of more than three vertices can be triangulated, and faces and vertices can be sorted based on a given rule.

An applied example is shown in Figure 12, where the same three-dimensional segmentation as in Figure 10 is presented, this time as a 3D mesh in Wavefront OBJ format.

```
/mesh/<vertices>,<faces>

/mesh/v 0 0 0,v 0 100 0,v 100 0 0,v 100 100 0,
    v 150 100 400,v 150 200 400,v 300 100
    400,v 300 200 400,f 1 5 6,f 1 5 7,f 1 6
    2,f 1 7 3,f 2 6 8,f 2 8 4,f 3 1 2,f 3 2
    4,f 5 7 8,f 5 8 6,f 7 3 4,f 7 4 8
```

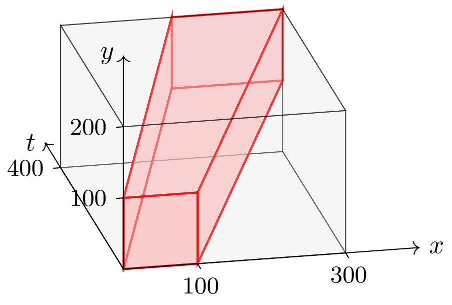

Figure 12 Three-dimensional segmentation defined using a 3D mesh. The serialized Wavefront OBJ format on the left corresponds to the red hexahedron on the right.

4.1.2 Cut Equation

An alternative to the straightforward boundary definition is the usage of algebraic expressions, which describe hypersurfaces and can cover a variety of smooth shapes. As these expressions require little description space, they are a good candidate for short URIs. In Figure 13, where the equation is $y > -0.019x^2 + 9.6x - 900$, all points above the curve are segmented as they fulfill the stated inequality. As a last step, the output can be cropped to the segment bounds, which is the whole image in this case.

```
/cut/<equation>

/cut/y>-0.019x^2+9.6x-900
```

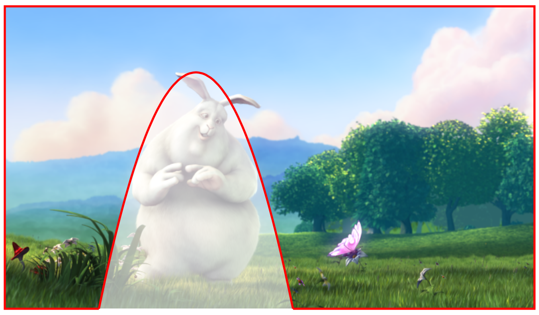

Figure 13 Two-dimensional segmentation of a screenshot from Big Buck Bunny [8] using a quadratic equation to define a cut. The URI must indicate which side of the cut to keep, either as a Boolean value or as an inequality. The URIs on the left yield the segment visualized on the right.

4.1.3 Explicit Masking

If describing segment boundaries becomes too complex, an alternative, which allows data point-accurate segment description, is a binary mask. Such a mask can be described in a URI as an enumeration of binary values. This scheme is usable for any media object type, but an enumeration order needs to be established for media of more than one dimension. For images, this can, for instance, be line-wise from top to bottom, as shown in the example in Figure 14. When the mask is two-dimensional, it can alternatively be represented as a serialized binary image, which can additionally be base64-encoded to shorten the length of it drastically. In that case, no ambiguities must be dealt with, provided the mask matches the media object. This approach, also presented in Figure 14, corresponds to the Data-URL scheme specified in [53]. In the example's case of only 10 × 10 pixels, listing binary values is more efficient, but with larger masks, the encoded PNG becomes superior. In the worst case, when the mask is completely randomized, the base64 encoding is one sixth the size of the binary enumeration. Otherwise, the encoding can be multiple orders of magnitude smaller with more coherent masks.

```
/mask/<mask>

/mask/0010001100110010011001100111000101
    011011010011011101110011101110011001
    001111100010011011110001110010

/mask/iVBORw0KGgoAAAANSUhEUgAAAAoAAAAKAQ
    AAAAClSfIQAAAAJklEQVR4XmNgZmA42cCQzs
    AQdoDB9wBDcQPDMwcG0waG2QcYZBoAcpAIJe
    jCIp4AAAAASUVORK5CYII=
```

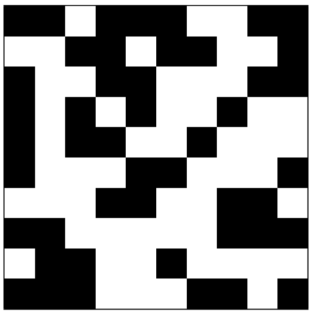

Figure 14 Two variants of describing a randomized two-dimensional segmentation mask, either as binary values or the base64 encoding of a binary PNG image.

4.1.4 Space-filling Curve

Clear enumeration orders of data points are provided by space-filling curves, making its application rather straightforward. Due to its locality preservation properties, the Hilbert curve is chosen for this purpose. While such a curve is defined to visit every data point, one can instead play with the curve's order. This determines the maximum index and thus controls the resolution of the segmentation. For example, a first-order Hilbert curve for two dimensions splits each dimension in half, yielding four sub-squares, each corresponding to one curve index. Therefore, the URI serialization of a segmentation needs to carry information about the order of the curve and a series of index ranges. An example of a third-order Hilbert curve applied to an image is given in Figure 15. It shows a roughly similar excerpt as the previous image segmentations, but precision could be improved by increasing the order and, hence, the number of index ranges. In the same way, Hilbert curves can also be used to segment media of higher dimensions, such as videos. In both cases, the media object can be cropped to the segment bounds, and data points not in the segment need to be neutralized.

4.2 Reduction Segmentation

Reduction segmentations have less variability, making them simpler to serialize and able to cover all desired segmentations simultaneously. In essence, one only needs to list the description dimensions to retain. The allowed dimension names depend largely on the concrete segmentation. In Figure 16, only the red and blue color channels are selected by listing them by their name in the URI. In the output image, the green color value is set to zero for all pixels, such that the image is still storable in RGB format.

```
/hilbert/<curve order>,<index ranges>

/hilbert/3,2,6-17,28-31,50,61
```

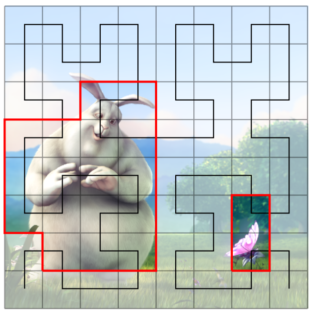

Figure 15 Hilbert curve segmentation of a square screenshot from Big Buck Bunny [8]. The Hilbert curve is of third order, which corresponds to 64 indices. The URI on the left first specifies the curve order, followed by the index ranges to segment. This segment is visualized on the right, where the curve starts with index 1 in the bottom left corner.

```
/color/<selections>

/color/red,blue
```

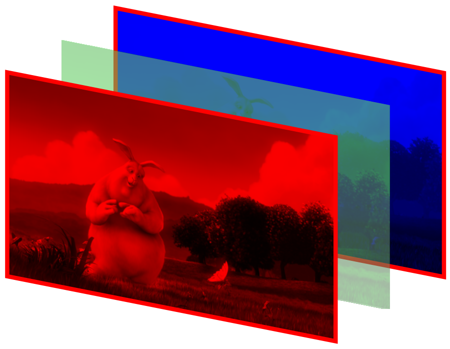

Figure 16 Segmentation by color of a screenshot from Big Buck Bunny [8]. The URI of a two-color selection on the left is visualized on the right.

4.3 Compound Media Segmentation

All types described so far also translate to compound media objects. In most practical examples, one or more sub-objects are segmented while the others remain untouched. The only thing left out is the special case of selecting a subset of media objects. Although this operation is, strictly speaking, filtering, it makes sense to classify this among the reduction segmentations for practicality. The main reason is that sub-objects to segment need to be specified in the same manner with a list of keywords. In the compound media example of a movie with video, audio, and subtitles, a segmentation for only audio could then look like `/stream/audio` or alternatively `/stream/1` if the audio stream is known to have an index of 1. The result of this segmentation is then solely the audio of the movie.

4.4 Segmentation Combination

As established in the formal model, filter functions can be combined using Boolean operations, which correspond to set operations between the segments. The intersection, in particular, is inherently already defined through the proposed URI scheme. All other operations, such as complement, union, or set difference, exist between two segments but would require additional care and investigation. An important fact for the intersection is that the result of segmentation on a media object is another valid media object. This means that a second segmentation can be applied. As shown, this consecutive application of two filter segmentations corresponds to the intersection of the segments, where the order in which the two are listed is irrelevant. Such a URI of arbitrarily many nested segmentations then has the form

```
/<segment type 1>/<segment description 1>/.../<segment type n>/<segment description n>
```

This allows two considerations. First, it creates the possibility of describing a high-dimensional segmentation in terms of a combination of lower-dimensional segmentations. For instance, a spatio-temporal segmentation of a square of a video snippet

```
/rotoscope/200,rect,0,100,0,100;600,rect,0,100,0,100
```

is equivalent to segmenting first by time and then by space or vice versa

```
/time/200-600/rect/0,100,0,100
```

This only works if the dimensions are not disjoint, as, in this example, the rectangle is independent of time and can thus be taken apart.

Second, two filter segmentations can operate on the same dimensions. From a theoretical perspective, this combines the filter functions using logical conjunctions, as described previously. In practice, however, one must be cautious, as a segmentation can introduce a linear translation of the data point coordinates. This is due to neutral data points outside of the segment boundaries being trimmed. For illustration, consider two segmentations that extract a rectangular shape from a 700×400 image, as shown in Figure 17. The first segmentation changes the image dimensions to 240×325, and the lower left corner moves from $(100, 35)$ to $(0, 0)$. The second segmentation must thus be linearly translated with respect to the first one. Only then can the two URIs be concatenated to result in the correct intersection segment.

```
/rect/100,340,35,360

/rect/165,500,10,315


/rect/100,340,35,360/rect/65,400,-25,280
```

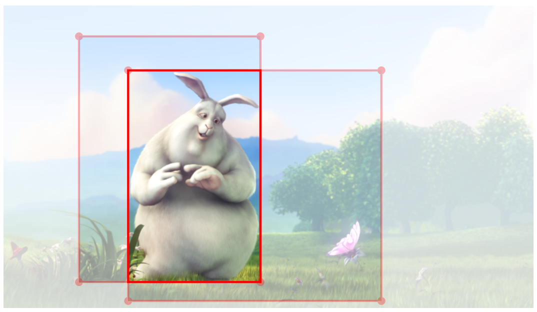

Figure 17 Intersection of two rectangular segmentations of a 700×400 screenshot from Big Buck Bunny [8]. On the left, combining the first two segmentations yields the third, which is adapted to the change in the image size. The corresponding visualization is on the right.

Reduction segmentation follows a similar setup. Here, the concatenation of two segment URIs is the intersection of the dimensions of the data point descriptions. Unlike filter segmentations, reduction segmentations can be combined without any transformations necessary. This is due to the elements to segment being dimensions that have no order or offset. Therefore, changing them does not change the boundaries of the media object. For instance, consider two image segments `/color/red,green` and `/color/green,blue` which, when combined, produce an image with only green-scale colors. Likewise, filter and reduction segmentations can be combined. As the two operate on fully separate layers, they are again independent, can be combined both ways and do not need any transformations.

5 Unified Multimedia Segmentation in Knowledge Graphs

While the segmentation mechanisms described above can be used for general-purpose media access analogously to IIIF, they was primarily designed with knowledge organization tasks in mind. Specifically, it can be effectively used in knowledge graphs in order to compactly and efficiently refer to a concept that is better represented by a specific part of a multimedia document rather than the document as a whole. To illustrate the utility of our proposed scheme, consider the example shown in Figure 18, which shows a rectangular region of an image being annotated with a label.

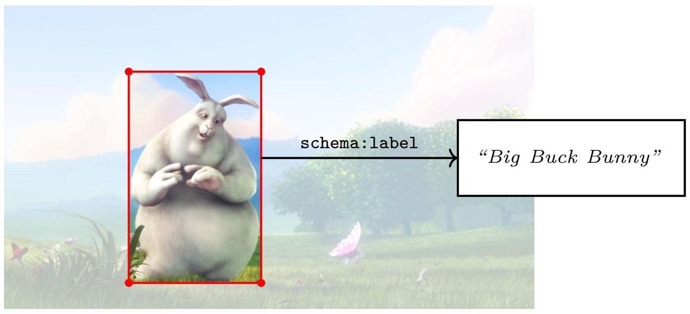

Figure 18 Rectangular segmentation of a larger screenshot from Big Buck Bunny [8] annotated with the label "Big Buck Bunny".

The information depicted in Figure 18 can be represented in different ways using established graph models. Listing 1 shows an example of how a representation using traditional RDF methods could look like. In order to capture the fact that a specific image region is supposed to be annotated, the example introduces an extra graph entity `<http://example.org/bunny>` and two relation types; `<image:in>` and `<image:rectRegion>`, representing the relationship of an image region and the type of region, respectively. Since existing RDF graph stores have no notion of multimedia documents or the information contained therein, the representation requires three triples together with extra relations, the semantics of which need to be known by a client.

Listing 1 Classical RDF graph example in N-Triples notation.

```
<http://example.org/bunny> <image:in> <https://peach.blender.org/wp-content/uploads/bbb-splash.png> .
<http://example.org/bunny> <image:rectRegion> "165,340,35,315" .
<http://example.org/bunny> <schema:label> "Big Buck Bunny" .
```

The same information could also be represented using a property graph. Listing 2 shows a possible structure using the Property Graph Exchange Format [14]. This example uses two nodes, one representing the whole image and one for the specific region. The latter has then both the label as well as the details about how the boundaries of the segmentation are defined as properties. An alternative representation would have the segmentation boundaries as properties of the relation, rather than the entity. The expressiveness would be equivalent in this case.

Listing 2 Property graph example in PGEF notation.

```
101 :Image src:"https://peach.blender.org/wp-content/uploads/bbb-splash.png"
102 :Segment type:rect coords:"165,340,35,315" label:"Big Buck Bunny"

102 -> 101 :segments
```

When using a graph store that is aware of the types of media documents and supports our proposed segmentation scheme, the same information can be represented more succinctly. Listing 3 shows an RDF N-Triple example that compacts the information previously represented in three triples into just one. This example assumes that the graph store listens to `example.org` and is also directly storing the media document and implementing the segmentation. A practical implementation of such a graph store is outlined in Section 6.

Listing 3 RDF graph example in N-Triples notation with support for unified multimedia segmentation scheme.

```
<http://example.org/bbb-splash/segment/rect/165,340,35,315> <schema:label> "Big Buck Bunny" .
```

RDF is especially suitable for our proposed scheme since it inherently uses URLs to represent nodes in the graph. Addressing segments of known documents, rather than the document as a whole, therefore, does not require the introduction of placeholder nodes with no inherent meaning as in Listing 1. Instead, both `<http://example.org/bbb-splash/segment/rect/165,340,35,315>` and `<http://example.org/bbb-splash>` in the context of Listing 3 would be valid URLs that directly return the expected content. An additional benefit of a graph store implementing our scheme directly is, that it would be aware that these two graph entities are related, specifically that the former is a segment of the latter. This information could then be exploited when querying the graph without the need to explicitly materialize such relations.

While our segmentation scheme is especially suited to RDF graph representations, it is by no means limited to it. Listing 4 exemplifies the use of the scheme in a property graph, which enables the representation of the entire relevant information content in a single node without the need for any relations.

Listing 4 Property graph example in PGEF notation using the unified multimedia segmentation scheme.

```
101 :Image src:"http://example.org/bbb-splash/segment/rect/165,340,35,315" label:"Big Buck Bunny"
```

6 Implementation

Based on the theoretical considerations of the previous sections, the Unified Segmentation Model is implemented in the MediaGraph Store (MeGraS) codebase,[6] a storage and querying engine for multimodal data written in Kotlin. MeGraS is a graph store that follows the MediaGraph concept, which treats media components of multimodal knowledge graphs not as a collection of opaque documents external to the graph but as parts of the graph itself. The goal of the implementation is to showcase the soundness of the formal model. To that end, the schemes established in the previous section serve as the basis. The scope of the implementation is set to cover the most important document types and types of segmentation. We also implement a test suite[7] for all segmentations supported by MeGraS, which serves not only as validation of our implementation but can also serve to validate other implementation of our proposed scheme.

The MediaGraph Store is constructed as a web server, exposing a REST API where users can add media and metadata and run queries on it. Metadata, including references to the corresponding files on the hard drive, are stored as RDF triples in a database.

Adding Multimedia Objects

Upon addition via an HTTP-POST request, each media object is stored in its raw format and a transcoded canonical format. The server then returns a canonical ID. With that generated canonical ID, the media object is then retrievable under the URI of the form

```
<server address>/<canonical id>
```

[6] https://github.com/lucaro/MeGraS
[7] https://github.com/lucaro/Unified-Media-Segmentation-Test-Suite

Accessing Media Segments

The implementation provided in this paper now adds functionality to additionally request segments of a media object using a URI of the form

```
<server address>/<canonical id>/segment/<segment type>/<segment description>
```

which, after processing, generates a segment. Segment type and description largely correspond to the syntax from Section 4. Because segments remain legitimate media objects of the same type as the original, segmentations can be arbitrarily nested. One can even directly request multiple segmentations in one request URI. These are then processed one by one.

The system follows a lazy-loading approach and caches all media, meaning that a certain segmentation is performed only the first time it is requested. Repeated requests then yield the cached result directly. For that, a canonical segment ID is generated from the segmentation type and description, with which a certain segmentation is uniquely identified and requested using a URI of the form

```
<server address>/<canonical id>/c/<segment id>
```

This is the only form of URIs that will directly return the actual content of the segment. All other requests containing a segment definition, as described above, will be re-directed to the representation containing the segment ID. This mechanism also allows the combination of equivalent segment definitions into the same segment with the same ID. The equivalence of different ways of referring to the same segment is also reflected in the underlying graph structure by inserting a triple using the `sameAs` relation. An additional advantage of this representation is its compactness compared to possibly very verbose segment definitions when it is used as part of graph queries.

7 Discussion

The goal of the Unified Segmentation Model was to generalize the process of segmentation and describe it in mathematical terms. For that, a multimedia object and the segmentation operation on arbitrary media objects were defined. Media objects were constructed as sets of data points whose properties are separated into coordinate dimensions and description dimensions. This separation proved to be valuable, as segmentation can target either of the two. Consequently, the segmentation operation could be defined as either a filter or a reduction, where a filter selects a subset of coordinates, and a reduction retains only a part of the description dimensions. The strength of this setup is its generality, as the model can portray all conceivable types of multimedia documents and segmentations of them. Additionally, its mathematical core allows deductions using set theory and Boolean algebra. This breadth of functionality, based on a proper formal foundation, enables our model to fulfill all criteria outlined in Table 1.

In literature, only the General Fragment Model (GFM) [23, 24] tries to take a similarly general perspective. All other accounts are more of a means to an end, as they are application-oriented and focus on concrete use cases. Unlike the Unified Segmentation Model presented here, the GFM is constructed as an ontology to systemize the definition of segments. Nonetheless, parallels can be observed and are summarized in Table 2. In the GFM, an indexer function maps tokens to fragments of the information artifact. This corresponds to a segmentation function that creates segments of the multimedia object. Concrete filter or reduction functions can be seen as tokens. Applying them to media objects corresponds to the GFM's notion of anchors. The token set of the GFM defines the set of possible tokens, which is the set of valid and complete filter and reduction functions. This shows that the proposed model can be represented in terms of the GFM. While the GFM is a powerful abstract formalization of segmentation description, our model far exceeds it by providing a formal model that adds concrete instantiations that allow the investigation of properties and the extraction of characteristics.

Table 2 Comparison of core concepts of the General Fragment Model with the proposed Unified Segmentation Model.

General Fragment Model	Unified Segmentation Model
information artifact	multimedia object
indexer function	segmentation function
tuple token	filter or reduction function
token set	set of all valid and complete functions
anchor	segmentation function applied to multimedia object

A vital part of the formal model is the reflection on how filter and reduction function internals can be described. Again, parallels to the GFM's indexer taxonomy are visible. Unlike the GFM, though, it paves the way toward implementation. It can be observed that the rules by which a segmentation operates are stateable in different fashions depending on the use case and can be translated into each other. This addresses some of the model's difficulties of its generality and creates a solid foundation for the implementation.

Before implementation was possible, the Unified Segmentation Model needed to be given a more concrete shape to prepare for implementation by closely examining the specific segmentation function types. The choice of a URI-based approach was never a question, as metadata-based segment description adds an indirection and thus defeats the purpose of the whole project: making media first-class citizens in knowledge graphs. The segmentation types and categories were created based on the investigation of how segmentation rules can be described. Surely, not all proposed segmentations are equally important for real-life applications. Space-filling curves, for instance, are a fascinating mental model to establish a locality-preserving order, but not something one would regularly create segments with. At the same time, URI serialization of the segment descriptions was investigated. The main challenge of this part is the right balance between generality and implementable concreteness. The resulting segmentation types and serializations should be capable of covering all desirable segmentations with little adaptations.

An issue introduced by this concretization is rigidity. To textually describe multi-dimensional segments and to do it uniquely requires a certain level of regulation. However, determining the best representation is not always trivial, as this can depend on the context. This factor can take away intuitiveness, as different users would represent a particular segmentation differently. To use the proposed URI schemes to describe segmentations, one has to use the right segment type keywords[8] and closely follow the segment description syntax.

This is comparable to URI-based segment annotation approaches from the literature, where the same necessary evil can be found many times. Most notably, existing approaches suffer at least one of two issues: weak segmentation coverage or ambiguity and complexity. For instance, the Media Fragments URI specification only supports limited segmentations but is very expressive, whereas the MPEG-21 standard is more powerful but overly ambiguous and unintuitive. This work tries to absorb the best of both worlds and combat both problems simultaneously. Against the first, the proposed schemes cover all conceivable segmentations, or at least the ones realistically used in practice. Adding more variations should be simple, though, as the same pattern can be continued. For the second, efforts were made to keep the URIs as concise and expressive as possible, such that the human eye can also understand them to a certain degree. Unlike almost all

[8] To standardize the kinds of segment types permissible, one could have a standard with a list of suggested/supported keywords or an "ontology"/vocabulary, which may include mathematical specifications that may/or may not be interpreted on the fly

standards, this work fully refrains from using URI fragments because they are intended to identify secondary resources or views instead of representing a segment as its own resource in relation to its parent. Additionally, a practical reason against fragments is that they are typically interpreted by the client application, which requires client support. In a case like the Media Fragments URI specification, clients must either segment locally or translate URI fragments and send requests for partial content, neither of which is fully implemented in standard browsers. URI queries are equally not desired, as they create new resources without a relation to the original. Instead, URI paths are employed, corresponding to how knowledge graph nodes are usually represented. IIIF served as a model here, as its image processing is purely server-side, and images are fetched as path URIs using standard HTTP requests. Additionally, IIIF is the only approach that combines Semantic Web technology with multimedia on a technical level, as it defines a consistent URI scheme for accessing full and partial image content. Unfortunately, IIIF's flexibility to describe segments is minimal, and only images are supported, despite various efforts to extend to more media types.

8 Conclusions

This work was guided by the question of how to universally represent segments of multimedia documents for use in knowledge graphs. While the annotation of partial media content, and thus the need for a textual segment representation, is nothing new to the literature and Semantic Web applications, all theoretical and practical solutions have certain shortcomings. Metadata-based annotations use a secondary document to define segments of the main document, thus introducing an indirection. URI-based approaches are either very limited in their capabilities or are too complex to use. Further, URI fragments are most often used, which are interpreted client-side and thus require special client software. Only IIIF, through its image API, provides a standard HTTP interface for image segment retrieval. Unfortunately, available solutions are limited in the range of supported multimedia documents and segmentation types. Furthermore, only the General Fragment Model tries to generalize the conceptual segmentation model but lacks expressiveness.

To fill the gap in the literature, this paper proposes a Unified Segmentation Model, which can express an arbitrary segmentation on any multimedia document and comprises both a formal model and an applied URI scheme. The former mathematically describes multimedia objects and investigates different types of segmentation functions, which allows the inference of set-theoretical properties. It is then deduced by which rules concrete segmentation functions select partial content and how these rules can be described. In contrast to the many approaches found in literature, this theoretical foundation is defined independently of any specific existing media type. Based on these investigations, a URI scheme is presented that covers all types of segmentations and proposes serializations of segmentations typically found in research and real life. Due to its content-agnostic theoretical foundation, this scheme is not tightly bound to existing media formats and can easily be extended to future information representations. This might even include currently not widely used data types and modalities, such as, for example, holographic information or smells. Our proposed URI scheme is then implemented in the MediaGraph Store, a storage and querying engine for multimedia knowledge graphs. The implementation encompasses a representative set of segmentations and supports various media, namely text, audio, image, video, PDF, and 3D mesh.

In the future, the Unified Segmentation Model can certainly allow for more deductions and considerations. For instance, only the intersection of two segments is inherently supported, as it corresponds to the concatenation of their URIs. Here, the formal model could be leveraged further to investigate other set operations, increasing the expressive power of media segments in knowledge graphs. Additionally, future work could delve into properties that efficiently determine equivalence, containment, and overlap, as the current deductions are too basic to be useable in practice. Some segment serializations tend to become very large, which is cumbersome, hinders performance, and ultimately can lead to HTTP errors. Therefore, more thought should go into alternative encodings of these segments to increase compaction without sacrificing segment quality. Beyond the sole representation of segments, future work should finally address the querying, discovery, and retrieval of media segments.

But even in the light of these limitations, the Unified Segmentation Model and its associated URI scheme seem to provide a crucial element to advance the state of multimedia data usage in knowledge graphs by breaking the atomicity of multimedia content in a theoretically-founded, media-type agnostic approach and on the Web and a stepping stone for extensions that can finally turn multimedia data into a first-class citizen in knowledge graphs and the Web of Data.

References

1 3D Systems, Inc. Stereolithography interface specification, October 1989.

2 Richard Arndt, Raphaël Troncy, Steffen Staab, Lynda Hardman, and Miroslav Vacura. COMM: Designing a well-founded multimedia ontology for the web. In *The Semantic Web*, pages 30–43. Springer, 2007. `doi:10.1007/978-3-540-76298-0_3`.

3 Ana B. Benitez, Seungyup Paek, Shih-Fu Chang, Atul Puri, Qian Huang, John R. Smith, Chung-Sheng Li, Lawrence D Bergman, and Charles N. Judice. Object-based multimedia content description schemes and applications for MPEG-7. *Signal Processing: Image Communication*, 16(1-2):235–269, September 2000. `doi:10.1016/s0923-5965(00)00030-8`.

4 Tim Berners-Lee. Linked data, July 2006. (Retrieved 2023-07-20). URL: `https://www.w3.org/DesignIssues/LinkedData`.

5 Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform resource identifiers (URI): generic syntax. *RFC*, 2396(2396):1–40, August 1998. `doi:10.17487/RFC2396`.

6 Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform resource identifier (URI): generic syntax. *RFC*, 3986(3986):1–61, January 2005. `doi:10.17487/RFC3986`.

7 Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

8 Blender Foundation. Big buck bunny, 2008. (Retrieved 2023-05-24). `doi:10.1145/1504271.1504321`.

9 Stephan Bloehdorn, Siegfried Handschuh, Steffen Staab, Yannis Avrithis, Yiannis Kompatsiaris, Vassilis Tzouvaras, Kosmas Petridis, Nikos Simou, and Michael G. Strintzis. Knowledge representation for semantic multimedia content analysis and reasoning. In *European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology (EWIMT)*, November 2004.

10 Jon Bosak and Tim Bray. Xml and the second-generation web. *Scientific American*, 280(5):89–93, 1999.

11 Ian S. Burnett, Fernando Pereira, Rik Van de Walle, and Rob Koenen. *The MPEG-21 Book*. Wiley, 2006.

12 Nick Burris and David Bokan. Text fragments. W3C community group draft report, W3C, December 2022. URL: `https://wicg.github.io/scroll-to-text-fragment/`.

13 Jeremy Carroll and Graham Klyne. Resource description framework (RDF): Concepts and abstract syntax. W3C recommendation, W3C, February 2004. URL: `https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.

14 Hirokazu Chiba, Ryota Yamanaka, and Shota Matsumoto. Property graph exchange format. *CoRR*, abs/1907.03936, 2019. `arXiv:1907.03936`, `doi:10.48550/arXiv.1907.03936`.

15 Tom Crane. Wellcome library ixif "interim" implementation, 2015. (Retrieved 2023-02-27). URL: `https://gist.github.com/tomcrane/7f86ac08d3b009c8af7c`.

16 Richard Cyganiak and Leo Sauermann. Cool URIs for the semantic web. W3C note, W3C, December 2008.

17 Mike Dean and Guus Schreiber. OWL web ontology language reference. W3c recommendation, W3C, February 2004. URL: `https://www.w3.org/TR/2004/REC-owl-ref-20040210/`.

18 Steven DeRose, Eve Maler, and Ron Daniel. XPointer xpointer() scheme. W3C working draft, W3C, December 2002. URL: `https://www.w3.org/TR/2002/WD-xptr-xpointer-20021219/`.

19 Davy Van Deursen, Wim Van Lancker, Erik Mannens, and Rik Van de Walle. NinSuna: A server-side w3c media fragments implementation. In *2010 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, July 2010. `doi:10.1109/icme.2010.5583192`.

20 Davy Van Deursen, Wim Van Lancker, Wesley De Neve, Tom Paridaens, Erik Mannens, and Rik Van de Walle. NinSuna: a fully integrated platform for format-independent multimedia content adaptation and delivery using semantic web technologies. *Multimedia Tools and Applications*, 46(2-3):371–398, September 2009. `doi:10.1007/s11042-009-0354-0`.

21 Davy Van Deursen, Raphaël Troncy, Erik Mannens, and Silvia Pfeiffer. Protocol for media fragments 1.0 resolution in HTTP. W3C working draft, W3C, December

2011. URL: https://www.w3.org/TR/2011/WD-media-frags-recipes-20111201/.

22 Davy Van Deursen, Raphaël Troncy, Erik Mannens, Silvia Pfeiffer, Yves Lafon, and Rik Van de Walle. Implementing the media fragments URI specification. In *International Conference on World Wide Web*. ACM, April 2010. doi:10.1145/1772690.1772931.

23 Sandro Rama Fiorini, Wallas Henrique Sousa dos Santos, Rodrigo Costa Mesquita Santos, Guilherme Augusto Ferreira Lima, and Márcio Ferreira Moreno. General fragment model for information artifacts. *CoRR*, abs/1909.04117, 2019. arXiv:1909.04117, doi:10.48550/arXiv.1909.04117.

24 Sandro Rama Fiorini, Guilherme Ferreira Lima, and Marcio F. Moreno. Demo: Tools for information fragmentation in knowledge graphs*. *CEUR Workshop Proceedings*, 2980:1–5, 2021. URL: https://ceur-ws.org/Vol-2980/paper377.pdf.

25 Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. Sweetening ontologies with DOLCE. In *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 166–181. Springer, 2002. doi:10.1007/3-540-45810-7_18.

26 Roberto García and Òscar Celma. Semantic integration and retrieval of multimedia metadata. In Siegfried Handschuh, Thierry Declerck, and Marja-Riitta Koivunen, editors, *Proceedings of the 5th International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2005) located at the 4rd International Semantic Web Conference ISWC 2005, 7th November 2005, Galway, Ireland*, volume 185 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005. URL: https://ceur-ws.org/Vol-185/semAnnot05-07.pdf.

27 Paul Grosso, Eve Maler, Jonathan Marsh, and Norman Walsh. XPointer element() scheme. W3C recommendation, W3C, March 2003. URL: https://www.w3.org/TR/2003/REC-xptr-element-20030325/.

28 Paul Grosso, Eve Maler, Jonathan Marsh, and Norman Walsh. XPointer framework. W3C recommendation, W3C, March 2003. URL: https://www.w3.org/TR/2003/REC-xptr-framework-20030325/.

29 Ramanathan Guha and Dan Brickley. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C, February 2004. URL: https://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

30 Michael Hausenblas, Raphaël Troncy, Tobias Bürger, and Yves Raimond. Interlinking multimedia: How to apply linked data principles to multimedia fragments. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Kingsley Idehen, editors, *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009. URL: https://ceur-ws.org/Vol-538/ldow2009_paper17.pdf.

31 Michael Hausenblas, Erik Wilde, and Jeni Tennison. URI fragment identifiers for the text/csv media type. *RFC*, 7111(7111):1–13, January 2014. doi:10.17487/RFC7111.

32 Ian Hickson, Robin Berjon, Steve Faulkner, Travis Leithead, Erika Doyle Navara, Theresa O'Connor, and Silvia Pfeiffer. HTML5. W3C recommendation, W3C, October 2014. URL: https://www.w3.org/TR/2014/REC-html5-20141028/.

33 Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. *ACM Comput. Surv.*, 54(4), July 2021. doi:10.1145/3447772.

34 Philipp Hoschka. Synchronized multimedia integration language (SMIL) 1.0 specification. W3C recommendation, W3C, June 1998. URL: https://www.w3.org/TR/1998/REC-smil-19980615/.

35 Jane Hunter. Adding multimedia to the semantic web: Building an MPEG-7 ontology. In Isabel F. Cruz, Stefan Decker, Jérôme Euzenat, and Deborah L. McGuinness, editors, *Proceedings of SWWS'01, The first Semantic Web Working Symposium, Stanford University, California, USA, July 30 - August 1, 2001*, pages 261–283. CEUR-WS.org, 2001. URL: http://www.semanticweb.org/SWWS/program/full/paper59.pdf, doi:10.1002/0470012617.ch3.

36 Antoine Isaac and Raphaël Troncy. Designing and Using an Audio-Visual Description Core Ontology. *Workshop on Core Ontologies in Ontology Engineering*, 118, 2004.

37 ISO/IEC 15938. Multimedia Content Description Interface (MPEG-7). Standard, Moving Picture Experts Group, 2001.

38 ISO/IEC 21000. Multimedia Framework (MPEG-21). Standard, Moving Picture Experts Group, 2002.

39 ISO/IEC 21000-17. Multimedia framework (mpeg-21) – part 17: Fragment identification of mpeg resources. Standard, Moving Picture Experts Group, 2006.

40 Dean Jackson. Scalable vector graphics (SVG): the world wide web consortium's recommendation for high quality web graphics. In Tom Appolloni, editor, *Proceedings of the 29th International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2002, San Antonio, Texas, USA, July 21-26, 2002, Abstracts and Applications*, page 319. ACM, January 2002. doi:10.1145/1242073.1242327.

41 Rob Koenen and Fernando Pereira. MPEG-7: The Generic Multimedia Content Description Standard, Part 1. *IEEE Multimedia*, 9(2):78–87, 2002. doi:10.1109/93.998074.

42 Thomas Kurz, Georg Güntner, Violeta Damjanovic, Sebastian Schaffert, and Manuel Fernandez. Semantic enhancement for media asset management systems - integrating the red bull content pool in the web of data. *Multim. Tools Appl.*, 70(2):949–975, August 2014. doi:10.1007/s11042-012-1197-7.

43 Thomas Kurz and Harald Kosch. Lifting media fragment uris to the next level. In *European Semantic Web Conference ESWC, Interna-*

tional Workshop on Linked Media, volume 1615. CEUR-WS.org, 2016. URL: `https://ceur-ws.org/Vol-1615/limePaper3.pdf`.

44 Thomas Kurz, Sebastian Schaffert, Kai Schlegel, Florian Stegmaier, and Harald Kosch. SPARQL-MM - extending SPARQL to media fragments. In *Lecture Notes in Computer Science*, pages 236–240. Springer International Publishing, 2014. `doi:10.1007/978-3-319-11955-7_26`.

45 Carl Lagoze and Jane Hunter. The ABC ontology and model. In Keizo Oyama and Hironobu Gotoda, editors, *2001 International Conference on Dublin Core and Metadata Applications*, pages 160–176. Dublin Core Metadata Initiative, 2001. URL: `http://dcpapers.dublincore.org/pubs/article/view/655`.

46 David H. Laidlaw, W. Benjamin Trumbore, and John F. Hughes. Constructive solid geometry for polyhedral objects. In *1986 Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. ACM, 1986. `doi:10.1145/15922.15904`.

47 Wim Van Lancker, Davy Van Deursen, Erik Mannens, and Rik Van de Walle. HTTP adaptive streaming with media fragment URIs. In *2011 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, July 2011. `doi:10.1109/icme.2011.6012149`.

48 Wim Van Lancker, Davy Van Deursen, Erik Mannens, and Rik Van de Walle. Implementation strategies for efficient media fragment retrieval. *Multimedia Tools and Applications*, 57(2):243–267, March 2011. `doi:10.1007/s11042-011-0785-2`.

49 Jim Ley. Thoughts on an image description vocabulary, August 2003. (Retrieved 2023-06-05). URL: `http://jibbering.com/discussion/image.html`.

50 Library of Congress. Streaming services: An audio and video (a/v) delivery api for the library of congress. (Retrieved 2023-03-01). URL: `https://www.loc.gov/apis/micro-services/streaming-services/`.

51 Gene Loh. Linked data and IIIF: Integrating taxonomy management with image annotation. In *2017 Pacific Neighborhood Consortium Annual Conference and Joint Meetings (PNC)*. IEEE, November 2017. `doi:10.23919/pnc.2017.8203521`.

52 Erik Mannens, Davy Van Deursen, Raphaël Troncy, Silvia Pfeiffer, Conrad Parker, Yves Lafon, Jack Jansen, Michael Hausenblas, and Rik Van De Walle. A URI-based approach for addressing fragments of media resources on the web. *Multimedia Tools and Applications*, 59(2):691–715, 2012. `doi:10.1007/s11042-010-0683-z`.

53 Larry Masinter. The "data" URL scheme. *RFC*, 2397(2397):1–5, August 1998. `doi:10.17487/RFC2397`.

54 James D. Murray and William vanRyper. *Encyclopedia of graphics file formats, 2nd Edition*. Springer, 2 edition, May 1996. URL: `http://oreilly.com/catalog/9781565921610`.

55 Klinsukon Nimkanjana and Suntorn Witosurapot. A simple approach for enabling sparql-based temporal queries for media fragments. In Kamal Zuhairi Zamli, Vitaliy Mezhuyev, and Luigi Benedicenti, editors, *Proceedings of the 7th International Conference on Software and Computer Applications, ICSCA 2018, Kuantan, Malaysia, February 08-10, 2018*, pages 212–216. ACM, 2018. `doi:10.1145/3185089.3185126`.

56 Lyndon J. B. Nixon, Matthias Bauer, Cristian Bara, Thomas Kurz, and John Pereira. Connectme: Semantic tools for enriching online video with web content. In *2012 I-SEMANTICS Posters & Demonstrations Track*, volume 932 of *CEUR Workshop Proceedings*, pages 55–62. CEUR-WS.org, 2012. URL: `https://ceur-ws.org/Vol-932/paper11.pdf`.

57 Silvia Pfeiffer, Conrad Parker, and Claudia Schremmer. Annodex: a simple architecture to enable hyperlinking, search & retrieval of time–continuous data on the web. In Nicu Sebe, Michael S. Lew, and Chabane Djeraba, editors, *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR 2003, November 7, 2003, Berkeley, CA, USA*, pages 87–93. ACM, 2003. `doi:10.1145/973264.973279`.

58 Silvia Pfeiffer and Craig Parker. Specifying time intervals in URI queries and fragments of time-based Web resources. Internet-Draft draft-pfeiffer-temporal-fragments-03, Internet Engineering Task Force, 2005. URL: `https://datatracker.ietf.org/doc/draft-pfeiffer-temporal-fragments/03/`.

59 Julien A. Raemy, Peter Fornaro, and Lukas Rosenthaler. Implementing a video framework based on IIIF: A customized approach from long-term preservation video formats to conversion on demand. *Archiving Conference*, 14(1):68–73, May 2017. `doi:10.2352/issn.2168-3204.2017.1.0.68`.

60 A. A. G. Requicha and H. B. Voelcker. Constructive solid geometry. Technical report, production automation project tm-25, University of Rochester, November 1977.

61 Lukas Rosenthaler, Peter Fornaro, Andrea Bianco, and Benjamin Geer. Simple image presentation interface (SIPI) – an IIIF-based image-server. *Archiving Conference*, 14:28–33, May 2017. `doi:10.2352/issn.2168-3204.2017.1.0.28`.

62 Luca Rossetto. lucaro/Unified-Media-Segmentation-Test-Suite. Audiovisual (visited on 2024-12-12). URL: `https://github.com/lucaro/Unified-Media-Segmentation-Test-Suite`, `doi:10.4230/artifacts.22623`.

63 Luca Rossetto and Jan Willi. lucaro/MeGraS. Software, swhId: `swh:1:dir:d2dfb84bd390d27ad96190f973dfa484b9e68bc0` (visited on 2024-12-12). URL: `https://github.com/lucaro/MeGraS`, `doi:10.4230/artifacts.22622`.

64 Carsten Saathoff and Ansgar Scherp. M3O: The multimedia metadata ontology. In *2009 Workshop on Semantic Multimedia Database Technologies, 10th International Workshop of the Multimedia Metadata Community (SeMuDaTe)*, volume 539, pages 4–15. CEUR-WS.org, 2009.

65 Philippe Salembier and John R. Smith. Mpeg-7 multimedia description schemes. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):748–759, 2001. `doi:10.1109/76.927435`.

66 Elena Sánchez-Nielsen, Francisco Chávez-Gutiérrez, Javier Lorenzo-Navarro, and Modesto Castrillón-Santana. A multimedia system to produce and deliver video fragments on demand

on parliamentary websites. *Multimedia Tools and Applications*, 76(5):6281–6307, February 2016. doi:10.1007/s11042-016-3306-5.

67 Felix Sasaki, Tobias Bürger, Wonsuk Lee, Florian Stegmaier, Joakim Söderberg, Jean-Pierre EVAIN, Werner Bailer, Thierry Michel, John Strassner, Véronique Malaisé, and Pierre-Antoine Champin. Ontology for media resources 1.0. W3C recommendation, W3C, February 2012. URL: https://www.w3.org/TR/2012/REC-mediaont-10-20120209/.

68 Andy Seaborne and Eric Prud'hommeaux. SPARQL query language for RDF. W3C recommendation, W3C, January 2008. URL: https://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

69 Ching-Kuang Shene. B-spline curves: Closed curves, 2011. (Retrieved 2023-05-24). URL: https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/bspline-curve-closed.html.

70 Tomo Sjekavica, Ines Obradović, and Gordan Gledec. Ontologies for Multimedia Annotation: An overview. In *2013 European Conference of Computer Science (ECCS)*, pages 123–129, 2013.

71 Stuart Snydman, Robert Sanderson, and Tom Cramer. The international image interoperability framework (iiif): A community & technology approach for web-based images. *Archiving Conference*, 12(1):16–16, 2015. doi:10.2352/issn.2168-3204.2015.12.1.art00005.

72 Peter Sorotokin, Garth Conboy, Brady Duga, John Rivlin, Don Beaver, Kevin Ballard, Alastair Fettes, and Daniel Weck. Epub canonical fragment identifiers 1.1. Recommended specification, IDPF, 2017. URL: https://idpf.org/epub/linking/cfi/.

73 Raphaël Troncy, Òscar Celma, Suzanne Little, Roberto García, and Chrisa Tsinaraki. MPEG-7 based multimedia ontologies: Interoperability support or interoperability issue. In *International Workshop on Multimedia Annotation and Retrieval enabled by Shared Ontologies (MARESO)*, pages 2–15, 2007.

74 Raphaël Troncy, Lynda Hardman, and Jacco Van Ossenbruggen. Identifying Spatial and Temporal Media Fragments on the Web. In *W3C Video on the Web Workshop*, pages 4–9, 2007.

75 Raphaël Troncy, Erik Mannens, Silvia Pfeiffer, and Davy Van Deursen. Media fragments URI 1.0 (basic). W3C recommendation, W3C, September 2012. URL: https://www.w3.org/TR/2012/REC-media-frags-20120925/.

76 Chrisa Tsinaraki, Panagiotis Polydoros, and Stavros Christodoulakis. Interoperability support for ontology-based video retrieval applications. In *Lecture Notes in Computer Science*, pages 582–591. Springer, Berlin, Heidelberg, 2004. doi:10.1007/978-3-540-27814-6_68.

77 Greg Turk. The ply polygon file format, 1994. (Retrieved 2023-05-28). URL: http://gamma.cs.unc.edu/POWERPLANT/papers/ply.pdf.

78 Erik Wilde and Marcel Baschnagel. Fragment identifiers for plain text files. In *ACM Conference on Hypertext and Hypermedia*. ACM, September 2005. doi:10.1145/1083356.1083398.

79 Erik Wilde and Martin J. Dürst. URI fragment identifiers for the text/plain media type. *RFC*, 5147(5147):1–17, April 2008. doi:10.17487/RFC5147.

80 Ting Wu, Zhuoming Xu, Lixian Ni, Yuanhang Zhuang, Junhua Wang, and Qin Yan. Towards a media fragment URI aware user agent. In *2014 Web Information System and Application Conference (WISA)*, pages 37–42. IEEE, September 2014. doi:10.1109/wisa.2014.15.