

Article

AI-Based Semantic Multimedia Indexing and Retrieval for Social Media on Smartphones

Stefan Wagenpfeil ^{1,*}, Felix Engel ^{1,†}, Paul Mc Kevitt ^{2,†} and Matthias Hemmje ^{1,†}

¹ Faculty of Mathematics and Computer Science, University of Hagen, Universitätsstrasse 1, D-58097 Hagen, Germany; felix.engel@fernuni-hagen.de (F.E.); Matthias.hemmje@fernuni-hagen.de (M.H.)
² Academy for International Science & Research (AISR), Derry/Londonderry, Ireland; p.mckevitt@aisr.org.uk
 * Correspondence: stefan.wagenpfeil@fernuni-hagen.de
 † This paper is an extended version of our paper published in SMAP 2020.

Abstract: To cope with the growing number of multimedia assets on smartphones and social media, an integrated approach for semantic indexing and retrieval is required. Here, we introduce a generic framework to fuse existing image and video analysis tools and algorithms into a unified semantic annotation, indexing and retrieval model resulting in a multimedia feature vector graph representing various levels of media content, media structures and media features. Utilizing artificial intelligence (AI) and machine learning (ML), these feature representations can provide accurate semantic indexing and retrieval. Here, we provide an overview of the generic multimedia analysis framework (GMAF) and the definition of a multimedia feature vector graph framework (MMFVGF). We also introduce AI4MMRA to detect differences, enhance semantics and refine weights in the feature vector graph. To address particular requirements on smartphones, we introduce an algorithm for fast indexing and retrieval of graph structures. Experiments to prove efficiency, effectiveness and quality of the algorithm are included. All in all, we describe a solution for highly flexible semantic indexing and retrieval that offers unique potential for applications such as social media or local applications on smartphones.

Keywords: semantic indexing; multimedia retrieval; framework; multimedia feature vector graph; MMFVGF; MMFVGG; AI4MMRA; GMAF; AI; graph encoding; graph code



Citation: Wagenpfeil, S.; Engel, F.; Mc Kevitt, P.; Hemmje, M. AI-Based Semantic Multimedia Indexing and Retrieval for Social Media on Smartphones. *Information* **2021**, *1*, 0. <https://doi.org/>

Received: 15 December 2020
 Accepted: 15 January 2021
 Published:

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction and Motivation

Every year, more than 1.2 trillion digital photos and videos are taken; more than 85% of them taken on smartphones, and this number is still increasing [1]. This volume of media is neither manageable for users nor for content providers and/or platforms such as social media. Capturing pictures readily wherever and whenever you want is unseen in the history of media creation. Additionally, cloud services and storage costs continue to decrease, which makes it simple and affordable for users to store large volumes of media assets on their smartphones [2]. Smartphone vendors provide solutions enabling access to all user media assets directly through the device by transparently up- and downloading them to cloud services [3]. The large volume of multimedia assets provides significant challenges for indexing, querying and retrieval algorithms. Although there is great progress in image analysis, object detection and content analysis [4], there is still potential for improvement and further research to optimize the results of information retrieval algorithms.

Artificial intelligence (AI) and machine learning (ML), including deep learning and pattern recognition in particular, have contributed significantly to research and development in recent years [4–8]. Utilizing these algorithms and methods, products and services that provide a set of functionalities to detect content of images, videos, text or audio have become available. One key example is Google's Vision AI, a service to access a fully trained and equipped neural network for object recognition and basic semantic feature detection in images [9]. Similar APIs are provided by Microsoft [10] and Amazon [11]. All of these

services provide basic object detection, text and pattern recognition and basic semantic feature detection.

Today's digital cameras and smartphones can automatically provide a large set of additional information for each picture. Starting with e.g., Date, Time, GEO-Location, Aperture, Exposure-Time, Lens-Type, Focal-Length, much of this information can enrich images and their metadata representation. The EXIF-standard describing a set of metadata fields is implemented in almost all camera models from smartphone to professional [12]. For videos, MPEG7 provides metadata about videos, shots, video content in a standardized format that is industry-accepted [13]. Semantic knowledge representation also evolved in providing standards, algorithms, data structures and frameworks [6], including products or projects like Google's Knowledge Graph [14] or the Semantic Web [15].

In 2021, we are facing a digital industry with remarkable technology benchmarks that have pushed the boundaries of physics, electricity and manufacturing beyond expectations. For example, Apple's A14 bionic processor in its iPhone 12 has 11.8 billion transistors within one smartphone [16]. The available storage appears to be growing exponentially, [17] reaching an unbelievable 6.8 zettabytes (10^{21} bytes) in 2020 [18]. Additionally, the total availability of information due to higher bandwidths has reached speeds of 1000 GBit/s in internal networks [19] and, starting with the 5G standard, also up to 1 GB/s, even in mobile networks [20]. This foundation enables developers to invent applications that utilize considerable computing power, transfer data in real time, index and retrieve terabytes of data, and allow people to collaborate seamlessly with others throughout the world. Multimedia applications particularly benefit from this infrastructure, as they can utilize larger files with higher resolution and can distribute them over faster network connections.

Summarizing this, a situation exists in which a significant set of technologies, standards and tools is available, providing an opportunity on which to build new structures and algorithms. Here, we provide an overview of a generic framework to integrate the syntax, semantics and metadata of multimedia content into a unified indexing model, facilitating direct retrieval or retrieval as an API for other applications. To address the various levels of multimedia content (technical, content, semantic and intracontent level) [21,22], we introduce a multimedia feature vector graph framework (MMFVGF) that can integrate the various metadata features of multimedia assets into a unified semantic data structure. We also describe an ML-based indexing and retrieval component to refine the MMFVGF and access the resulting multimedia assets, which is specially optimized for processing feature graphs on mobile devices.

2. State of the Art and Related Work

This section provides an initial selection of the most relevant state of the art in science and technology and work related to this paper covering multimedia processing, semantic analysis, multimedia querying and retrieval, graph processing, AI pattern matching and social media. It also identifies how our approach addresses open problems. An earlier instantiation of the ideas in this paper is given in [23].

First, we introduce multimedia processing technologies as a baseline in Section 2.1. Our generic multimedia analysis framework (GMAF) utilizes and integrates as building blocks several of the algorithms and technologies presented in this section. Thus, the functionality of the GMAF will also be introduced here. As our approach aims for a semantic solution, Section 2.2 (Semantic Analysis and Representation) will provide a brief overview and introduce the terms and technologies required. Section 2.3 (Multimedia Indexing, Querying and Retrieval) then provides the background and technologies to generate high-level semantic features of multimedia assets. It also summarizes technologies for querying and retrieval and their remaining challenges, especially when working on graph-based semantic structures, which is covered in Section 2.4 (Graph Representation Processing). To improve graph processing on smartphones, we will introduce an optimized graph encoding algorithm, which requires a basic understanding of the underlying mathematical concepts presented in Section 2.5 (Mathematical Background) and will define a graph

representation that is optimized for use within AI pattern matching systems (Section 2.6). All of these concepts can be applied to social media applications as described in Section 2.7. Section 2.8 aligns our approach to a standard multimedia processing pipeline and defines the scope of this paper.

2.1. Multimedia Processing

In multimedia processing, numerous tools, algorithms and APIs are available. The most common products are already defined in the previous section [9–12,14,24] and emanate from years of research, development and training. Deep learning methods utilize neural networks to provide better accuracy [8]. Furthermore, large sets of annotated training data are available [25]. Additionally, numerous standards have been defined in recent years, including image annotations (such as EXIF [12]) and video metadata (such as MPEG7 [13]). There are still several vendor-specific metadata models, such as Adobe's XMP standard [26] or the production industry standard MXF [27], and all of these can easily be integrated into GMAF by writing a simple plugin and attaching it to the framework (see Figure 5). There is significant progress in detecting objects, relevant regions, the color footprint or even activities or moods of an image or a region. All the technical metadata are embedded in the image or video itself [12,13], and digital assistants such as Siri [28], Cortana [29] and Alexa [30] already provide the basic technology to use natural language for query input. Object and region detection provide promising results [9], and scene detection and automatic extraction of a video's key frames are largely solved [24]. There are also numerous standards for describing general metadata for video [13], and if this is applied recursively to video, any image processing algorithm is applicable to video, depending on the computation time.

2.2. Semantic Analysis and Representation

Vocabularies and ontologies are the basis for semantic representation, which are usually defined in RDF Schema [31]. An overview of the current state of the art is given in [32] and previously in [33]. In our work, we utilize and contribute to these existing standards. For example, spatiotemporal annotations in movies can feed composition relationships (cr, see Section 3.3) into the MMFVG but in return, detected cr information can be written into the asset's corresponding MPEG7 sections [13]. To explore associations between terms and image regions as visual context information, several models have been defined and developed. Examples are the co-occurrence model, the translation model, the continuous relevance model and the multiple Bernoulli relevance model [34]. In addition, with Google's Knowledge Graph project [14], a large collection of semantic relationships, synonyms and metadata is available for public use. For training, classification or automation purposes, several libraries of annotated media are available [25,35], which are widely applied to training AI components in multimedia detection and analysis. These datasets can also be applied to the concepts discussed in this paper. However, algorithms operating over plain implementations of these semantic representations are inefficient. Hence, we introduce an approach that is compatible with these representations, but performs much better on smartphones in particular.

2.3. Multimedia Indexing, Querying and Retrieval

Indexing of multimedia content can be performed at various levels. Overviews are given in [36–38]. The technical level provides mathematical representations of images in the form of histograms [39] or color histograms [40] and can be enhanced using signal processing techniques such as the Fourier transformation [41]. The low-level feature extraction process delivers edge detection, phase congruency and localized features; the high-level feature extraction process works on template matching and shape modeling [37]. Pattern recognition starts to detect similar shapes based on low-level features and applies mathematical methods to process these features [4]. Based on these methods, the semantic level

has been introduced, where object detection algorithms [9] provide semantic information that can be combined and integrated with semantic Web technologies [31].

According to [34], three levels of visual information retrieval may be distinguished. Level 1 contains low-level information, e.g., color, shape, location of image elements. Level 2 provides attributes or semantic content such as the retrieval of objects of a given type or class, including the retrieval of individual objects. Level 3 gives abstract attributes and includes search requests for named events or activities. As part of the concept-based image retrieval process, our approach utilizes existing segmentation algorithms to refine the image or video semantic information and utilize existing semantic representations of Level 1 to 3 to refine detected information. The information is represented in a MMFVG compatible with RDF, but requiring fast and efficient indexing and retrieval in form of graph codes. In general, this work contributes to closing the semantic gap: “the semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation” ([42] p. 1353).

However, applications exist where all these levels for indexing and retrieval would have to be applied in an integrated way. Hence, our approach combines and fuses these various levels into a unified semantic graph structure, the multimedia feature vector graph (MMFVG).

2.4. Graph Representation Processing

Semantic information such as the semantic Web is typically represented with graphs. Building graph structures for representing data is prevalent in computer science [43]. Based on these semantic graphs, algorithms for indexing and retrieval including graph pattern matching have been defined and developed [44]. Storing these data structures in databases such as Neo4J [45] is typical. Graph embedding can represent graphs mathematically [46] and AI-based approaches optimize pattern matching in graphs [47]. Semantic analysis based on graphs provides comprehensive query languages like SPARQL [48], RDF Schemas and many algorithms for graph transformations. In [49], a comparison of algorithms for graph similarity is given. AI-based methods are becoming more common [47]. Here, an efficient graph-based indexing and pattern matching is important, particularly on smartphone hardware. However, in many cases, such processing is inefficient. Hence, we introduce a graph encoding algorithm, which provides a fast and scalable approach as it transforms graphs into a more efficient form for processing that is specifically optimized for smartphones.

2.5. Mathematical Background

From a mathematical perspective, graphs can be represented through their adjacency matrix. For weighted graphs, valuation matrices integrate the information on the weights of edges [50]. This representation also enables the application of mathematical concepts to graphs [51]. Feature-based similarity calculation on adjacency matrices can be performed with the eigenvalue method [50]. However, the mathematical approach usually has a complexity of $O(n + e)^2$. Several approaches are described to improve this for a particular set of data or within particular conditions [52], but applied to MMFVGs on smartphones, the performance must still be improved. With our graph encoding algorithm, which uses an extension to valuation matrices of graphs, this task can be performed in $O(n + e) + O(1)$ on smartphones.

2.6. AI Pattern Matching

In recent years, AI has progressed, particularly in machine learning, which has evolved to the point of becoming relevant for both research and industrial applications. The most relevant technology for our approach is deep learning, which is a subarea of machine learning and comprises algorithms to model high-level abstractions in data by employing architectures composed of multiple nonlinear transformations [7,32]. In particular,

convolutional networks (CNN) are employed as feedforward mechanisms to increase the accuracy of feature detection. With the GMAF, CNNs are indirectly integrated into our solution. However, the current CNNs are trained to detect the most relevant part of an image and hence fail in some cases that our approach could solve. Hence, our approach includes training sets for AI image processing that is not based on the input image, but on its corresponding graph code representation of the MMFVG.

2.7. Social Media

In social media, a multimedia asset is much more than “just” a video or an image. In addition to the raw asset, metadata such as posts, text, description, corresponding posts, comments, likes or dislikes and the user’s personal settings can contribute to the relevance of a multimedia asset. Currently, it is difficult to combine all of this data into a unified model to index semantically all the corresponding components. However, in social media, usually independent searches based on keywords are applied [53]. Hence, our approach integrates and fuses detected features of these independent searches by attaching them as plugins to the GMAF and publishing their results as semantic features into the MMFVG of related multimedia assets.

2.8. Related Work

A standard multimedia processing pipeline [54] exists for multimedia processing, to which our relevant contributions can be aligned as discussed previously in [23]. The general data processing pipeline can be mapped to this standard multimedia processing pipeline [54], providing a multimedia-specific representation of the eight typical processing steps of multimedia content (see Figure 1). According to this pipeline model, multimedia processing is separated into creation, composition and delivery.

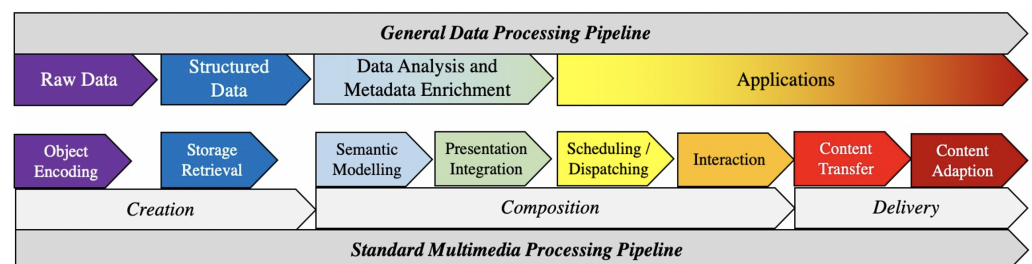


Figure 1. Mapping of the general data processing pipeline to the standard multimedia processing pipeline [54].

Multimedia creation is the process of object encoding and storage retrieval. Here, the raw data (image, video or text) on users’ smartphones and the corresponding structured metadata are used and are provided by the smartphone itself or by user applications to annotate these assets. The assumption is that metadata are directly attached to the asset in a way such as that detailed in [12] or [13]. The multimedia creation process step also includes assets that are received by other channels such as social media or messaging, where the set of metadata can vary.

In the multimedia composition phase, numerous algorithms for the lower-level of feature extraction, such as SIFT or Haralick, have been defined, implemented and applied to multimedia content [21,32,55,56]. Segmentation algorithms such as MSER or SIMSER [32,57–59] help to identify the correct bounding boxes of objects within images, and several ontology-based models for semantic annotation have been proposed in [32,60–63]. Recent approaches regarding feature fusion in special topic areas are contributing to the approach; we are taking [64,65]. In [66], a confidence-based ensemble is proposed to provide semantics and feature discovery to dynamically extend the traditionally static semantic classifiers and a mathematical model is provided to resolve semantic conflicts and to discover new semantics. The multimedia delivery, scheduling and interaction phases are not relevant in the context of this paper.

2.9. Discussion and Summary

As stated above, there are many concepts, algorithms and tools that provide solutions for topics in the area of multimedia processing. However, even deploying these, it is currently not straightforward to retrieve Multimedia Content for queries such as “show me a picture where I got my new watch” or “show me the video where my daughter had her first concert”. One explanation is that most of the described algorithms usually run independently from each other and do not contribute to a common metadata model. A second explanation is that there is a great amount of unused potential by focusing only on a single asset and not taking into account that related or similar assets can also contribute to the metadata of the asset under investigation.

Our work discussed here contributes to closing this gap by providing a generic framework to integrate the current state-of-the-art algorithms (GMAF), a data structure to fuse existing features into a single unified model (MMFVGF) and an ML component to refine the feature model and to provide semantic indexing, querying and retrieval (AI4MMRA).

3. Conceptual Design

Our approach is aligned with the user-centered system design defined by Draper [67] and based on Nunamaker’s methodology [68]. To formalize our model in a visual language, we use the UML [69] and present an initial use case design for the most relevant activities. In our context, two major actors can be identified; they interact with a multimedia asset management system (see Figure 2). The producer creates or modifies multimedia content by taking pictures, videos, posting to social media or by manually creating annotations. The consumer will then search, filter and retrieve the asset. The producer and consumer can be the same person in real-world scenarios. All use cases are part of the data analysis and metadata enrichment phase of the general data processing pipeline (see Figure 1).

The use cases Create or Modify Asset, Manual Annotation or Asset, Search Asset, Filter and Retrieve Asset are already in place on almost every smartphone or multimedia application. Our approach will extend these basic use cases by providing additional functionalities. From the users’ point of view, nothing will change, and existing applications can be used in the same way as before. We will enhance the search accuracy of these applications by providing additional components that will increase the semantic quality of multimedia retrieval (see Figure 2).

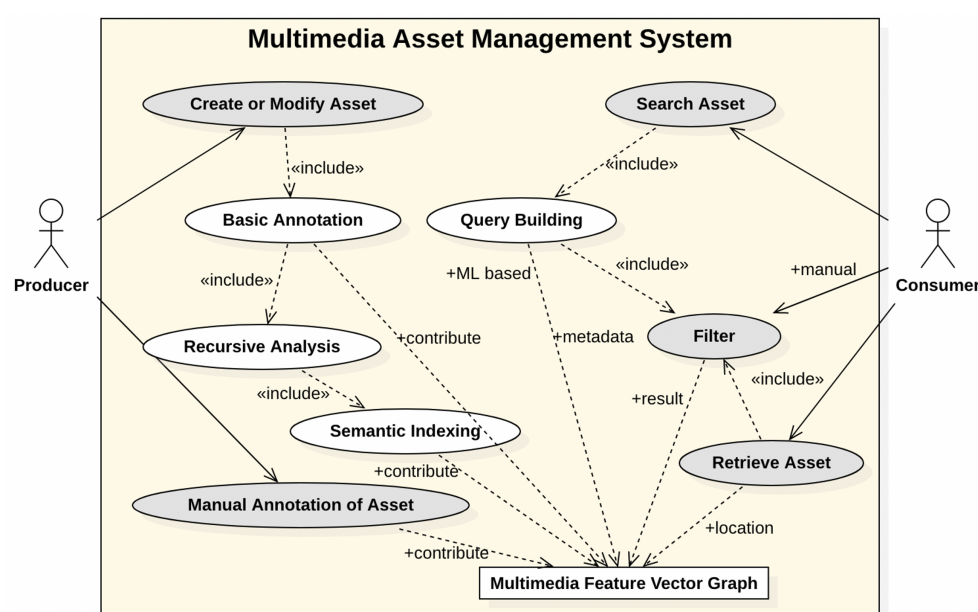


Figure 2. Initial use case design to extend a multimedia asset management system.

Many smartphones are already able to provide the Text use case and integrate meta-data based on the standards such as EXIF or MPEG7 into their multimedia assets. These include location, date and time and are extended in some cases by built-in face detection or object detection [70]. We extend this use case by feeding its data into the multimedia feature vector graph (MMFVG) as part of the multimedia feature vector graph framework (MMFVGF), which enriches and also fuses it with other metadata. These additional meta-data are generated in subsequent processes. The recursive analysis applies a chain of filters to every detected object or region in order to refine the detection results. Semantic indexing fuses the various metadata into a unified model represented by the MMFVG. To take full advantage of the MMFVG, the Search use case has to also be extended by introducing a query building component that employs the MMFVGF.

3.1. Framework Overview

From a conceptual architecture perspective, three frameworks are required to fulfill the requirements given by the use cases (see Figure 3): (1) The generic multimedia analysis framework to analyze multimedia assets, (2) the multimedia feature vector graph framework to maintain a central data structure and (3) the AI4MMRA to optimize the results based on ML. Here, we use the term framework to describe software components that work together in a standardized way in order to provide a certain functionality. A brief description of these frameworks is given here:

Generic multimedia analysis framework (GMAF): a framework capable of combining and utilizing existing multimedia processing systems for image, video and text data (such as social media posts). It can be regarded as a chain of semantic metadata providers, which will produce all the data that is required to construct a semantic index. The GMAF represents the use cases Basic Annotation, Recursive Analysis and Semantic Indexing.

Multimedia feature vector graph framework (MMFVGF): a framework that fuses all the semantic metadata into a large semantic feature vector graph and assigns it to each media asset. Weights, nodes and references are structured in a way such that the relevance of features within a multimedia asset can be determined accurately.

AI for multimedia retrieval and access (AI4MMRA): an ML-component to process and permanently adjust the feature vector graphs of each image or video in order to refine the relevance data. AI4MMRA provides a permanent reprocessing in terms of the use case Semantic Indexing in addition to a Query Builder component.

A central shared resource of this approach is the multimedia feature vector graph (MMFVG), which fuses existing ML-based semantic indexing methods into a single representation for multimedia query and retrieval. The MMFVG is the result of the generic multimedia analysis framework (GMAF) and will be employed within the AI for multimedia retrieval and access (AI4MMRA).

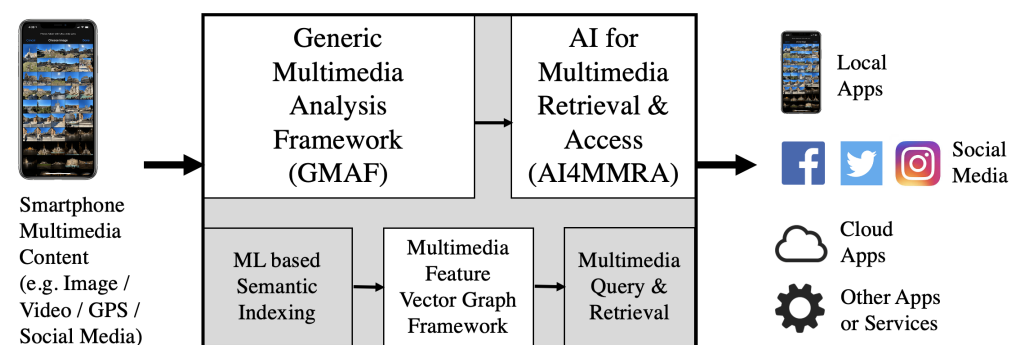


Figure 3. Multimedia asset management system architecture including the generic multimedia analysis framework (GMAF), multimedia feature vector graph (MMFVG) and AI for multimedia retrieval and access (AI4MMRA).

3.2. Initial Query Types

To show the expressive potential of the MMFVG, let us look at some initial query types supported by this data structure, which could represent queries where current algorithms are likely to fail and the reasons for this:

- Q1: “show me the picture from Florida, where Jane wore flipflops”—Some of the algorithms will locate pictures of Jane, and possibly also pictures where a person is wearing flipflops, but most of the algorithms will fail on their intersection. One would also retrieve pictures of Jane, where another person next to her is wearing flipflops.
- Q2: “show me a picture where I got my new watch”—This type of query is problematic for most of the current algorithms. One reason is that data from more than one picture are required to solve this query. A comparison of “old” pictures with those where a “new” object appears is required. The second reason is that all the object detection algorithms are trained to find relevant objects, which usually a watch is not. Even employing the fine-grained object detection, the “watch” will not receive any relevance for indexing. Even more difficult to answer is the question, “what happens if my “new” watch becomes “old” again”. (This is one reason why AI4MMRA performs a permanent reprocessing of feature vectors graphs.)
- Q3: “can you find the picture where I had my leg broken”—This is also difficult to solve with current technology. How should a broken leg be detected? The semantics that broken legs might be detected by a white plaster cast in combination with the region detection “leg” is hardly possible. Maybe I broke my arm at another time? How should current algorithms distinguish? Even if you would have a Twitter post with that picture, none of the current algorithms would be able to combine this properly.

The combination of GMAF and MMFVG provides an approach toward a solution to how such queries could be processed with accurate results. The GMAF produces and analyzes feature vectors and the MMFVG integrates and provides the MMFVG data structure for querying.

3.3. The Multimedia Feature Vector Graph

The MMFVG has not been designed to be human-readable or -viewable, but to provide optimum AI processing support. The basic concepts are closely aligned to graph theory, so visualizations can be provided of parts of the MMFVG, but in general, an MMFVG for a given multimedia asset will be too cumbersome to be visualized in a sensible manner. We define the MMFVG as a directed, weighted graph with attributed vertices and edges. It can be applied to a single multimedia asset to represent semantic, technical and content-based metadata. In addition, each asset’s MMFVG has edges to parts of other assets’ MMFVGs, which lead to a recursive MMFVG graph structure, supporting loops and subgraphs. Figure 4 shows the UML class diagram of the MMFVG to illustrate the formal definition. Operations and other details are hidden in this visualization. According to [43], the MMFVG is a directed graph with subclasses of directed cographs defined as:

$$MMFVG_{Asset} = (V, E) \quad (1)$$

where V is the set of vertices and E is the set of Edges. Both vertices and edges are detailed by attributed subclasses, which represent the corresponding multimedia features. Vertices V in the context of the MMFVG use the following attributed subclasses $V \in \{n, r, e, w, cn, t, m, s, l, p\}$ (UML classes are formatted in bold letters to link the graph objects to the corresponding elements in Figure 4.):

- **Nodes** $n_1, \dots, n_n$ represent objects, activities or regions of relevance that have been detected within an asset. The root node r is the dominating node of the MMFVG for a given multimedia asset and is also represented by the node class. External nodes e represent external information structures such as [14,15] or parts of other MMFVGs and are also represented by the node class.

- **Weights** $w_1, \dots, w_n$ represent the relevance of a node according to a special context. The context is an important description to refine the search and query scope. It is created by the related metadata of an MMFVG structure and helps to determine the correct usage of homonyms. The context can also represent different aspects of the same object, such as in the timeline of a movie, where the same object can be used differently during the story of the movie. A weight represents the importance of the node in the overall image or video and the deviation from “normal” content compared to other assets of a similar kind (c.f. the “new watch” example). It is calculated initially by the GMAF, but constantly refined by AI4MMRA.
- **Childnodes** $cn_1, \dots, cn_n$ are subobjects that have been detected by recursive application of the GMAF, e.g., Person $\rightarrow$ Body $\rightarrow$ Arm $\rightarrow$ Left Hand $\rightarrow$ Fourth Finger $\rightarrow$ Ring. So, one of the Person’s child nodes would be the “Body”, one of the Body’s child nodes would be “Arm”, and so on. Child relationships in the MMFVG are transitive.
- **TechnicalAttributes** $t_1, \dots, t_n$ represent nonsemantic features of a node. These can be the color footprint of this node (c), its bounding box (b) within the image (defined by x , y , width, height), the DPI resolution of the section, information about, e.g., sharpness or blurring, etc.
- **GeneralMetadata** m represent the asset’s general metadata object, which is extracted by EXIF or MPEG7 [12,13] and contains information such as, e.g., Date, Time, GEO-coding, Aperture, Exposure, Lens, Focal Length, Height, Width and Resolution. In m , a unique identifier for the multimedia asset is also stored.
- **Location** Nodes $l_1, \dots, l_n$ represent locations where the original multimedia asset or copies of the asset in original or different resolutions or segments are placed.
- **SynonymInformation** $s_1, \dots, s_n$ point to a normalized synonym graph, where the “is a” relationship is modeled. So, for a “Person”, we would find also, e.g., “individual”, “being”, “human”, “creature” and “human being”.
- **Security** and **Privacy** p define the asset’s privacy and security settings, such as for an access control List. This ensures that users will only process and receive their own content and content that has been shared with them by other users.

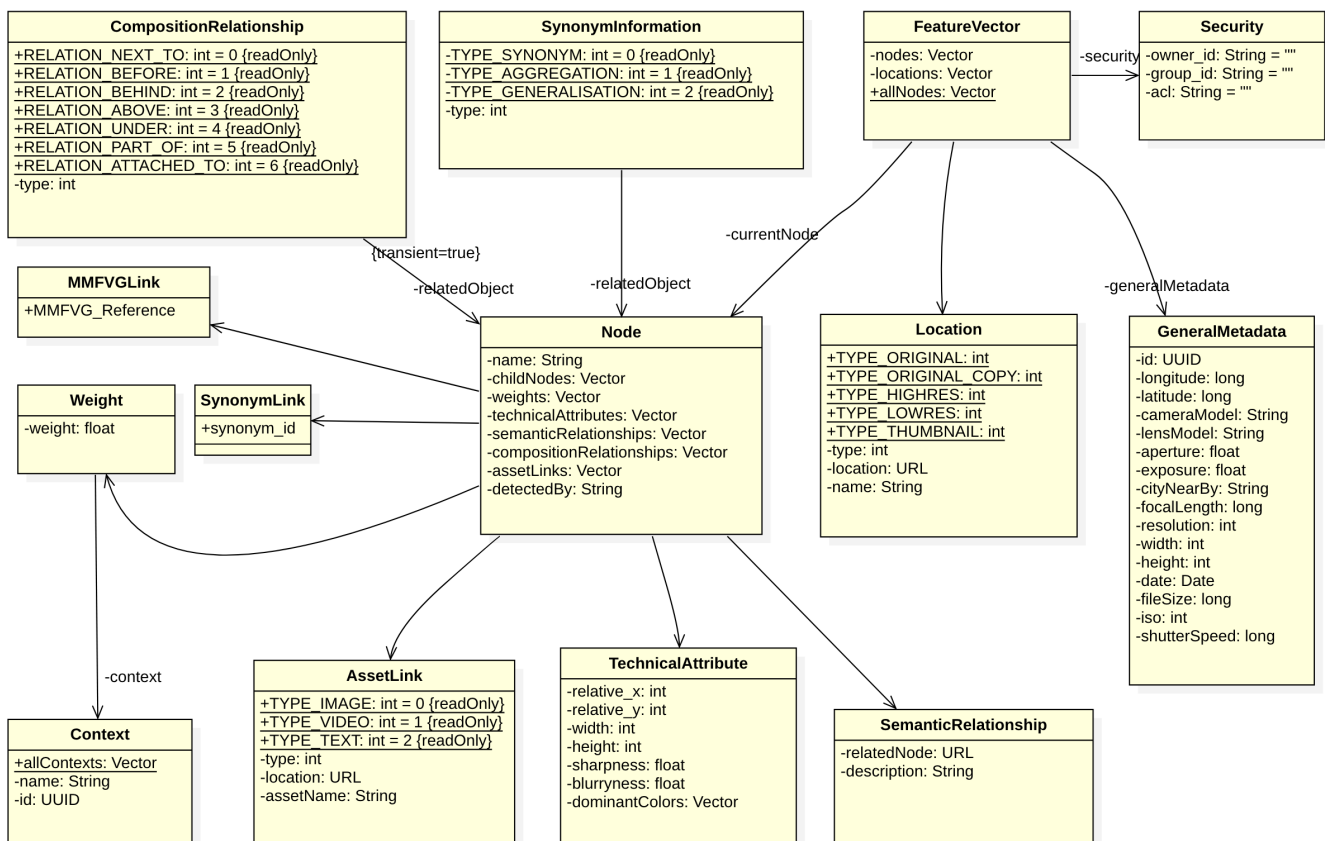


Figure 4. UML class diagram of the MMFVG structure.

Edges E in the context of the MMFVG are also represented by attributed subclasses and hence can be defined as $E \in \{cr, sr, sn, mm, al\}$ with:

- **CompositionRelationships** $cr_1, \dots, cr_n$ providing a relationship between a multimedia asset's objects. It contains information such as, e.g., "next to", "in front of", "behind", "attached to", and is calculated by recursively applying the object bounding boxes and measuring the distances between them.
- **Semantic Relationships** $sr_1, \dots, sr_n$ connect each node to a position within external semantic knowledge graphs such as [14].
- **SynonymLinks** $sn_1, \dots, sn_n$ reference synonym information.
- **MMFVGLinks** $mm_1, \dots, mm_n$ represent the connection to other MMFVGs as standard references in the node class.
- **AssetLinks** $al_1, \dots, al_n$ point to location nodes.

The MMFVG itself does not provide any functionality. It is a data structure for general use to recursively describe multimedia assets and their semantic relationships and to provide a generic structure to which the standards of existing specifications can be mapped. An instance diagram of a MMFVG is shown in Figure 5. For illustration purposes, only some of the relationships are shown Figure 5. At what point of time during the overall process the MMFVG has to be constructed and when the permanent reprocessing of the AI4MMRA is performed must be investigated. This is subject to further analysis with respect to performance, runtime and data volume and will refine the overall architectural topology.

Due to the graph-based nature of the MMFVG and the expected number of nodes and edges, it is clear that current graph-based indexing algorithms will require substantial processing time for indexing and retrieval. Therefore, we will employ a projection of the MMFVG into a two-dimensional pixel image for indexing and retrieval, which leads to improved performance, particularly when utilizing current smartphone hardware. The next section describes this approach.

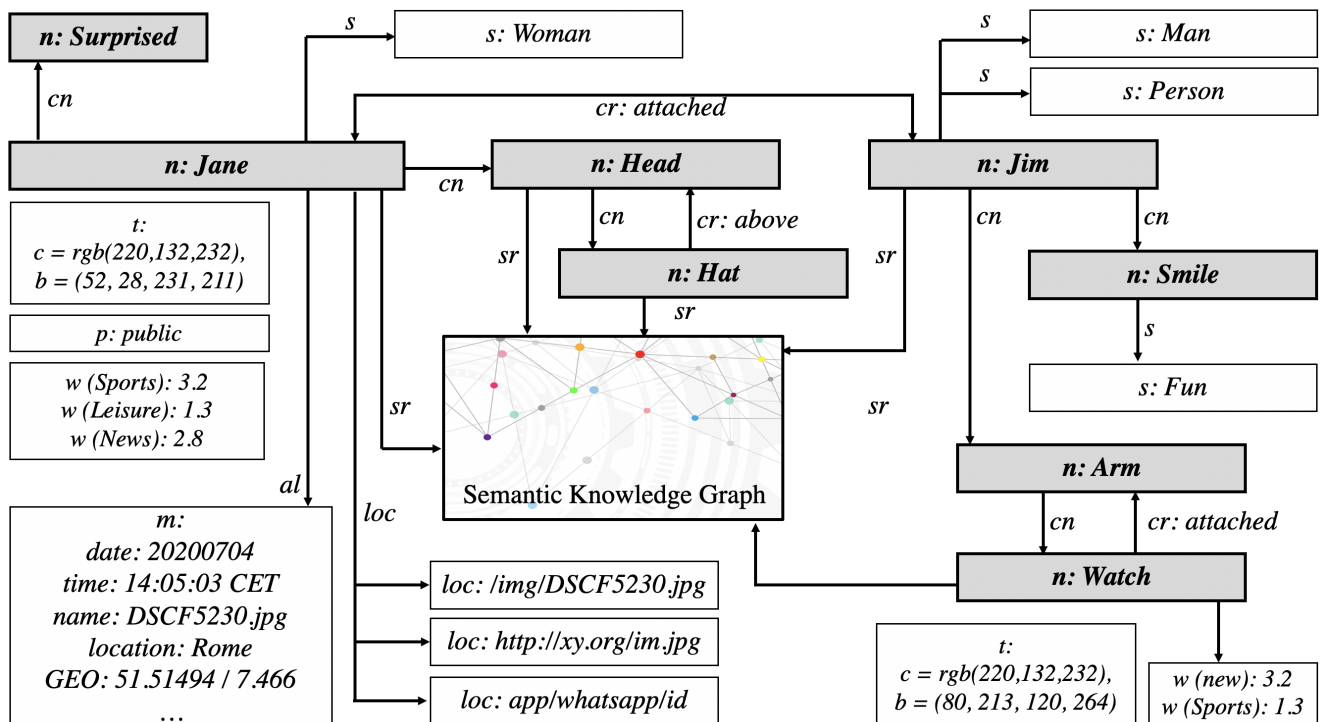


Figure 5. Exemplary multimedia feature vector graph (MMFVG) of an instance with attributes.

3.4. Fast Indexing and Retrieval for Graphs on Smartphones

For a typical image such as those covered in [35], the MMFVG will contain approximately 1000 nodes and 10,000 edges. Current graph algorithms for indexing and retrieval [44,71] with this number of nodes and edges are difficult to implement on smartphones due to memory and runtime restrictions as some of them require $O(n + e)^2$ to find similarities in graphs. Hence, we have defined and evaluated an indexing and retrieval algorithm that is particularly optimized for smartphones and associated integrated hardware. The proposed graph encoding algorithm projects a MMFVG into a two-dimensional pixel image and performs operations on pixels instead of graph traversal. This pixel image should not visualize the graph in a human understandable way as other tools do [72]; rather, the pixel image should be regarded more in the sense of machine-readable barcodes [73] or QR codes [74] (see Figure 6). Thus, in the following sections, we refer to this image as graph code (GC) and to the calculation algorithm of the image as the graph encoding algorithm.

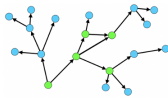
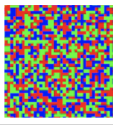
	Barcode	QR-Code	Graph Code
Input	1234567890	A simple text as QR Code	
Output			

Figure 6. A visual comparison of barcodes, QR codes and graph codes.

In general, the graph encoding algorithm receives a graph structure as input—in our case, a MMFVG. A flattening function f_{GC} projects this graph into a two-dimensional pixel image (the graph code), which then also represents the structural information of the graph:

$$f_{GC}(MMFGV_{Asset}) = \begin{pmatrix} x_{1,1} & \dots & x_{n,1} \\ \dots & \dots & \dots \\ x_{1,n} & \dots & x_{n,n} \end{pmatrix} \quad (2)$$

where $x_{i,j}$ represents color of the pixel at position (i, j) in the generated graph code. The graph code basically uses the MMFVG's valuation matrix [75] and extends it by assigning an RGB color to each position of the matrix. Thus, not only weights or the number of edges can be coded in the valuation matrix but also node or edge types can be stored using up to 16.8 million colors per position (see Figure 7). It must be guaranteed that the pixel position of each node and edge is unique. Thus, f_{GC} has to be applied once per graph to calculate the graph code, and once the GC is calculated, no more graph traversal is required. Compared to mathematical flattening functions based on linear algebra such as those in [44,47], we follow a more algorithmic approach to define the transformation of a graph into the two-dimensional pixel space based on a colored valuation matrix. An example of this approach is given below in Section 3.5.



Figure 7. Illustration of the graph code generation.

As f_{GC} is lossless, for each given pixel, the corresponding objects (nodes or edges) of the underlying graph can be identified directly by the function f'_{GC} , which will return the original MMFVG and the corresponding assets. This reduces time required for semantic retrieval, as the matching object can be accessed immediately (see Figure 8). The function f'_{GC} is also employed as a retrieval function and returns a reference to the underlying semantic feature graph. The concrete implementation of f'_{GC} depends on the architecture for storing the graph codes in a database.

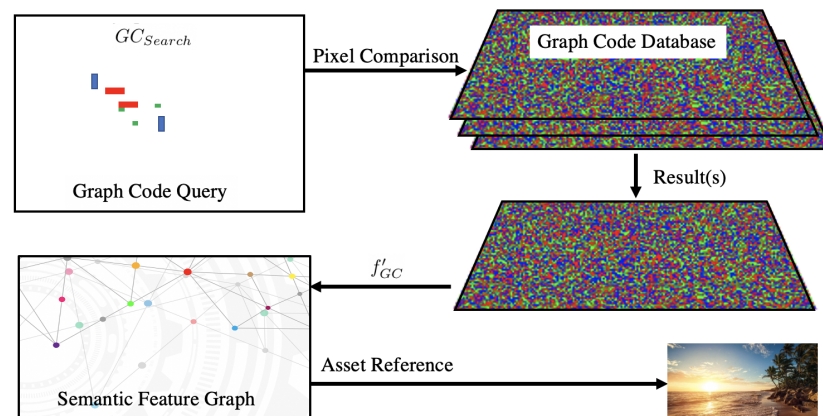


Figure 8. Graph-code-based querying and retrieval.

Finding similar semantic information or calculating the overall similarity of two MMFVGs is a key task for querying and retrieval. Instead of using mathematical operations like the eigenvalue calculation [75], the graph encoding algorithm provides the opportunity to use image-based operations such as pixel comparison, edge detection and AI-based pattern matching instead and to utilize special hardware for parallelization. As for each MMFVG, a GC is created, representing the complete semantic information of the corresponding asset, multimedia graphs can be compared to each other using image comparison and pixel-based calculations instead of graph traversal algorithms. These pixel-based operations can be highly parallelized [76–78]. If these operations run on machines such as the Nvidia RTX 2080 GPU [79] or on some special smartphone hardware such as Apple’s A14 Bionic [16], almost all processing of an MMFVG comparison can be completed in a single parallel step.

For querying and retrieval, the graph encoding algorithm is based on the query-by-example paradigm [80]: a query is represented as a reduced (in most cases, very reduced) subgraph to which the GC algorithm is applied resulting in a reduced GC_{Search} structure, containing only minimal pixels. In the worst case, we must check every GC in the database, as to whether they contain matching pixels at the same positions. This may appear inefficient, but given that each comparison is performed in $O(1)$, and that the overall number of assets is not too voluminous on smartphones [81], this is rather exemplary. In addition, it is straightforward to reduce the number of GCs required for comparison by applying a content-based prefiltering. As our images are already annotated, we can simply grab the first level of detail including the most relevant objects and apply the retrieval only to other images with similar objects.

Thus, graph codes and the graph encoding algorithm provide an optimized approach to comparing MMFVGs and furthermore employ specialized hardware for image processing of many current devices.

3.5. A Graph Encoding Algorithm Example

The following graph encoding example shows how f_{GC} works in general. The application of f_{GC} on multimedia is demonstrated in Sections 4 and 5. For this example, we will employ the input graph shown in Figure 5. Table 1 shows the corresponding pixel map. For the overall ontology, we define a selection of nodes, contexts, and synonyms in this example, containing “Jane”, “Head”, “Hat”, “Surprised”, “Woman”, “Jim”, “Arm”,

“Watch”, “Smile”, “Fun”, “Sports” and “New”. As this example is for illustration purposes only, we will ignore all the other nodes that would be in the MMFVG. Thus, the corresponding graph code will result in a 12×12 pixel image, containing 144 pixels in total. For the description of weights, edge types or node types and the aggregated information of more than one MMFVG, we can use up to 12.8 million colors per pixel. In Table 1, we paint nodes in black, synonyms in red and contexts in brown. Relationships such as “attached” are painted according to their type (attached = green, above = yellow, child = blue) and colors for weights are calculated according to their value in gray tones.

Table 1. Graph code image for the MMFVG example shown in Figure 5.

	Jane	Head	Hat	Surprised	Woman	Jim	Arm	Watch	Smile	Fun	Sports	New
Jane	bl					g						
Head	b	bl	y									
Hat	b	b	bl									
Surprised	b			r								
Woman	b				r							
Jim	g					bl						
Arm						b	bl	g				
Watch						b	b	bl				
Smile						b			bl			
Fun						b				r		
Sports		lg						gr			br	
New								lg				br

The corresponding extended valuation matrix for this case is defined as follows. (In this matrix, the following abbreviations are used: bl = black, b = blue, br = brown, r = red, g = green, lg = lightgray, gr = gray, y = yellow. The corresponding rows and columns are taken from Table 1.):

$$\begin{pmatrix}
 bl & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 & 0 \\
 b & bl & y & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 b & b & bl & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 b & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 bu & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 g & 0 & 0 & 0 & 0 & bl & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & b & bl & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & b & b & bl & g & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & b & 0 & 0 & bl & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & b & 0 & 0 & 0 & r & 0 & 0 \\
 lg & 0 & 0 & 0 & 0 & 0 & 0 & gr & 0 & 0 & br & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & lg & 0 & 0 & 0 & br
 \end{pmatrix}$$

Whilst current graph algorithms have a complexity of $O(n + e)^2$ for each comparison or pattern matching step, our algorithm can solve the same problem in $O(n + 1) + O(1)$. In the first step ($O(n + e)$), the graph needs to be transformed into a graph code representation, which requires visiting each node and edge once. The second step can then be performed almost in parallel on current smartphones ($O(1)$). As the first step can be performed in advance, only the second step has to be considered in regards of the retrieval process. This improvement in performance compared to graph traversal algorithms is significant.

For our prototypical implementation, we employ a slightly modified GC algorithm, which contains nodes only for the actually detected features. Whenever additional features are detected in multimedia assets, we expand the matrix size for the GC and perform a

recalculation of the models to match this expanded matrix. Applied to the above example, we would start with only eight rows and columns. Once new nodes for “Steve” or “Mike” are inserted, the GCs would simply grow into more columns and rows “on demand”.

3.6. The Generic Multimedia Annotation Framework

Following discussion of the central datastructure MMFVG, this section discusses the purpose and functionality of the GMAF. Figure 9 shows this conceptual approach and how the multimedia feature vector graph is constructed within GMAF and MMFVGF. The MMFVGF can facilitate multiple dimensions of feature extraction, in this way significantly extending and refining the standard multimedia layers as defined in the so-called strata model [22], in which each media segment has its own layer distinguished by horizontal layers (mostly time-based) and vertical layers (synchronized media signals). Based on these layers, additional feature representations are introduced as dimensions within the MMFVG:

- Features within a single multimedia asset, such as object detection, technical metadata and basic semantic metadata;
- Features on higher semantic levels, such as people or face recognition, mood detection and ontology links;
- Recursive features, which result from applying the GMAF recursively to subsets of the multimedia asset;
- Indirect features, which result from linked assets and refine the features of the current asset (other scenes in a video, corresponding social media posts);
- Intra-asset features, which use information from assets of the same series, location or timestamp or data from generally similar assets to refine an asset’s features.

These feature levels can be classified into horizontal levels, which mostly contain nontemporal indirect features or intra-asset features (besides the temporal dimension), and vertical levels containing all features within a single asset, including higher semantic levels and recursive features. GMAF and MMFVGF will produce a unified multimedia feature vector graph that represents the fused data from all these levels. Due to the plugin architecture nature of GMAF, additional feature extraction APIs can easily be integrated. Figure 10 shows the sequence diagram of the feature extraction in general.

Typically, the GMAF processing is performed both locally (on the smartphone) and centrally (on a server). On the smartphone, the local functionalities for object detection such as [70] will be used to identify and create an initial MMFVG, which synchronously or asynchronously is extended by more information fetched by the GMAF Web service, where additional plugins such as those in [9] or [11] are available. The fusion into a single MMFVG is performed locally (see Figure 10).

3.7. Natural Language Processing (NLP)

Most Smartphones have integrated Digital Assistants like Siri [28,82]. These assistants can provide Natural Language Processing (NLP) spoken dialogue interaction interfaces (chatbot assistants) to users. NLP can facilitate processing the types of queries discussed in the previous section. In general, we follow the approach given in [83] to generate SPARQL-Queries based on NLP, but incorporate an additional component to generate a simple Graph Code based on the keywords of a SPARQL-query, which then facilitates fast Graph-Code-Retrieval.

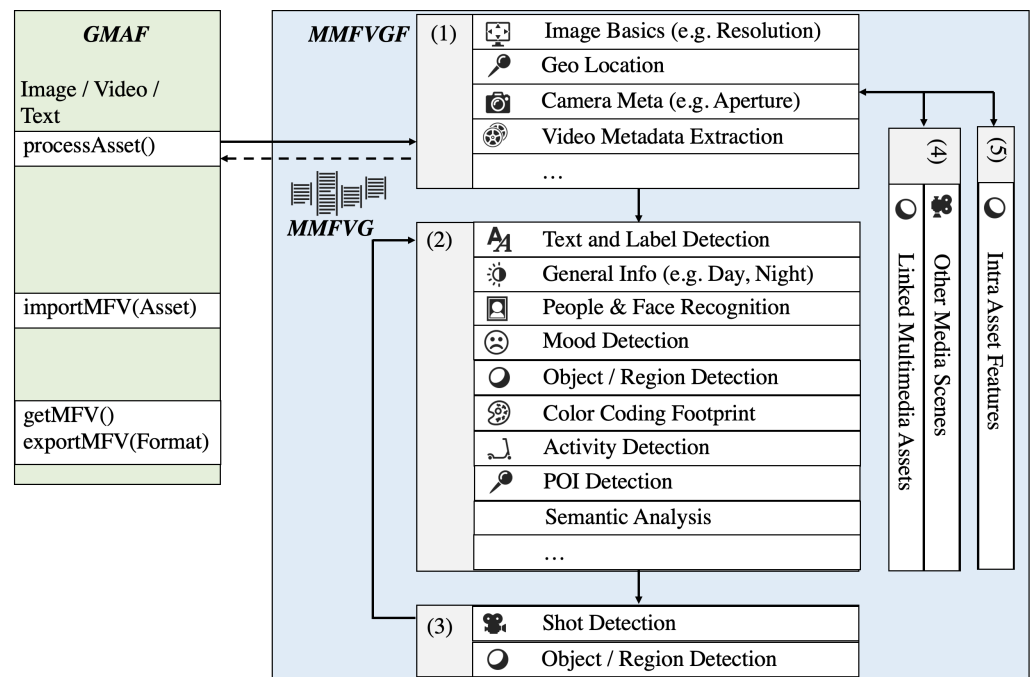


Figure 9. Conceptual processing approach with basic structure of GMAF and MMFVGF.

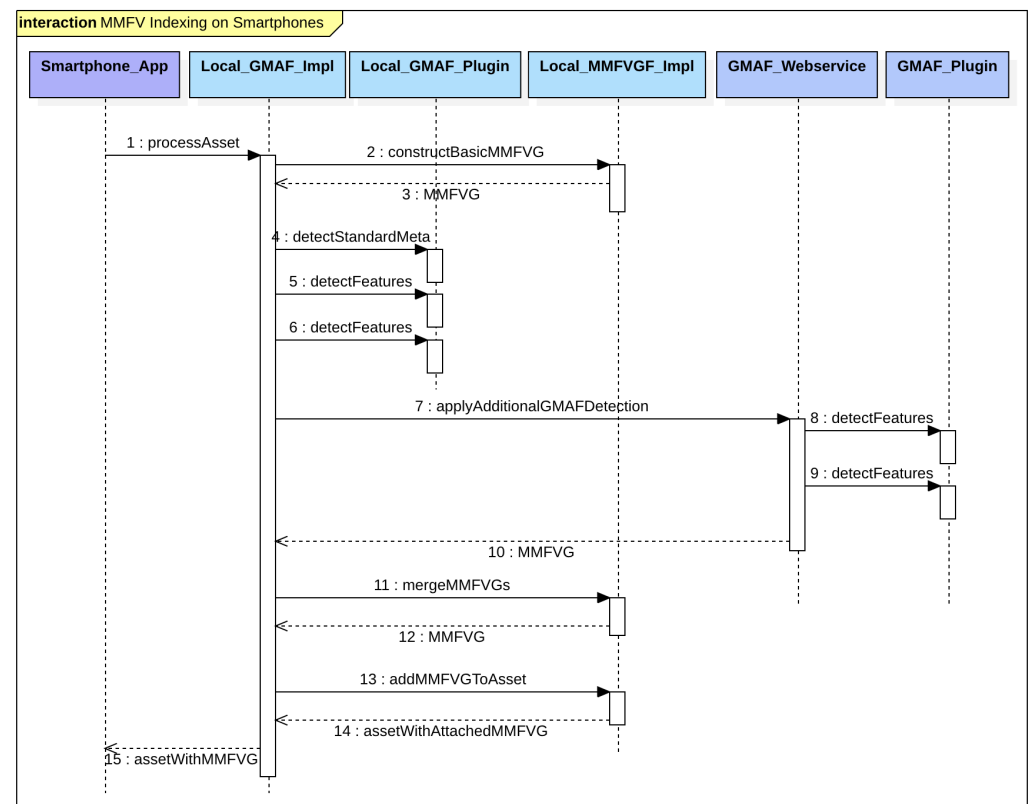


Figure 10. Sequence diagram of typical processing flow within the GMAF.

3.8. Social Media

In the social media domain, our concepts can provide several improvements. Users will benefit from more efficient local indexing and retrieval, as social media posts are one of the sources utilized for optimizing the MMFGVs. Hence, social media can enrich the semantic information stored on multimedia assets. If local multimedia assets are posted to social media, they will be also annotated with a MMFVG, so that social media providers

will be able to read and reuse all the semantic data, which has also been calculated on the local smartphone. Embedding graph codes into the original asset would also increase the overall retrieval performance, but this will first require a standardization of the underlying ontology. This could be established if a social media provider such as Facebook integrates its own ontology both in the smartphone application and in its server-side processing. In this case, it would receive perfectly annotated and already optimized multimedia assets. For users, this kind of seamless integration would be the optimal solution, so that existing smartphone applications would transparently handle the creation of MMFVGs.

3.9. Summary

In this section, we discussed our model of layers and dimensions and corresponding basic technologies and algorithms. Our frameworks generate a multimedia feature vector graph, which is then processed by smartphone-optimized algorithms to create rapid indexing. For retrieval, we employ NLP and the corresponding SPARQL representation. In the following section, we will briefly describe our prototypical implementation.

4. Implementation

To exhibit our conceptual approach, we provide several selected implementations of the relevant components of GMAF, MMFVGF and graph codes. Whilst the GMAF is based on standard technologies (Java, SOAP, REST, XML and HTTP) and an optimal means of combining and utilizing them, the MMFVGF and the AI4MMRA cannot be achieved with standard technologies and must be derived from methods given in [12,13,15,27]. The foundations and the implementation of graph codes are introduced in this section. The prototype implementation of the GMAF including the MMFVGF and graph codes is given at [84]. Our prototype employs methods of [9,12] as well as [85] to implement the plugins for GMAF also shown in Figure 5. Several different semantic models have to be combined, fused and mapped during the process of creating the MMFVG.

4.1. Generic Multimedia Annotation Framework (GMAF)

The GMAF is implemented as a framework that employs standard image processing technologies like object detection, basic semantic annotations and standard image metadata as described in previous sections by applying methods in [9,12,86]. New algorithms can be easily attached as plugins and a recursive feature enrichment is also part of the GMAF and basis for the MMFVG (Figure 5). Usage of the GMAF is straightforward, as it already provides ready-to-use adapters for the major multimedia processing tools. In general, the GMAF will process each uploaded multimedia asset according to a processing chain, where any available plugin is applied to enrich the MMFVG. The GMAF is available as a standard JEE Web application and can be deployed to any JEE compatible server or container [86,87]. Figure 11 shows a sample application of the GMAF with an uploaded image (picture licensed from Adobe Stock [88]), which is then internally processed into a MMFVG and exported for visualization purposes to GraphML, visualized with yEd [72] (see Figure 12). Current results of GMAF processing demonstrate that the level of detail increases significantly due to algorithm recursion. The level of detail that will provide the best MMFVG-results for further processing is currently the subject of investigation. The implementation of further GMAF plugins is ongoing.

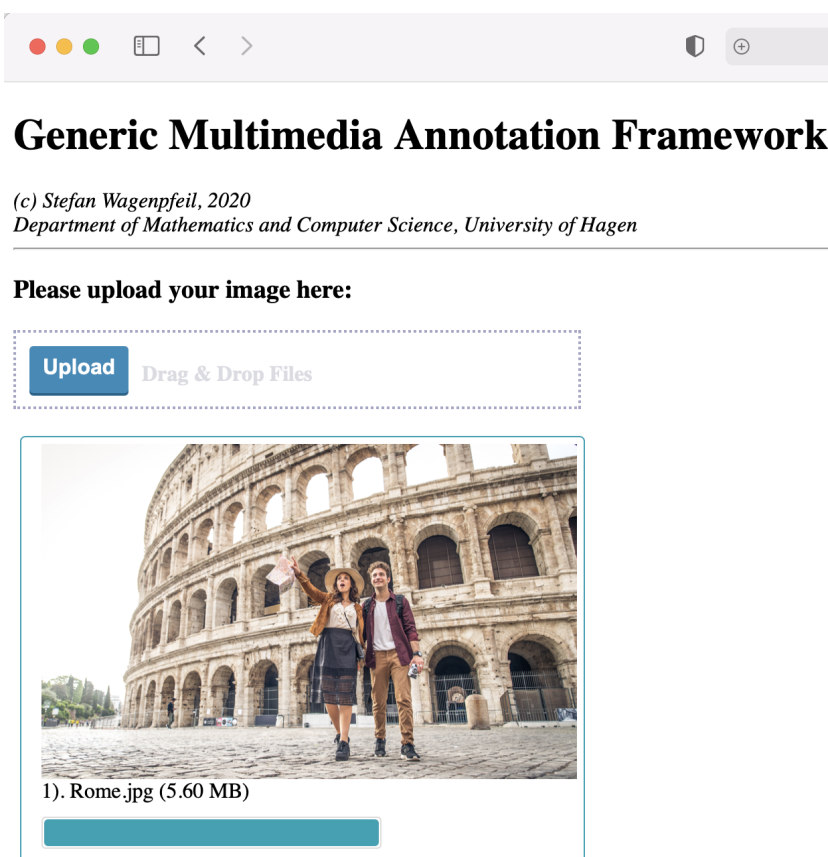


Figure 11. GMAF Web application with uploaded image of Rome.

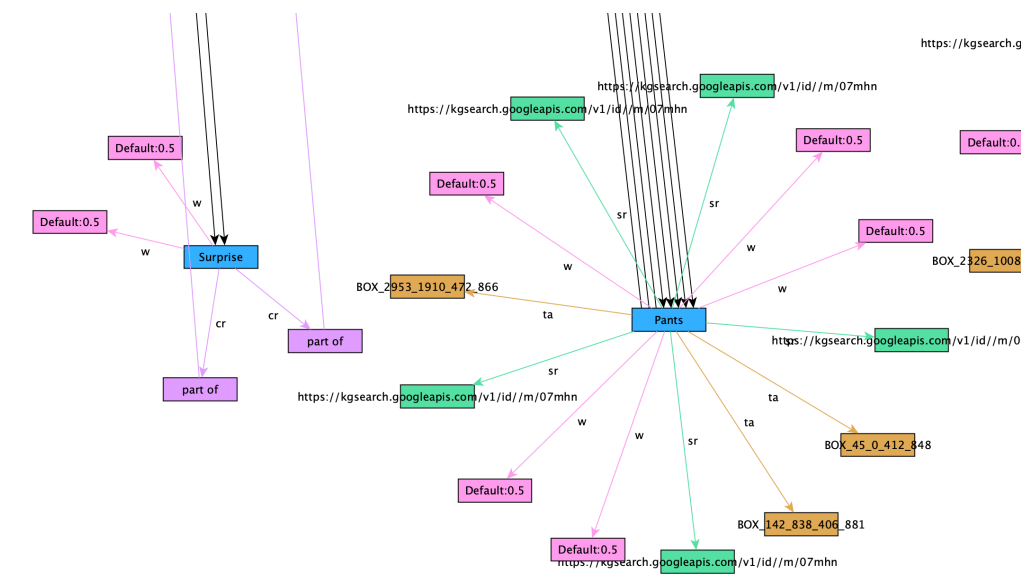


Figure 12. GraphML export of MMFVG constructed from the sample image.

4.2. Multidimensional Feature Vector Framework (MMFVGF)

The MMFVGF calculates the multidimensional feature vector (MMFVG) and provides structures to organize and maintain this data structure. MMFVGF is also implemented in Java and includes importers and exporters to various formats (e.g., JSON, GraphML, XML) and APIs to employ the MMFVGF and access the MMFVG via SOAP and REST.

Implementation of the various objects of the MMFVG is based on current multimedia standards such as those in [12–14,24,72,85].

4.3. Graph Code Implementation

Our prototype employs an initial implementation of the GC algorithm to generate graph code representations of the MMFVG. The implementation has been completed in Java for illustration purposes, and additional hardware-based implementations employing CUDA [16,79] are in progress. The graph code of the sample image in the previous section is shown in Figure 13.

The image in Figure 13 is output from a Java implementation. The algorithm step of processing a MMFVG node is shown here:

```
public void process(Node n) {
    for (ChildNode cn : childnodes) {
        int x = getIndexFor(n.getNodeName());
        int y = getIndexFor(cn.getNodeName());
        Color color = getColorForType(Node.CHILD_NODE);
        pixels[x][y].setColor(color);

        process(cn);
    }
}

public int getIndexFor(String name) {
    // wordMap is a Hashtable containing all
    // words of the current ontology
    return wordMap.get(name);
}

public Color getColorForType(int type) {
    if (type == Node.CHILD_NODE)
        return new Color(220,220,100);
    if (type == Node.SYNONYM)
        return new Color(100, 100,220);
    // ...
}

public Color getColorForWeight(float weight,
Context c) {
    // weight is between 0 and 1
    String name = c.getName();
    int r = getColorForContext(name);
    int g = (int)()255 * weight);
    int b = 255;
    return new Color(r, g, b);
}
```



Figure 13. Snippet of generated graph code from the original input image.

4.4. AI4MMRA

In order to process queries such as Q1, Q2 or Q3 discussed in Section 3.2, AI4MMRA is introduced as a specialized AI algorithm to index and retrieve multimedia content. It is based on the IVIS reference model for advanced interfaces [89] and adapts this model to be compatible with the EDISON project [90].

The key task for AI4MMRA is to detect deviations of an asset compared to typical other assets of this type. Whereas the GMAF calculates weights according to standard rules, AI4MMRA is trained to detect even minor changes of one asset such as the “new watch” in example Q2 noticing that the “watch” itself usually would not be of any significant relevance for the picture’s description. For this deviation, a semantic comparison must be implemented, which can include texts from social media (“look, I have a new watch”), other images (with the “old watch”), location, date and time information (“me at the watch store”) and also a comparison of the MMFVG from different multimedia assets. As semantic relevance directly correlates to query context, the relevance information must be calculated for each context.

AI4MMRA utilizes the graph codes as a machine-learning-compatible representation of the MMFVG to detect the relevant parts of the MMFVG according to user or query context, deviation from other multimedia content and supporting information such as social media. Graph codes are a solid basis for ML, as there are already many training models for pattern matching and deviation detection. The main goal of AI4MMRA is to detect semantic changes of multimedia assets across multiple images or video. Graph codes, being the pixel representation of semantic graph data, provide convenient input for ML components. In case of the “new watch” example, AI4MMRA will detect a deviation based on graph codes (e.g., the color of Jim’s watch has changed), as pixels do not match the current pattern and will introduce new attributes to the underlying MMFVG, e.g., introduce the edge new to the watch node. After this, a reprocessing of a particular set of MMFVGs and corresponding graph codes is performed. In addition to detecting deviations, AI4MMRA will also detect similarities and confirm them by increasing the weights of the corresponding MMFVG edges accordingly (see Figure 14). Due to the design of graph codes, it is straightforward for AI4MMRA to detect deviations or similarities, as they appear as lines or connected sections in the graph code, which is a common task for ML-based pattern matching (see Figure 13).

Applied on smartphones, the concept of AI4MMRA is promising due to the employment of special AI-optimized hardware like A14-Bionic [16], which has been specially designed to support pattern matching and ML tasks on smartphone devices. Graph codes perform even better on this smartphone hardware, as many existing ML models and optimizations can be employed directly, whereas ML-based models for graph similarity are quite rare on smartphones.

4.5. Querying

As the MMFVG has been designed to be represented in RDF [32] and hence is compatible to all corresponding concepts [32], a query to the MMFVG can be written in the SPARQL [48] formal query language. For Q2, the corresponding SPARQL query looks like this:

```
SELECT ?x ?y ?z
WHERE {
    ?x rdfs:subClassOf:watch .
    ?x mmfvg:attribute:new .
    ?y rdfs:subClassOf:Person .
    ?y mmfvg:name:Jim .
    ?x mmfvg:partof:?y .
    ?z mmfvg:type:Image
}
```


By enhancing the current functionality of [48] with MMFVG attributes, nodes and links, accuracy and functionality of current solutions can be significantly improved. Queries are transformed into graph codes within the MMFVG and thus follow the query-by-example (QBE) paradigm [80], which constructs a $MMFVG_{Search}$ data structure representing the user queries. Such queries will be processed within the MMFVG to find matching results.

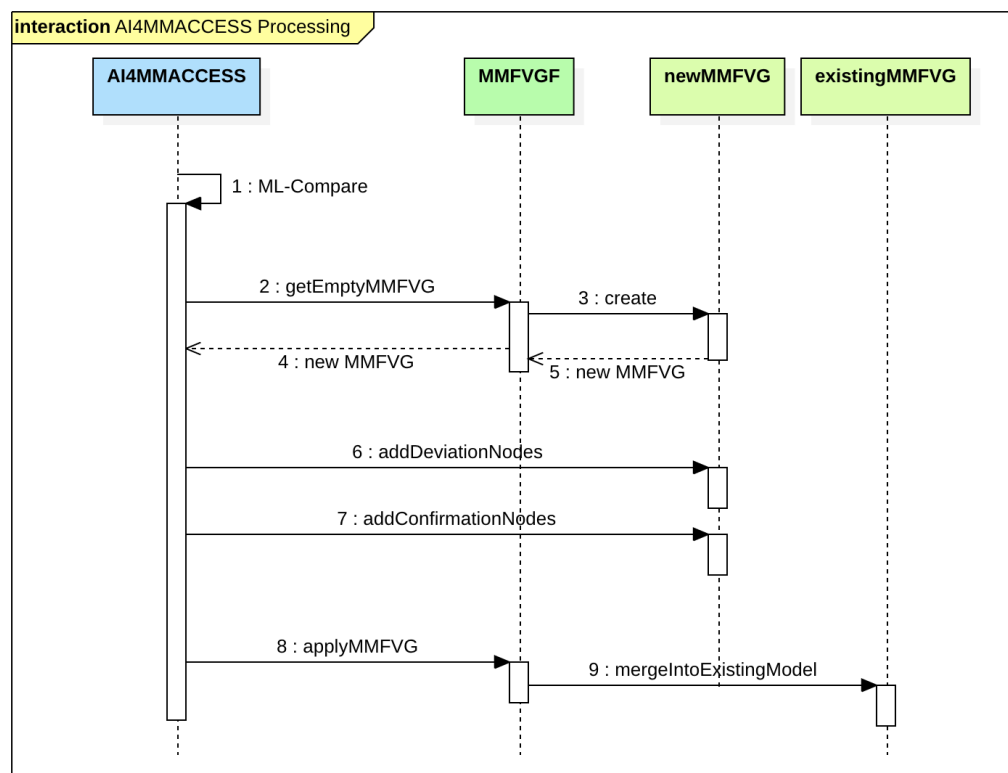


Figure 14. Processing of deviations and confirmations within the frameworks.

4.6. Swift Prototype

For use on smartphones, we are currently developing a Swift prototype to run on Apple iPads and iPhones with the goal of integrating high-performance local computing with the Java GMAF Prototype as shown in Figure 10. The Swift prototype will utilize specially optimized APIs such as that in [70] for local object detection to create a local MMFVG. If the results are not promising, the algorithm also uses the GMAF Web service to enrich them with the Java-based plugins based on Amazon [11] or Google [9].

4.7. Summary

Our prototypical implementation shows that the foundations of our approach are valid for AI-based semantic multimedia indexing and retrieval for social media. Currently, our implementation analyzes vertical levels with some selected plugins and calculates a MMFVG. The horizontal levels, including cross-referencing and the AI component to detect context-based deviations, are still subject to further investigation and implementation. Optimization and validation of our frameworks is currently ongoing, including the definition of an approach to the mapping of existing multimedia services covering data redundancy, information uncertainty and the building of new semantic relations during the aggregation process of MMFVGs. After extending the prototype with further analysis plugins, we can commence training the AI4MMRA based on calculated MMFVGs to detect context-based deviations within the MMFVGs. Finally, the prototype implementation can be adapted and optimized for smartphones.

5. Evaluation and Experiments

The current prototype of GMAF, MMFVG, AI4MMRA including the implementation of graph codes is a foundation for three experiments addressing the efficiency, effectiveness and quality of the implementation. In this section, we summarize and compare our results to other approaches.

5.1. Java Prototype Implementation

Our Java prototype implementation reflects our scientific approach with associated conceptual foundations. The implementation can facilitate extending, integrating and evaluating plugins, algorithms or concepts. The current implementation is basically a REST webservice, which processes a single image through the GMAF and MMFVG frameworks. We have also included a basic HTML form for the user to upload and process an image and obtain the MMFVG. The implementation of Video handling (including scene detection) is currently under development. The UML model including the Source Code is published on Github [91]. The Prototype contains a complete object detection implementation of Google's Vision AI [9], the MMFVG and MMFVGf implementation and the GMAF Framework. Importers and Exporters for GraphML, JSON, XML and HTML are also included. The Graph Encoding Algorithm has been implemented as an additional Exporter and provides a downloadable Bitmap file for a given sample image. The Prototype is available for test via a Java Web application [84].

5.2. Swift/UI5-Prototype

For application on smartphones, we completed a Swift implementation, building an iPhone and iPad application that focuses on local processing and integrates the Java prototype for further object detection as shown in Figure 15. Pictures taken from Google Images are for illustration purposes only. (The prototype is designed to support natural language input via speech-to-text and translate it into RDF. [83] gives an approach towards a corresponding solution, but is currently not yet implemented in the prototype.) The prototype has uncovered the following findings:

- **Software design:** The described design (see Section 3.1) has to be adapted to the requirements for Swift and iOS programming, especially regarding UI handling and requirements regarding Apple's App Development program [92]. As the prototype focuses on the capabilities of local processing, the GMAF contains only Apple plugins for local object detection such as those in [70].
- **Local processing:** Compared to the Web-based services for object detection such as [9], local detection is much faster. Local processing does not need to upload images to a remote server and wait for remote processing results, and can employ specialized hardware in smartphones such as that in [16]. While remote object detection usually takes about 25 seconds for an image to be processed, the local call is finished in milliseconds.
- **Multimedia source:** We built the Swift prototype to access the user's local image library to index and retrieve multimedia objects and to comply to Apple's standards.

The current version of the Swift prototype can also be found on Github [91].

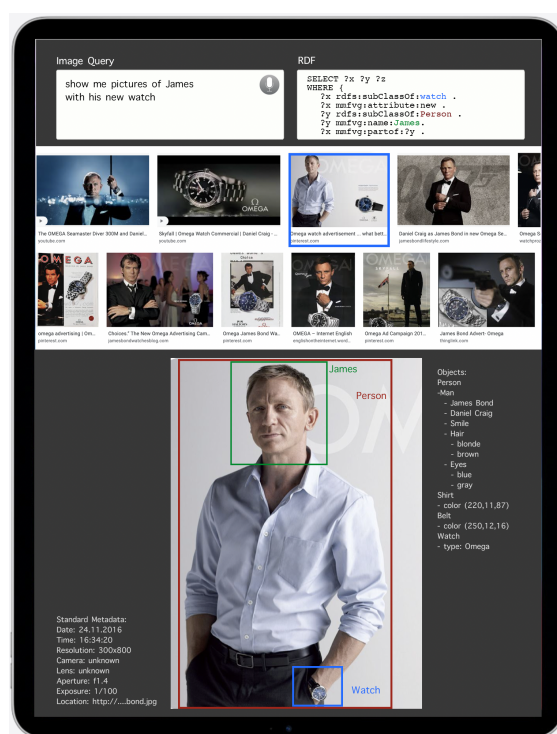


Figure 15. The Swift prototype design for GMAF, MMFVG and graph codes.

5.3. Graph Encoding Algorithm

The current graph code implementation shows that graph codes are applicable to any kind of graph structure. When the following prerequisites are met, the graph encoding algorithm is a particularly optimal approach for pattern-matching within multimedia graph structures:

- **Density:** The graph structure must be dense enough to employ as many pixels as possible. Large white spots in the graph code lead to reduced performance. For multimedia assets, this is typical, but for records such as the IMDb dataset, the performance of the graph encoding algorithm is low.
- **Ontology size:** As the graph encoding algorithm is based on a finite set of possible nodes, the size of this set directly affects performance. Our tests give promising results with ontologies including less than one million possible nodes. Dynamic ontologies based on the available content can be considered to increase the processing performance.
- **Hardware support:** The more parallelism can be employed, the better the performance achieved. In the best case, the graph encoding algorithm performs in $O(1)$. For smaller graphs, an unparallelized version can also provide suitable performance, but for larger graphs, special hardware must be employed.

It has to be stated that graph codes are not applicable for some of the typical graph operations such as shortest path or community detection. Graph codes really focus on providing a readily calculated pattern matching structure to detect deviations and similarities. Further analysis and the evaluation of graph codes on different machines such as those covered in [79] or [16] is part of our ongoing work.

5.4. AI4MMRA

In general, training an ML component to detect deviations and confirmations of given graph code images is a common task that is well described and evaluated [4,7,8,66]. Our prototypical implementation employs Apple's ML model [93] to implement AI4MMRA on smartphones. Apple's API follows common standards, so most existing approaches are

also applicable. In the current prototype, we focus on the refinement of the deviation and confirmation MMFVGs in order to update the model accordingly.

5.5. Experiments

To illustrate the validity of our modeling and implementation, we designed and built three experiments. With these experiments, we want to prove the following hypotheses (see Table 2):

Table 2. Hypotheses to be proved by the experiments.

Hypothesis	Description
A. Efficiency	The described approach of using graph codes is more efficient than current solutions based on graph traversal algorithms.
B. Effectiveness	Images that have been processed with the GMAF can deliver good semantic results.
C. Quality	Recursive processing within the GMAF leads to more levels of semantic details.

To prove hypothesis A (Efficiency), we designed Experiment 1, which compares the efficiency of graph code processing to standard graph-based algorithms of a Neo4J [94] graph database. Hypothesis B (Effectiveness) is validated with Experiment 2 by calculating precision and recall values of a random selection of images from the Flickr30k dataset [35]. Finally, Experiment 3 addresses hypothesis C (Quality) by determining the level of detail in the GMAF's object detection according to the number of recursions. For Experiments 1 and 2, we used the Flickr30k dataset [35] as an input; Experiment 3 was conducted with the uploaded image of Figure 11. Figure 16 shows the design for Experiments 1 and 2.

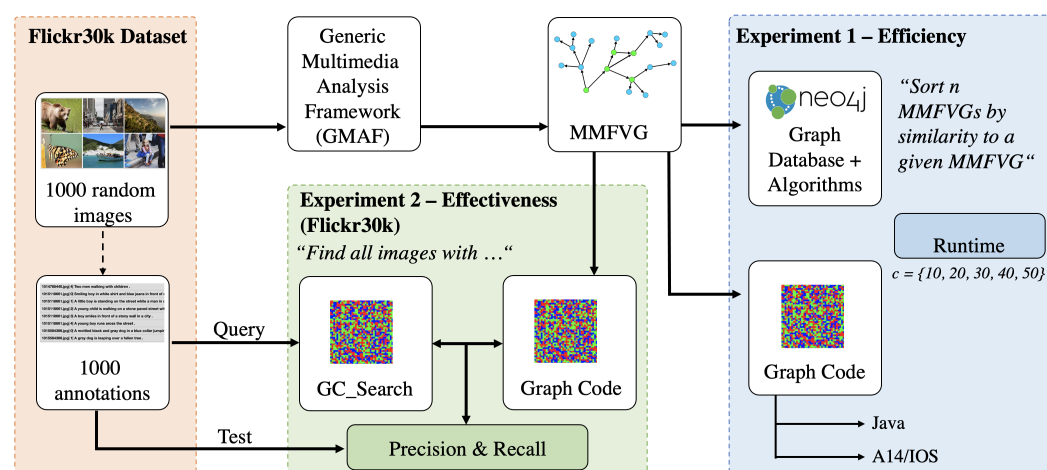


Figure 16. Design of experiments 1 and 2.

5.5.1. Experiment 1—Efficiency

This experiment focuses on the efficiency of the graph code algorithms. We employed a random selection of c images from the Flickr30k dataset [35], which contains low-resolution images including annotations. The images were processed through the GMAF to calculate the corresponding MMFVGs. For these, MMFVGs and the corresponding graph codes were generated. The same MMFVG is stored into a Neo4J graph-database [94]. Table 3 and Figure 17 show the processing results as well as the number of nodes n and the number of vertices v of the corresponding MMFVGs. The main algorithm of the experiment is a similarity calculation based on graphs and graph codes. On Neo4J side, we applied the production-ready algorithm for node similarity as follows:

```

MATCH (r1:Node {name: 'N_Root_Image_Search'})
    -[*..4]->(child1)
WITH r1, collect(id(child1)) AS r1Child
MATCH (i:Image)-[:img]->
    (r2:Node {name: 'N_Root_Image'})
    -[*..4]->(child2)
WHERE r1 <> r2
WITH r1, r1Child, r2, collect(id(child2))
    AS r2Child
RETURN r1.image AS from, r2.image AS to,
    gds.alpha.similarity.jaccard
    (r1Child, r2Child)
AS similarity

```

Experiment 1 was executed on a MacBook Pro Machine with 64GB of RAM, where both the Java implementation of graph codes and the Neo4J database were installed. As reference implementation for smartphones, we chose an Apple iPhone Pro with the A14 bionic chip [16] and implemented the graph code algorithm in Swift.

Table 3. Experiment 1—Similarity comparison of images performance on the Flickr30k dataset.

<i>c</i>	<i>n</i>	<i>v</i>	Neo4J	Java	iPhone Pro
10	326	1591	8 ms	9 ms	8 ms
20	634	3218	33 ms	18 ms	14 ms
30	885	4843	62 ms	40 ms	21 ms
40	1100	5140	196 ms	42 ms	28 ms
50	1384	7512	272 ms	48 ms	31 ms

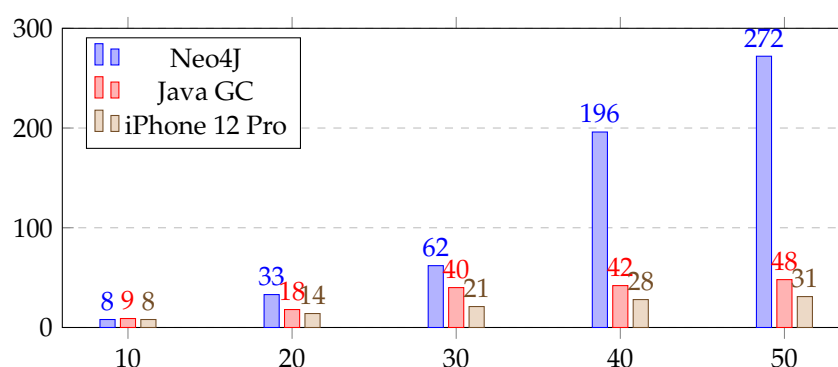


Figure 17. Experiment 1—Runtimes (see Table 3).

Results of Experiment 1 prove hypothesis A (Efficiency) by showing that graph codes outperform current graph-traversal-based algorithms by a factor of 3 to 5, and even higher factors when comparing iPhone and Neo4J results. This effect increases with the number of detected features. Of course, on the Neo4J side, some optimizations may also improve the performance, as we just used the recommended standard setup and the production algorithms (no beta versions) for similarity calculations. However, even then, graph traversal will still lead to square or exponential runtimes, whilst the processing of graph codes remains linear.

5.5.2. Experiment 2—Effectiveness

To determine the effectiveness of the graph-code-based indexing and retrieval, we selected five test queries from the annotations of a random set of 1000 Flickr30k images and calculated respective precision and recall values. For Experiment 2, we did not feed any

metadata into the GMAF, which we would normally do. Hence, the results reflect the pure object detection capabilities of the GMAF framework without any semantic enrichment. Indexing and retrieval has been performed using graph codes. Table 4 shows values of relevant *rel* in the dataset, the selection *rrel* by the retrieval algorithm, values of true positive results *tp*, values of true negative results *tn*, precision $P = \frac{tp}{rrel}$ and recall $R = \frac{tp}{rel}$. Results are also shown in Figure 18.

Table 4. Effectiveness of GMAF and graph code processing within 1000 random images from the Flickr30k dataset.

Query	<i>rel</i>	<i>rrel</i>	<i>tp</i>	<i>tn</i>	<i>P</i>	<i>R</i>
(A) “Dog”	206	190	188	2	0.98	0.91
(B) “Man”	629	398	294	104	0.76	0.47
(C) “Golf”	11	7	5	2	0.71	0.45
(D) “Guitar”	36	30	24	6	0.80	0.66
(E) “Bicycle”	66	79	60	19	0.75	0.90

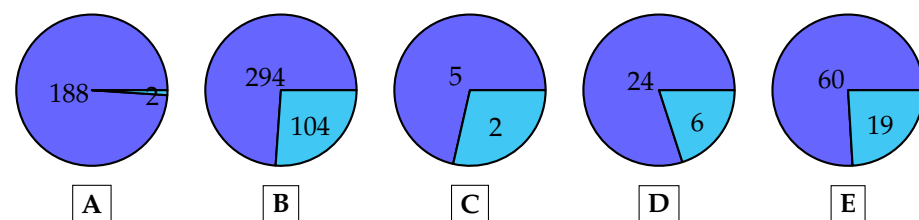


Figure 18. Experiment 2—true positives (blue), false positives (cyan). See Table 4.

These results did not meet our expectations and give lower values for precision and recall than in our own test datasets. Hence, we investigated the reasons for this and discovered that in fact, the GMAF processing has been much more accurate than the manual annotations provided with the Flickr30k dataset. Figure 19 shows an example for this: the annotation of image (a) does not mention the term “guitar”, while image (b) is wrongly annotated with “bass” instead of “guitar”. In our own manually evaluated dataset, we were able to produce precision and recall values of more than 96%. Thus, this experiment only partly proves hypothesis B (Effectiveness) due to the low quality of image annotations in the dataset. For future tests, a high-resolution dataset with accurate annotations and reference data is required.

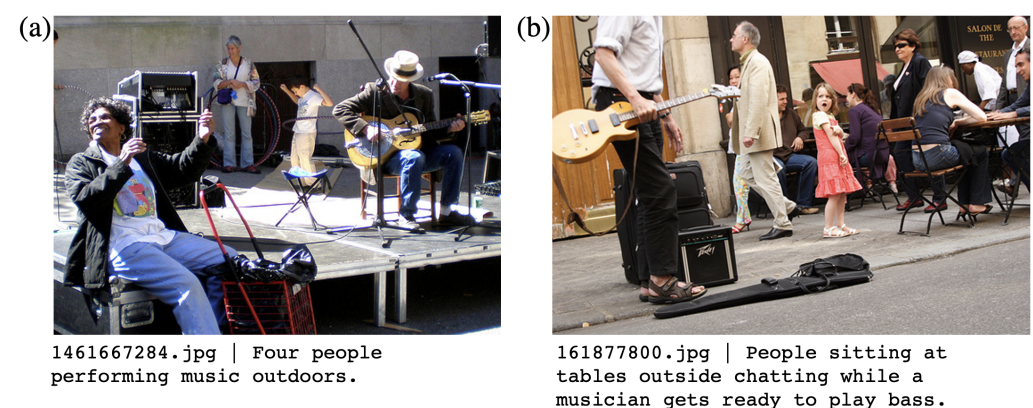


Figure 19. Example images with corresponding annotations from the Flickr30k dataset.

5.5.3. Experiment 3—Quality

The Flickr30k dataset contains only low-resolution images, which limits the level of detail of the GMAF feature detection. For efficiency and effectiveness, this is acceptable, but to measure the feature detection capabilities of the GMAF, higher resolution images must be employed. Thus, for Experiment 3, we used the high-resolution image shown already in Figure 11 to show the increase in GMAF feature detection performance according to the number of recursions. In our experiment, we processed the image through the GMAF with different numbers of recursions. The results are generated only by applying object detection, and no additional metadata have been attached to the MMFVGs, which would further increase the number of nodes and edges. For a better understanding of the results, we also printed some examples of the detected features in Table 5. If this MMFVG would be enriched with information, e.g., from the semantic Web, additional annotations and metadata or AI4MMRA processing, it would contain about 500 nodes and 5,000 vertices.

Table 5. Experiment 3—GMAF processing and object detection based on recursion iterations.

Recursions	Nodes	Edges	Detection Example
0	53	204	Person, Travel, Piazza Venezia, Joy
1	71	274	Pants, Top, Pocket, Street fashion
2	119	475	Camera, Bermuda Shorts, Shoe
3	192	789	Sun Hat, Fashion accessory, Watch, Bergen County (Logo)
4	228	956	Mouth, Lip, Eyebrow, Pocket, Maroon
4	274	1189	Leather, Grey, Single-lens reflex camera

This example proves hypothesis C (Quality) by showing that the recursive application of the GMAF processing can detect more details and provide a much higher level of semantic information.

5.5.4. Discussion

Our experiments show the capabilities of GMAF, MMFVGF and graph codes. Experiment 1 shows that using graph codes improves the performance of retrieval by at least a factor of 3 and performs best on an iPhone 12 Pro, and it proves hypothesis A (Efficiency). It also demonstrates the validity of this approach for applications on smartphones. No additional databases, libraries or frameworks are required. Graph codes also outperform current production systems such as Neo4J. Experiment 2 shows that the feature detection of the GMAF is able to produce promising precision and recall results. Unfortunately, due to a lack of fully annotated high resolution image datasets, these results had to be verified manually. Thus, hypothesis B (Effectiveness) can only be partially proved by this experiment. Finally, Experiment 3 shows the level of detail which can be produced by employing GMAF and the MMFVGF and that the aggregation of several object detection plugins and their recursive appearance is able to calculate detailed search terms for further processing. By this, hypothesis C (Quality) is also proved. All experiments and sample data are available on Github [91].

6. Summary and Relevance

Filling the semantic gap has been a research topic for years. Much research has helped to gain better basic analysis of multimedia content. However, current technologies still cannot cope with the substantial volume of generated content. The frameworks, concepts, architectures and implementations described here are designed to close the gap and to provide an extensible solution for AI-based indexing and retrieval of multimedia content with a particular focus on social media on smartphones.

The GMAF framework and the MMFVGF and AI4MMRA contribute to research and technology and provide basic components and solutions for other applications. Appli-

cations based on this technology will increase the quality of multimedia querying and retrieval for the user and/or the service providers such as social media. Graph codes may become an important part of multimedia processing on smartphones, as they bring a new approach to employing mobile hardware in a particular context.

Integrating more plugins into the GMAF, optimizing the frameworks for smartphones, training AI4MMRA and evaluating the performance of graph codes on different machines is part of our ongoing work and will be published on Github [91] and other channels.

In summary, our approach provides a solid basis for further research, optimisation and technologies, which specialize on smartphones, employ the existing hardware in an optimal way and provide an extensible and modular set of frameworks for current and future applications of multimedia indexing and retrieval. The presented approach contributes to indexing and retrieval due to better performance and better accuracy and performs well on smartphones. The research results on graph codes can also be applied to other graph-based algorithms not related to multimedia. Thus, this paper describes a solid basis and implementation for future research and technologies in numerous application areas.

Author Contributions: Conceptualization and methodology: Stefan Wagenpfeil and Matthias Hemmje. Software, validation, formal analysis, investigation, resources, data curation, writing: Stefan Wagenpfeil. Review, editing and supervision: Felix Engel, Paul Mc Kevitt and Matthias Hemmje. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available in [91].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nudelman, M. Smartphones Cause a Photography Boom. 2020. Available online: <http://www.businessinsider.com/12-trillion-photos-to-be-taken-in-2017-thanks-to-smartphones-chart-2017-8> (accessed on 23 August 2020).
2. Mazhelis, O. Impact of Storage Acquisition Intervals on the Cost-Efficiency of the Private vs. Public Storage. In Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, 24–29 June 2012; pp 646–653.
3. Apple.com. iCloud—the Best Place for Photos. 2020. Available online: <http://www.apple.com/icloud/> (accessed on 23 August 2020).
4. Beyerer, J. *Pattern Recognition-Introduction*; Walter de Gruyter GmbH & Co KG: Berlin, Germany, 2017; pp. 10–37, ISBN 978-3-110-53794-9.
5. Bond, R.R.; Engel, F.; Fuchs, M.; Hemmje, M.; McKeivitt, P.M.; McTear, M.; Mulvenna, M.; Walsh, P.; Zheng, H. Digital empathy secures Frankenstein’s monster. In Proceedings of the 5th Collaborative European Research Conference (CERC 2019), Hochschule Darmstadt, University of Applied Sciences, Faculty of Computer Science, Darmstadt, Germany, 29–30 March 2019; Volume 2348, pp. 335–349.
6. Beierle, C. *Methoden Wissensbasierter Systeme-Grundlagen*; Springer-Verlag: Berlin/Heidelberg, Germany; New York, NY, USA, 2019; pp. 89–157, ISBN 978-3-658-27084-1.
7. Goodfellow, I. *Deep Learning*; MIT Press: Cambridge, UK, 2016; ISBN 978-0-262-03561-3.
8. Heaton, J. *Deep Learning and Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 1–25, ISBN 978-1505714340.
9. Google.com. Google Vision AI—Derive Insights from Images. 2020. Available online: <http://cloud.google.com/vision> (accessed on 23 August 2020).
10. Microsoft.com. Machine Visioning. 2020. Available online: <http://azure.microsoft.com/services/cognitive-services/computer-vision> (accessed on 23 August 2020).
11. Amazon.com. Amazon Recognition. 2020. Available online: <http://aws.amazon.com/recognition> (accessed on 23 August 2020).
12. MIT—Massachusetts Institute of Technology. *Description of Exif File Format*. 2020. Available online: <http://media.mit.edu/pia/Research/deepview/exif.html> (accessed on 23 August 2020).
13. Chang, S. Overview of the MPEG-7 standard. *IEEE Trans. Circuits Syst. Video Technol.* **2001**, *11*, 688–695.
14. Google.com. Google Knowledge Search API. 2020. Available online: <http://developers.google.com/knowledge-graph> (accessed on 23 August 2020).
15. W3C.org. W3C Semantic Web Activity. 2020. Available online: <http://w3.org/2001/sw> (accessed on 23 August 2020).
16. Wikipedia.com. Apple A14 Bionic. 2020. Available online: https://en.wikipedia.org/wiki/Apple_A14 (accessed on 28 October 2020).

17. Wikipedia.com. Information Age. 2020. Available online: https://en.wikipedia.org/wiki/Information_Age (accessed on 28 October 2020).
18. Storage Newsletter. Total WW Storage Data at 6.8ZB in 2020. Available online: <https://www.storagenewsletter.com/2020/05/14/total-ww-storage-data-at-6-8zb-in-2020-up-17-from-2019-idc/> (accessed on 24 October 2020).
19. Wikipedia.com. Bandwidth (Computing). 2020. Available online: [https://en.wikipedia.org/wiki/Bandwidth_\(computing\)](https://en.wikipedia.org/wiki/Bandwidth_(computing)) (accessed on 27 October 2020).
20. Wikipedia.com. 5G. 2020. Available online: <https://en.wikipedia.org/wiki/5G> (accessed on 27 October 2020).
21. Avola, D. Low-Level Feature Detectors and Descriptors for Smart Image and Video Analysis: A Comparative Study. *Intelligent Systems Reference Library*; Springer: Berlin/Heidelberg, Germany: 2018; pp. 7–29; ISBN 978-3-319-73890-1.
22. Kankanhalli, M. Video modeling using strata-based annotation. *IEEE Multimed.* **2000**, *7*, 68–74.
23. Wagenpfeil, S. Towards AI-bases Semantic Multimedia Indexing and Retrieval for Social Media on Smartphones; In Proceedings of the 2020 15th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP 2020 Conference Paper), Zakynthos, Greece, 29–30 October 2020.
24. FFMpeg.org. ffmpeg Documentation. 2020. Available online: <http://ffmpeg.org> (accessed on 23 August 2020).
25. Open Images. Overview of Open Images V6. 2020. Available online: <http://storage.googleapis.com/openimages/web/factsfigures.html> (accessed on 23 August 2020).
26. Adobe.com. Work with Metadata in Adobe Bridge. 2020. Available online: <http://helpx.adobe.com/bridge/using/metadata-adobe-bridge.html> (accessed on 23 August 2020).
27. EBU Recommendations. Material Exchange Format. 2007. Available online: <http://mx.f.irt.de/information/eburecommendations/R121-2007.pdf> (accessed on 23 August 2020).
28. Apple.com. Siri for Developers. 2020. Available online: <https://developer.apple.com/siri/> (accessed on 23 August 2020).
29. Microsoft.com. Your Personal Productivity Assistant in Microsoft 365. Available online: <https://www.microsoft.com/en-us/cortana> (accessed on 14 November 2020).
30. Amazon.com. Amazon Alexa Home. Available online: <https://developer.amazon.com/en-US/alexa> (accessed on 11 December 2020).
31. Domingue, J. Introduction to the Semantic Web Technologies. 2011. Available online: <https://doi.org/10.1007/978-3-540-92913-0> (accessed on 02 December 2020).
32. Kwasnicka. *Bridging the Semantic Gap in Image and Video Analysis*; Springer: Berlin, Germany, 2018; pp. 97–118, ISBN 978-3-319-73891-8.
33. Mc Kevitt, P. MultiModal semantic representation. In *SIGSEM Working Group on the Representation of MultiModal Semantic Information*; Bunt, H., Lee, K., Romary, L., Krahmer, E., Eds.; Tilburg University: Tilburg, Netherlands, 2003.
34. Spyrou. *Semantic Multimedia Analysis and Processing*; CRC Press: Boca Raton, FL, USA, 2017; pp. 31–63, ISBN 978-1-351-83183-3.
35. MIRFlickr25000 dataset. The MIRFlickr Retrieval Evaluation. 2020. Available online: <http://press.liacs.nl/mirflickr> (accessed on 23 August 2020).
36. Scherer, R. *Computer Vision Methods for Fast Image Classification and Retrieval*; Polish Academy of Science: Warsaw, Poland, 2020; pp. 33–134, ISBN 978-3-030-12194-5.
37. Nixon, M. *Feature Extraction and Image Processing for Computer Vision*; Academic Press Elsevier: Cambridge, MA, USA, 2020.
38. Bhute, B. Multimedia Indexing and Retrieval Techniques: A Review. *Int. J. Comput. Appl.* **2012**, *58*, 35–42.
39. Wikipedia.com. Image Histograms. Available online: https://en.wikipedia.org/wiki/Image_histogram (accessed on 10 October 2020).
40. Wikipedia.com. Color Histograms. Available online: https://en.wikipedia.org/wiki/Color_histogram (accessed on 10 October 2020).
41. Wikipedia.com. Fast Fourier Transformation. Available online: https://en.wikipedia.org/wiki/Fast_Fourier_transform (accessed on 13 November 2020).
42. Smeulders, A. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1349–1380.
43. Gurski, F. On Characterizations for Subclasses of Directed Co-Graphs. 2019. Available online: <http://arxiv.org/abs/1907.00801> (accessed on 24 August 2020).
44. Needham, M. *Graph Algorithms, Practical Examples in Apache Spark and Neo4j*; O'Reilly: Newton, MA, USA, 2020; ISBN 978-1-492-05781-9.
45. Robinson, I. *Graph Databases*; O'Reilly: Newton, MA, USA; ISBN 978-1-491-93089-2.
46. Jiezhong, Q. Network Embedding as Matrix Factorization: Unifying DeepWalk. 2017. Available online: <http://arxiv.org/abs/1710.02971> (accessed on 24 September 2020).
47. Bai, Y.; Ding, H.; Bian, S.; Chen, T.; Sun, Y.; Wang, W. SimGNN: A Neural Network Approach to Fast Graph Similarity Computation. 2019. Available online: <https://doi.org/10.1145/3289600.3290967> (accessed on 24 November 2020).
48. W3C.org. SPARQL Query Language for RDF. 2013. Available online: <https://www.w3.org/TR/sparql11-overview/> (accessed on 23 August 2020).
49. Nkgau, T. Graph similarity algorithm evaluation. In Proceedings of the 2017 Computing Conference, London, UK, 18–20 July 2017; pp. 272–278.

50. Sciencedirect.com. Adjacency Matrix. Available online: <https://www.sciencedirect.com/topics/mathematics/adjacency-matrix> (accessed on 17 December 2020).
51. Fischer, G. *Linear Algebra*; Springer Spektrum Wiesbaden: Berlin/Heidelberg, Germany; pp. 143–163, ISBN 978-3-658-03945-5.
52. Yuan, Y. Graph Similarity Search on Large Uncertain Graph Databases. 2015. Available online: <https://doi.org/10.1007/s00778-014-0373-y> (accessed on 14 December 2020).
53. Samir, S. Seo For Social Media: It ranked first in the search engines. Kindle Edition, ASIN: B08B434ZM2 (accessed on 14 October 2020).
54. Bultermann, D. Socially-Aware Multimedia Authoring: Past. *Acm Trans. Multimed. Comput. Commun. Appl.* **2013**.
55. Krig, S. *Interest Point Detector and Feature Descriptor Survey*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 187–246; ISBN 978-3-319-33761-6.
56. Hannane, R.; Elboushaki, A.; Afdel, K.; Naghabhushan, P.; Javed, M. An efficient method for video shot boundary detection and keyframe extraction using SIFT-point distribution histogram. *Int. J. Multimed. Inf. Retr.* **2016**, *5*, 89–104.
57. Sluzek, A. *Local Detection and Identification of Visual Data*; LAP LAMBERT Academic Publishing: Saarbrücken, Germany, 2013.
58. Sevak, J.S.; Kapadia, A.D.; Chavda, J.B.; Shah, A.; Rahevar, M. Survey on semantic image segmentation techniques. In Proceedings of the 2017 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 7–8 December 2017; pp. 306–313.
59. Wang, G. *Efficient Perceptual Region Detector Based on Object Boundary*; Springer: Cham, Switzerland, 2016; pp. 66–78, ISBN 978-3-319-27673-1.
60. Ballan, L.; Bertini, M.; Del Bimbo, A.; Seidenari, L.; Serra, G. Event Detection and Recognition for Semantic Annotation of Video. 2011. Available online: <https://doi.org/10.1007/s11042-010-0643-7> (accessed on 11 December 2020).
61. Arndt, R.; Troncy, R.; Staab, S.; Hardman, L. COMM: A Core Ontology for Multimedia Annotation. *J. Comb. Theory* **2008**, 403–421.
62. Ni, J.; Qian, X.; Li, Q.; Xu, X. Research on Semantic Annotation Based Image Fusion Algorithm. In Proceedings of the 2017 International Conference on Computer Systems, Electronics and Control (ICCSEC), Dalian, China, 25–27 December 2017; pp. 945–948.
63. Gayathri, N. An Efficient Video Indexing and Retrieval Algorithm using Ensemble Classifier. In Proceedings of the 2019 4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECOT), Karnataka, India, 13–14 December 2019; pp. 250–258.
64. Zhao, F. Learning Specific and General Realm Feature Representations for Image Fusion. *IEEE Trans. Multimed.* **2020**, *1*.
65. Hu, Q.; Wu, C.; Chi, J.; Yu, X.; Wang, H. Multi-level Feature Fusion Facial Expression Recognition Network. In Proceedings of the 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 23–25 May 2020; pp. 5267–5272.
66. Goh, K.; Li, B.; Chang, E.Y. Semantics and Feature Discovery via Confidence-Based Ensemble. 2005. Available online: <https://doi.org/10.1145/1062253.1062257> (accessed on 24 November 2020).
67. Draper, N. *User Centered System Design-New Perspectives on Human-computer Interaction*; Taylor & Francis, Justus-Liebig-Universität: Oxfordshire, UK, 1986; ISBN 978-0-898-59872-8.
68. Nunamaker, J. Systems Development in Information Systems Research. *J. Manag. Inf. Syst.* **1991**, 89–106.
69. Fowler, M. *UML Distilled-A Brief Guide to the Standard Object Modeling Language*; Addison-Wesley Professional: Boston, MA, USA, 2004; ISBN 978-0-321-19368-1.
70. Apple.com. Face Recognition in Apple Fotos. 2020. Available online: <https://support.apple.com/de-de/guide/photos/phtad9d981ab/mac> (accessed on 23 August 2020).
71. Iyer, G. A Graph-Based Approach for Data Fusion and Segmentation of Multimodal Images. *IEEE Trans. Geosci. Remote. Sens.* **2020**, 1–11.
72. yWorks GmbH. yEd Graph Editor. 2020. Available online: <https://www.yworks.com/products/yed> (accessed on 23 August 2020).
73. Wikipedia.org. Barcodes. Available online: <https://en.wikipedia.org/wiki/Barcode> (accessed on 11 December 2020).
74. Wikipedia.org. QR Codes. Available online: https://en.wikipedia.org/wiki/QR_code (accessed on 11 December 2020).
75. Aggarwal, C. *Linear Algebra and Optimization for Machine Learning: A Textbook*; Springer Publishing: Cham, Switzerland, 2020; ISBN 978-3030403430.
76. Foster, I. *Designing and Building Parallel Programs*; Addison Wesley: Boston, MA, USA, 1995; ISBN 0-201-57594-9.
77. Planche, B. *Computer Vision with TensorFlow 2*; Packt Publishing: Birmingham, UK, 2019; pp. 77–99, ISBN 978-1-78883-064-5.
78. Tuomanen, B. *GPU Programming with Python and CUDA*; Packt Publishing: Birmingham, UK, 2018; pp. 101–184, ISBN 978-1-78899-391-3.
79. Nvidia.com. RTX 2080. Available online: <https://www.nvidia.com/de-de/geforce/graphics-cards/rtx-2080/> (accessed on 10 November 2020).
80. Schmitt, I. WS-QBE: A QBE-Like Query Language for Complex Multimedia Queries. In Proceedings of the 11th International Multimedia Modelling Conference, Melbourne, Australia, 12–14 January 2005; pp. 222–229.
81. Nudelman, M. Smartphones Cause a Photography Boom. 2020. Available online: <http://www.businessinsider.com/12-trillion-photos-to-be-taken-in-2017-thanks-to-smartphones-chart-2017-8> (accessed on 23 August 2020).
82. Dufour, R.; Esteve, Y.; Deléglise, P.; Béchet, F. Local and global models for spontaneous speech segment detection and characterization. In Proceedings of the 2009 IEEE Workshop on Automatic Speech Recognition & Understanding, Meran, Italy, 13–17 December 2010; pp. 558–561.

-
83. Jung, H. Automated Conversion from Natural Language Query to SPARQL Query. 2020. Available online: <https://doi.org/10.1007/s10844-019-00589-2> (accessed on 24 October 2020).
 84. Wagenpfeil, S. GMAF Prototype. 2020. Available online: <http://diss.step2e.de:8080/GMAFWeb/> (accessed on 23 August 2020).
 85. Apache Software Foundation. Apache Commons Imaging API. 2020. Available online: <https://commons.apache.org/proper/commons-imaging/> (accessed on 23 August 2020).
 86. Oracle.com. Java Enterprise Edition. 2020. Available online: <https://www.oracle.com/de/java/technologies/java-ee-glance.html> (accessed on 23 August 2020).
 87. Docker.Inc. What Is a Container. 2020. Available online: <https://www.docker.com/resources/what-container> (accessed on 23 August 2020).
 88. Adobe.com. Adobe Stock. 2020. Available online: <https://stock.adobe.com> (accessed on 2 October 2020).
 89. Bornschlegel, M. IVIS4BigData: A Reference Model for Advanced Visual Interfaces Supporting Big Data Analysis in Virtual Research Environments. In *AVI Workshop on Big Data Applications*; Springer: Cham, Switzerland, 2020.
 90. EDISON Project-European Union's Horizon 2020 research-grant agreement No. 675419. Available online: <https://cordis.europa.eu/project/id/675419/de> (accessed on 14 December 2020).
 91. Wagenpfeil, S. Github Repository of GMAF and MMFVG. 2020. Available online: <https://github.com/stefanwagenpfeil/GMAF/> (accessed on 25 September 2020).
 92. Apple.com. Apple Development Programme. Available online: <http://developer.apple.com> (accessed on 21 November 2020).
 93. Apple.com. Apple Machine Learning. Available online: <https://developer.apple.com/machine-learning/> (accessed on 24 November 2020).
 94. Neo4J.com. *Neo4J Graph Database*. Available online: <https://neo4j.com/> (accessed on 24 November 2020).