



**CZECH TECHNICAL  
UNIVERSITY  
IN PRAGUE**

**F8**

**Faculty of Information Technology  
Department of Applied Mathematics**

**Dissertation Thesis**

# **Knowledge Extraction from Multimedia Content**

**with focus on explainability and processing speed**

**Petr Pulc**

A dissertation thesis submitted to the Faculty of Information Technology, Czech Technical University in Prague, in partial fulfillment of the requirements for the degree of Doctor.

**Dissertation degree study programme: Informatics  
Supervisor: prof. RNDr. Ing. Martin Holeňa, CSc.  
Prague, February 2022**

Department of Applied Mathematics  
Faculty of Information Technology  
Czech Technical University in Prague  
Thákurova 9  
160 00 – Prague 6  
Czech Republic

Copyright © 2022 Petr Pulc

## Acknowledgement / Declaration

I would like to express my gratitude to my wife for providing me with shelter, to my supervisor prof. Martin Holeňa for all doctorate-related support, to Martin Kopp, Ph.D. and Lukáš Bajer, Ph.D. for tremendous support in many other areas and all my collaborators – Eric Rosenzweig, Tomáš Šabata, Michal Kopp, Jiří Tumpach, Jiří Kožusznik, Lukáš Korel, Matěj Fanta, Oliver Kerul-Kmec, and many others – for shaping our research towards this Thesis.

The research reported in this thesis has been supported by the Czech Science Foundation (GAČR) grants 13-17187S, 17-01251 and 18-18080S, Grant Agency of the Czech Technical University in Prague SGS17/210/OHK3/3T/18, and by the Institutional Support for Long-term Conceptual Development of Research Organization programme, provided by Ministry of Education, Youth and Sports, Czech Republic.

Computational resources were supplied by the project “e-Infrastruktura CZ” (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on February 28th 2022.

.....

## Abstrakt / Abstract

Kdo, co, kde, kdy, jak a proč? Záznam odpoví na tyto základní otázky dokáže objasnit okolnosti děje v reálném světě. V případě fotografií, nebo obrazového záznamu ale dokážeme na tyto otázky odpovědět jen velmi obecně, pokud neznáme kontext, nebo záznam nevyvolá přímou vzpomínku.

Naopak při sestavování narativu z jednotlivých audiovizuálních klipů potřebujeme mít odpovědi na tyto otázky k dispozici, a následně je také předat divákovi ve srozumitelné formě. Jenže velmi často strojově čitelné informace obsahující odpovědi alespoň na část těchto otázek chybí. Z toho důvodu se metody automatického získávání znalostí z obrazových dat, pomáhajících poskytnout některé odpovědi, staly zásadní motivací pro tuto práci.

Celou situaci navíc značně komplikuje to, že multimédia jsou značně rozmanitou doménou. V jednom audiovizuálním projektu se může snadno mísit několik typů obsahu, kdy každý vyžaduje jiné metody zpracování. Za tímto účelem jsme tak navrhli teoretický systém využívající principy meta-učení.

Jako jeden ze základních kamenů pak uvažujeme metodu rozpoznání scény, na základě jejíž znalosti bychom dokázali lépe vybrat navazující metody zpracování. Ani tento problém však nemá triviální řešení a tak navrhujeme další pokročilé metody vedoucí až k problému sledování objektů ve scéně; metody, jejíž urychlení a zpřesnění by následně pomohlo nejen rozpoznávání scény ale také pro zvýšení výkonu metod, které by popisovaly samotné objekty ve scéně.

**Klíčová slova:** Zpracování obrazu, Kombinace klasifikačních metod, Metaučení, Hledání struktury v datech, Rozpoznávání scény, Sledování objektů

Who, what, where, when, how, and why? Recording the answers to these basic questions can shed light on the circumstances of real-world events. However, in the case of photos or video recordings, we can only answer these questions vaguely unless we know the context or it triggers a direct memory.

On the other hand, when constructing a narrative from individual audiovisual clips, we need to have the answers to these questions available to convey them to the viewer in an understandable form. Yet, the machine-readable information containing answers to at least some of these questions is commonly missing. For this reason, methods for automatic knowledge extraction from image data that provide some of the answers have become a significant motivation for this dissertation thesis.

The whole situation is further complicated because multimedia is a highly diverse domain. Several types of content can easily be mixed in a single audiovisual project, each requiring different processing methods. To this end, we propose a theoretical system that uses meta-learning principles.

We then consider scene recognition as one of the cornerstone methods, allowing us to select better downstream processing methods. However, even this problem does not have a trivial solution, and so we propose additional advanced methods leading up to the problem of tracking objects in the scene; methods whose speed-up and refinement would subsequently help not only the scene recognition but also increase the performance of methods that describe objects in the scene.

**Keywords:** Image processing, Classification combiner, Metalearning, Data structure discovery, Scene recognition, Visual object tracking

# Contents /

<b>1 Introduction</b>	<b>1</b>		
1.1 Personal Motivation . . . . .	2		
1.2 Multimedia archive and index . . . . .	4		
1.3 NARRA – Platform for Open Narratives . . . . .	8		
1.4 From a Book to a Movie (and back) . . . . .	10		
1.5 Goal of our Research . . . . .	13		
1.6 Contributions . . . . .	14		
1.7 Structure of the Thesis . . . . .	15		
<b>2 Multimedia Acquisition</b>	<b>17</b>		
2.1 Audio . . . . .	17		
2.1.1 Digitisation . . . . .	18		
2.1.2 Lossy Compression . . . . .	18		
2.2 Video . . . . .	19		
2.2.1 Lens . . . . .	19		
2.2.2 Filtering . . . . .	19		
2.2.3 Color Sensing . . . . .	20		
2.2.4 Shutter and Gain . . . . .	21		
2.2.5 Lossy Compression . . . . .	21		
2.3 Metadata . . . . .	22		
<b>3 Theoretical Background</b>	<b>24</b>		
3.1 Dimensionality Reduction . . . . .	24		
3.1.1 Bag-of-words . . . . .	24		
3.1.2 Bag-of-concepts . . . . .	25		
3.1.3 Embeddings . . . . .	25		
3.1.4 Histograms . . . . .	27		
3.2 Distance Measures . . . . .	27		
3.2.1 Euclidean Distance . . . . .	27		
3.2.2 Manhattan Distance . . . . .	27		
3.2.3 Cosine Distance . . . . .	28		
3.2.4 Hamming Distance . . . . .	28		
3.2.5 Dynamic Time Warping . . . . .	28		
3.3 Convolution . . . . .	29		
3.4 Deconvolution . . . . .	29		
3.5 Fourier Transform . . . . .	30		
3.6 Discrete Cosine Transform . . . . .	31		
<b>4 State-of-the-art Multimedia Processing Methods</b>	<b>32</b>		
4.1 Text Extraction and Processing . . . . .	32		
4.2 Audio Signal Processing . . . . .	32		
4.2.1 Automated Speech Recognition . . . . .	32		
4.2.2 Audio Fingerprinting . . . . .	33		
4.3 Still Image Processing . . . . .	34		
4.3.1 Edge Detection . . . . .	34		
4.3.2 Interest Point Detection and Description . . . . .	35		
4.3.3 Fiducial Marker Detection . . . . .	37		
4.3.4 Object Detection . . . . .	38		
4.3.5 Image Classification . . . . .	40		
4.3.6 Scene Recognition . . . . .	41		
4.3.7 Image Segmentation . . . . .	41		
4.4 Video Processing . . . . .	42		
4.4.1 Individual Frame Reconstruction . . . . .	42		
4.4.2 Optical Character Recognition . . . . .	43		
4.4.3 Video Shot Detection and Shot Sequence Segmentation . . . . .	43		
4.4.4 Frame Registration and Action Recognition . . . . .	43		
4.4.5 Simultaneous Localization and Mapping . . . . .	44		
4.5 Knowledge Extraction as a Tool . . . . .	44		
4.5.1 Multimedia (re)Discovery . . . . .	45		
4.5.2 Content Deduplication and Re-use Detection . . . . .	46		
4.5.3 Recommender systems . . . . .	47		
4.5.4 Online Annotation . . . . .	48		
<b>5 Efficient Multimodal Classification</b>	<b>49</b>		
5.1 Classifier Fusion and Meta-learning . . . . .	49		
5.1.1 Monolithic Classifier . . . . .	49		
5.1.2 Classifier Fusion . . . . .	50		
5.1.3 Double Fusion . . . . .	50		
5.1.4 Meta-learning . . . . .	51		
5.2 Meta-learning Framework for Multimedia Data . . . . .	52		
5.3 Search for Structure in Multimedia Data . . . . .	54		
5.4 Rule Extraction from Multimedia Data . . . . .	58		
5.5 Steps Towards Efficient Multimedia Classification . . . . .	60		

<b>6 Video Scene Recognition</b>	<b>61</b>
6.1 Video Scene Classification from Statistical Features . . . .	62
6.2 Video Scene Classification with Neural Networks . . . . .	64
6.3 Challenges in Video Scene Classification . . . . .	67
6.3.1 Color Invariance . . . . .	67
6.3.2 Incorporating Time Insensitivity . . . . .	68
6.3.3 Partial Scene Matching . .	68
6.3.4 Foreground Object Removal . . . . .	69
<b>7 Local Visual Features in Multimedia Processing</b>	<b>71</b>
7.1 Methods of Visual Feature Matching . . . . .	71
7.1.1 Kanade-Lucas-Tomasi . . .	71
7.1.2 Descriptor Distance Matching . . . . .	71
7.2 Hierarchical Visual Feature Matching . . . . .	74
7.3 Foreground Segmentation based on Estimated Motion . .	77
<b>8 Multiple Object Tracking</b>	<b>78</b>
8.1 Tracking by Detection . . . . .	78
8.2 Feature Tracking and Object Detection . . . . .	79
8.3 Fine-tuning a Re-identification Model on a Custom Dataset . . . . .	80
<b>9 Conclusions</b>	<b>82</b>
<b>References</b>	<b>85</b>
Reviewed Publications of the Author Relevant to the Thesis . . . . .	94
Diploma Theses Supervised by the Author Relevant to the Thesis . . .	95
Remaining Publications of the Author Relevant to the Thesis . . . . .	95

<b>A Week of Science and Technology Lecture Dataset</b>	<b>97</b>
<b>B The Big Bang Theory Scene Dataset</b>	<b>99</b>

## Tables / Figures

<b>4.1</b> Discrete gradient detectors.....	35	<b>1.1</b> User interface of NARRA. ....	8
<b>4.2</b> Edge information encoding. ...	36	<b>1.2</b> Visualization of text document similarity in NARRA. ....	9
<b>5.1</b> Comparison of rule extraction methods.....	60	<b>1.3</b> Simplified feature film creation and approaches to extract semantic information.....	11
<b>6.1</b> Accuracy of a set of scene classifiers over the collected histograms. ....	64	<b>2.1</b> Photographic lens internal structure. ....	19
<b>6.2</b> Improvement of scene classification by semi-supervised and active learning. ....	64	<b>2.2</b> Optical low-pass filter. ....	20
<b>6.3</b> Comparison of single-frame feature combination approaches for scene classification. ....	66	<b>2.3</b> Colour sensing in digital cameras. ....	20
<b>7.1</b> Comparison of processing time of KLT and our ORB matcher on one pair of frames. .	74	<b>2.4</b> Common chroma subsampling.....	22
		<b>3.1</b> Truncated Latent Semantic Indexing.....	25
		<b>3.2</b> Example of word2vec skip-gram training. ....	26
		<b>4.1</b> Canny edge detector. ....	34
		<b>4.2</b> SIFT, SURF and ORB.....	37
		<b>4.3</b> Interest point registration.....	38
		<b>4.4</b> Haar-like features.....	39
		<b>4.5</b> YOLO and Mask R-CNN. ....	40
		<b>4.6</b> U-net architecture. ....	42
		<b>5.1</b> Feature-space coverage of a single classifier.....	49
		<b>5.2</b> Feature-space coverage of a classifier ensemble. ....	50
		<b>5.3</b> Division of feature-space by meta-learning. ....	51
		<b>5.4</b> Hierarchical meta-learning framework.....	52
		<b>5.5</b> Meta-learning framework building blocks.....	53
		<b>5.6</b> Self-organizing map and hierarchical clustering quality....	57
		<b>6.1</b> Comparison of color histograms collected from clips recorded in two different livingrooms. ....	62
		<b>6.2</b> Network architecture for scene classification from color histograms. ....	63
		<b>6.3</b> Preparation of video sequence for neural scene classification from multiple images.....	65

<b>6.4</b>	Changes in color tonality over time.....	68
<b>6.5</b>	Changes in visual content over time.....	69
<b>6.6</b>	Object removal with image inpainting.....	70
<b>6.7</b>	Background extraction by video frame stitching.....	70
<b>7.1</b>	Comparisson of extracted motion vectors from ORB, SIFT and SURF visual features.....	73
<b>7.2</b>	Hierarchical visual feature matching.....	75
<b>7.3</b>	Scheme of hierarchical ORB tracker.....	76
<b>7.4</b>	Tracklets extracted by hierarchical visual feature matcher.....	76
<b>7.5</b>	Error of hierarchical visual feature matcher over time.....	76
<b>7.6</b>	Result of frame segmentation based on clustered motion vectors.....	77
<b>8.1</b>	Visualization of object tracking by neural object detector and our visual feature tracker. .	79
<b>8.2</b>	Cosine metric model retraining precision.....	81
<b>8.3</b>	Fine tuning of the cosine metric with our dataset.....	81



# Chapter 1

## Introduction

People like to share stories and record them for the future.

First, we recorded our surroundings by drawing pictures and creating sculptures, which allowed us to capture object appearance and basic actions, yet (without additional context) they lack the power to tell complex stories.

Then we adopted writing systems that allowed us to capture and safely record increasingly complex thoughts. However, complex written text requires a skilled writer and a reader who shares the writing system, language, and cultural background. Reading of written stories also requires a significant amount of imagination, which may be only aided by illustrations, effectively creating the first multimedia documents.

Thanks to the technical advancements in the 19th century, we started recording reality directly. The introduction of photography removed the need to copy the visual appearance by hand or carefully selected words. Similarly, the advances in sound capture provided an alternative to describing sounds with onomatopoeia and relying on music scores. Soon after, it was possible (at least in theory) to combine a sequential series of still images with a sound recording covering the accompanying music and noises of the environment. In the late 1930s, the recording started including dialogue and narration. However, in the beginning, there was no reliable way of synchronizing these two modalities during playback.

Acquiring multiple pictures per second then enabled the movie artists to capture the action and motion of objects in the scene. Much later, by introducing an editing process, the developed film strip, possibly from multiple camera positions, could be spliced to capture at least some of the story without explicit narration. However, until the introduction of music film, most of the narrative and dialogues were still provided in inserted graphics – intertitles. A reliable method of storing the audio signal alongside the visual footage and playing it back in a synchronized manner was introduced only at the beginning of the 20th century.

The equipment needed for producing such records was, however, expensive. And not only the recording but also the postprocessing of the material required significant technical skill in multiple areas. While this might be still true for professional movie production, which closely follows the traditional approaches based on film stock, miniaturization in consumer electronics allowed us to carry a high-quality camera in our pockets at all times, including a storage capacity to record several hours of video material or even a possibility to broadcast the compressed video and audio signal over the internet to an audience located anywhere on the planet with a delay of just a couple of seconds. Moreover, the technology is increasingly accessible in terms of cost and ease of use. Nowadays, with a smartphone in our hand, we are usually just two clicks away from taking a photo or recording a video clip and sharing the result with the rest of the world.

Current accessibility of equipment for content creation and sharing platforms allows anyone to produce massive amounts of animations, raw video footage, edited content, copies, compilations, reaction videos, music videos, mashups, gameplay recordings, and

other multimedia content to such extent that any multimedia archive is getting increasingly harder to navigate. The diversity of the published content also causes an issue, as many subgenres of the multimedia content require significantly different processing for indexing purposes.

The advancements in video signal processing and machine learning also opened many possibilities for using video in other areas, ranging from security to self-driving vehicles. Many of these use-cases then have an additional requirement of providing a decision in a fraction of a second. Such systems, therefore, have to process the incoming signal online and as many times per second as possible. Furthermore, as human safety may depend on such systems, we would also prefer to have some insight into the reasoning of such systems.

## 1.1 Personal Motivation

I consider myself very lucky to be a part of the generation that grew alongside consumer audiovisual technology. My early childhood was captured in color photographs almost from the very beginning. The photographic film was a limited resource, usually allowing less than 36 images per roll. However, the processing of photographs was not exceedingly expensive in the early 1990s, compared to processing the 16 mm black and white film stock for capturing early home video (without sound, of course).

However, as the technology progressed, the audiovisual recording on magnetic tapes got much more accessible, first allowing to capture the analog sound and video directly on compact VHS tape in a limited resolution of 288 horizontal lines in PAL regions and substantially subsampled color information in two fields per frame, 25 frames per second. Later our home video recording changed to a competing Hi8 tape format with 400 horizontal lines and less aggressive color subsampling. However, as the Hi8 tapes were quite expensive, most of the footage was transferred to significantly cheaper VHS tapes anyway. As we transferred the content between two devices – one playing back the recorded footage and the second recording the signal onto the VHS tape – we gained the first direct possibility of linear editing. However, the transfer of the video material had an adverse effect on the quality of the footage and caused a loss of all metadata (unless burned in the video signal itself).

At the turn of the millennia, the digital video format started to dominate not only in the professional video market with a full standard definition resolution of  $720 \times 576$  visible pixels in PAL regions. The chroma resolution stored on MiniDV tapes was  $360 \times 288$  pixels (4:2:0 subsampling). As the recordings were still on a magnetic tape, they were cumbersome to navigate and had to be played back in their recording speed to transfer them anywhere. However, even the consumer-grade camcorders had a digital output allowing a direct connection to a computer and video content transfer without any additional loss in quality. However, the metadata loss (such as the date and time of the recording) still presented a significant issue for many archival and editing purposes.

After transferring the audiovisual material onto a hard drive, taking approximately 11 GB of storage per hour of recording, and splitting automatically into individual clips based on the timestamp recorded by the camcorder, the audiovisual material is much easier to navigate. Nonlinear video editing software then allowed easy trimming of the video material, combining the individual clips into sequences, overlaying with music or commentary, and much more. Consequently, our family started recording more and more, assuming that we could always edit it out but not record again. However, with the increasing volume of content, the editing was an increasingly laborious process

(multimedia index was practically nonexistent, metadata was scarce at best, and there was usually no agreement on what parts of the video should be kept and shared).

In the end, we usually archived both the raw material and sometimes the edited video, organized by year and event. The archive organization was not a significant issue at the time, as we still archived just our home video so far.

As I started my bachelor studies, I also joined the Audiovisual club of CTU students, where thanks to association CESNET, we had access to better equipment allowing high definition video recording (with a resolution of  $1440 \times 1080$  pixels and the same chroma subsampling 4:2:0) on the same miniDV tapes thanks to MPEG-2 compression. Later on, the club invested in new equipment that allowed recording on affordable memory media, which removed the irritating re-capturing of the footage into the computer, thus recording even more footage and preserving some of the primary metadata. Recently, the resolution of some of the footage increased to 4K ( $3840 \times 2160$  pixels).

In addition to the resolution increase, the significant change was that we commonly edited unscripted content recorded by someone else. Therefore, we sometimes had to rewatch the whole set of the recoded clips and reconstruct the story to start editing, as the footage was usually sorted into folders only by association to a project in our archive. However, significant exceptions in the archive structure were needed to allow different workflows, such as processing the lecture recordings that we started on the Faculty of information technology of the Czech technical university in Prague with Marián Šuch.

During my membership in the Audiovisual club, I also participated in a couple of amateur production feature films as both a cinematographer and editor. Even though these projects usually had some script prepared beforehand, the decision on individual camera shots was, in contrast to large productions, usually made during the shoot or editing. I also had a couple of short encounters with professional productions, either as an actor, digital imaging technician, or special effects assistant, where the individual artistic freedom is usually strictly limited and has to adhere to the script.

Recording scripted content usually involves a lot of hand-written notes that are very helpful in editing. However, even the best notes usually did not save me from watching the raw footage over and over again, multiple iterations of edits, discovering errors in the raw material, and trying to fix them in postproduction; even though I was usually the person preparing the shot breakdown, operating the camera, writing down the notes and editing the final product. Sharing all the information involved with anyone else creates significant overhead that is usually not worth it on small-scale production.

However, information sharing is crucial in projects that involve tens of people. Even though professional production usually has many people already present on the set – checking the compliance of the shot with their requirements. Therefore, they can usually decide which clips can be used in the final product already during the shoot, and the rest is kept only as a backup.

The earlier mentioned lecture recordings are much simpler to edit. Usually, we trim the ends of the recording and possibly take out some portions of the talk. However, the issue arises with the slides. The slides may be captured as a video with a separate device connected anywhere on the path from the computer to the projector, thus including the timing. However, we have to set the capture quality relatively high to preserve the quality of the image, as commonly used video codecs are not well suited for computer graphics; consequently, resulting files are enormous. The other approach involves obtaining the original keynote (as a set of images) and recording only the timing information. However, this approach requires either additional software on the speaker's computer that records the screens on keypress or mouse click; or tedious

manual synchronization of the slide images with the degraded slides captured from a projection screen in postproduction.

We also have two distinct approaches to sharing the slides that we have to decide during the editing of the talk. Either we keep it separate and present the two feeds side-by-side in a player (regardless of the slides being just individual pictures with timing or a full video feed), or we can combine the two video feeds into one picture-in-picture video. The side-by-side playback has a significant disadvantage of a limited number of platforms it may be shared on, whereas the picture-in-picture can be shared anywhere, but the semantics of individual slides and ability to navigate the lecture by slide gets lost.

I have been personally working with two platforms for presentation recording and sharing that produce a side-by-side video and navigable slides. The solution called MediaSite by Sonic Foundry uses an all-in-one capturing and streaming device that takes the video input sent to the projector. The software solution by a Czech company called SlidesLive can be used both on a separated device capturing the video feed or as an executable on the presenter's device, where it takes a snapshot of the presentation on specific keyboard and mouse events.

The recordings of the Week of Science (a two-week-long popularization science festival held by the Czech Academy of Sciences) captured by the MediaSite devices even inspired me for my master's thesis. The recordings had a relatively simple export format, and I had several years of the festival archived thanks to the long-term participation of the Audiovisual club on the festival.

The original goal of my master's thesis was to make the talks searchable and thus available for a broader audience. In the end, this goal was achieved by a speech-to-text conversion and optical character recognition, both using commercially available tools. Therefore, my master's thesis slightly shifted into the possibility of using all of the extracted features in a double fusion framework for coarse topic classification, with a possibility to predict the class of each slide (as some of the lectures were interdisciplinary) given the labels only on the level of whole lectures.

## 1.2 Multimedia archive and index

With the steadily increasing amount of diverse digital multimedia content, we faced a range of issues:

### Where to store the media files?

The full answer mainly depends on the state of the project.

The digital multimedia content is usually stored on memory cards and SSD drives in cameras and audio recorders during the shoot. The video signal can also be transferred to a director's monitor and recorded independently as a first backup. After a set of shots, the digital imaging technician (DIT) collects all media, creates several backups of the primary multimedia files, and sometimes even creates the first previews (with conversion to standard gamut and basic color correction) for the rest of the crew.

In the case of live-streamed events, the device responsible for the online editing and transcoding usually stores the edited version of the content on top of that (after transcoding, possibly with a different bitrate than the broadcasted one) and, if enabled, the live-streaming service stores copy of the broadcasted content as well.

Therefore, we may already have more than five copies of virtually the same content in the worst case. Only two or three are the binary equivalent copies of the original that is suitable for editing. The rest of the files may eventually capture the same image

or sound but are fundamentally different and offer lower flexibility in postproduction. However, they can be more beneficial for some parts of the creative process:

- The recording on the director's monitor can be used privately to check the shot for continuity.
- Proxy files can be easily transferred to VFX or other departments to start preparing the color grading and composition of the effects.
- The version converted to standard gamut can be provided for editing to speed up the rough cut.
- The online edit from the streaming machine can be directly used as the output of the event if no modifications in the cut are necessary.

Nevertheless, all of these files take up precious space. One of the most accessible and flexible storage solutions is still a hard drive. Magnetic tapes are cheaper per data unit, but only after a relatively high threshold, and navigating the tape is slow. And on top of that, none of these storage methods promise a long-term archival of the media. Therefore, long-term preservation of old film material is occasionally carried out by digital scanning of the original film strip, digital restoration of the acquired content, and, paradoxically, storing back the content by laser-printing the set of image fields back on a film strip.

In the reality of budget-limited productions, especially in documentary making, we are usually stuck with a limited number of offline hard drives, and, therefore, we usually need to decide which versions should be kept. If we decide to collaborate on multimedia material over the internet, we soon discover that the private online space is still relatively expensive, and we need to select an even smaller section of our archive to share. Although seemingly limitless, the public sharing space (as presented by services such as YouTube that allow the upload of up to thirty 12-hour long videos per day) also come with their flaws. The service is provided with no guarantees. Quality is significantly degraded. The content can be taken down for a long list of reasons, and the service requires the uploader to provide a worldwide, non-exclusive license which is at least troublesome for many authors.

### **Store the original media files or proxy?**

Once we finish editing the project and publish its results, we usually do not need the source material anymore. Or do we? What if we find a significant mistake after the release? What if we end up cutting away an interview piece that had no value for us now but may bring up a significantly different context for future viewers? What if the B-roll contains a special moment worth sharing in a behind-the-scenes video? What if the content will gain its historical value over time?

On the other hand, we may capture some visual signals in unreasonably high bitrate (resolution, framerate). We commonly used a high-end HDMI recorder for the slide capture to give the most glaring example. We had to use the best possible quality to preserve fine lines in the image during the capture (ProRes 422HQ), and as a result, the media files had approximately 100 GB per hour. Recompression of such video signal with static-image optimized codecs resulted in postproduction resulted in much smaller files with comparable quality. Alternatively, we used a method of slide detection and converted the video signal to a set of images and timing information, as we discuss in the thesis of Richard Burkoň [B3].

Digital media also have a significant tactile disadvantage over traditional film media, as the non-tangible digital files do not end up lying on the proverbial cutting-room floor as physical artifacts in a single copy. Instead, we tend to either hoard multimedia

content in several copies across multiple drives we cannot track, or we mindlessly delete the “unnecessary” digital files once the project is closed, neither of which is the ideal case.

Digital media archiving on its own is for sure a topic for another thesis. In our case, let us assume that we need to solve the problem of archiving most of the content (for possible future use) in a confined amount of space. One possible solution mimics the previously mentioned transfer of Hi8 tapes onto VHS—recode the content in lower bitrate with a possible loss in quality and troublesome metadata conservation. However, this approach requires massive IT support, extensive effort to track everything, and creates binary non-equivalent duplicates that are hard to discover.

### **How to handle duplicities on the file storage?**

Tackling exact binary duplicates (binary equivalent backups) is relatively easy, as we can transfer the problem of file duplicity search to a search of identical checksums, millions of which can fit into memory. Even if we take the improbable hash collisions into account, we significantly limit the number of file pairs to be checked for actual equivalency.

However, once we modify the media file in any way, possibly with just the change of included metadata, we lose such a simple method of duplicity discovery and have to resort to methods of media fingerprinting and introduction of an index that we have to construct as invariant to such editing operations.

One of the most prominent examples of such fingerprinting algorithms is YouTube Content ID; however, very little information about the internal implementation is publicly available. From the YouTube help portal, the music structure is considered in the fingerprint and, therefore, remixes and covers of a song may be flagged by Content ID as well. For video, there are only rumors on the use of neural networks pre-processing every frame of the reference video and measuring the distance of newly uploaded clips by the number of frames with matching low-dimensional representation.

However, a significant portion of this thesis will discuss the possibilities of feature extraction from a video signal and how we can use them for automatic index construction.

### **How to structure the archive?**

A simple folder structure on shared storage may suffice to minimize the duplication of the content and mixup of media files within a single media house. In the Audiovisual club, we historically used a system of consecutive IDs managed by a central project repository. This system was far from ideal, as we had no provisioning for recurring events, and it would require us to use a structure-less ID to mark all assets. Usually, we provided the id only to exports and kept the source material in poorly structured auxiliary folders. Therefore, the heavily compressed output files were relatively easy to find and manage, but the manageability of the rest differed significantly from project to project.

In 2009 the Audiovisual club ditched the approach of sequential IDs and started to identify the project folders with a date and short name of the project. This approach required no interaction with the central project repository and allowed a quick search for event-based media directly on the storage without knowing the ID. Moreover, this simplification promoted better archival practices as recurring events were no longer hoarded in a single project folder but remained searchable by a keyword in the folder name. However, tracking the project’s progress was significantly more challenging as we lost the direct association with the project repository.

IDs are a more manageable approach when multimedia project management relies on several people. However, it does make sense to provide some structure to the identifier. For example, Czech television, the national broadcaster, uses a numerical ID that contains the year of manufacture, department number, sequential number of the program, and a part number for recurring programs. All assets then use this identifier in central multimedia storage. On the other hand, this approach practically requires a single ingestion point and specialized department to manage the assets from their preparation for editing to broadcast and archive.

The indexing approaches mentioned above are then most suitable for time-limited events or closed projects but do not offer the flexibility needed in collaborative projects that rely on sources from many authors or projects spanning several years and productions. In such cases, we are more interested in the content of the multimedia file itself than the date and event it was captured on.

### **How to allow re-use and collaboration?**

In the traditional setup, the media re-use is commonly facilitated by the archival department that keeps the essential information on all material in their hand-curated index. Such index allows retrieving the audiovisual material that belongs to a given time range, event, program, including specific words, people, places, objects, actions, and possibly much more. With a narrowed-down selection of clips, the archival department can rewatch the content and pick only the parts of interest.

Collaboration between organizations (or even of separate departments of the same company) in this traditional setup then involves extensive communication with each archive, which is usually very time-consuming.

In the ideal case and with unlimited internet capabilities, the index can be theoretically provided to other parties, at least for the primary search. However, such an index would have to be entirely understandable to the searcher, use the same terminology, and most importantly, contain the type of information that the collaborator expects. Therefore, in reality, the archive search is, to this day, usually a laborious process conducted by people who retain some primary knowledge about the material in the particular archive they manage. In the case of public organizations, only a limited set of the stored information may be provided to the public – such as the name of the program and short annotation – to provide some visibility, but the more complex search is usually not available. One of the very few examples of public projects that provides search in the index of text spoken in movies is PlayPhrase.me which seems to be just a fun project of Eugene Potapenko.

We advocate that the utilization of the internet in multimedia archival does not even have to end with the simple sharing of the index, but that should allow collaboration on top of the metadata, annotations, and the multimedia material itself. One of the best examples of a long-term public multimedia archive is project Pad.ma (Public Access Digital Media Archive) that contains densely annotated documentary video material (both edited and as individual clips), and project 0xDB.org, which contains an extensive collection of edited movies.

Both of these projects use an open-source software pan.do/ra, which allows ingestion of local network or internet resources, dense manual annotation of the material with timestamped keywords and description, extensible metadata, and includes even an editor that allows creating sequences of clips directly in the interface of the web application.

However, all of the annotations have to be added manually to pan.do/ra, search is enabled only with text on text annotations and the sequences, no sense of clip similarity

is present in the system, and sequences have to be manually created and stay within the realm of linear narratives. The team around Eric Rosenzveig tried to tackle these issues in NARRA.

### 1.3 NARRA – Platform for Open Narratives

The screenshot shows the NARRA web interface. At the top, there are navigation links: NARRA, Projects, Libraries, Visualizations, and a '+ Add New +' button. A user profile 'Petr Pulc' is visible in the top right. Below the navigation, the breadcrumb path is 'ŠUMAVA / ŠUMAVA BLOCKADE 2008-11 / ANICKA\_SABATOVA'. The main content area is divided into two tabs: 'Metadata' (selected) and 'Player'. The 'Metadata' tab displays two sections: 'USER' and 'SOURCE'. The 'USER' section contains a table with fields like Author, Description, Keywords, Language, Location, Organizations, People, and Shot Type, each with a value and a timestamp. The 'SOURCE' section contains a table with fields like Audio Channels, Audio Codec, Audio Sample Rate, and Bitrate, each with a value and a timestamp. To the right of the 'USER' table, there is a video player showing a scene in a forest with a red and white caution tape. The video player has a '00:00' timestamp and a play button.

USER			
Author	Štěpánka Rocková		
Description	Anna Šabatová se policistů ptá na zákonnost kácení v Národním parku		
Keywords	Ombudsmanka Anna Šabatová, Policie ČR, blokáda, paseka, nelegální kácení, les		
Language	cs		
Location	48.986187857951236, 13.498377799987793		
Organizations	Policie ČR, Hnutí Duha, Ombudsmanka	2015-07-29 16:10:43	Eric Rosenzveig
People	Ombudsmanka Anna Šabatová	2015-01-29 18:10:20	Eric Rosenzveig
Shot Type	Medium Shot, Pan, Handheld Shot	2015-10-29 18:10:11	Eric Rosenzveig

SOURCE			
Audio Channels	2	2015-07-29 17:10:04	
Audio Codec	pcm_s16le (sowt / 0x74776F73)	2015-07-29 17:10:04	
Audio Sample Rate	48000	2015-07-29 17:10:04	
Bitrate	5155	2015-07-29 17:10:04	

**Figure 1.1.** Example of an item in the NARRA collaborative environment with associated metadata from different origins: User metadata are filled manually by an annotator, source metadata are taken directly from the ingested file.

At the start of my Ph.D. studies, I joined a project NARRA of Centre of Audiovisual Studies of the Film and TV School of Academy of Performing Arts in Prague as a part of proposed broader collaboration with my home faculty – the Faculty of information technology at the Czech technical university. The project’s main goal was to provide a collaborative platform for creating open narratives – arrangements of multimedia content that allowed the viewer to freely explore the corpora of text and multimedia documents in a branching manner and at their own pace.

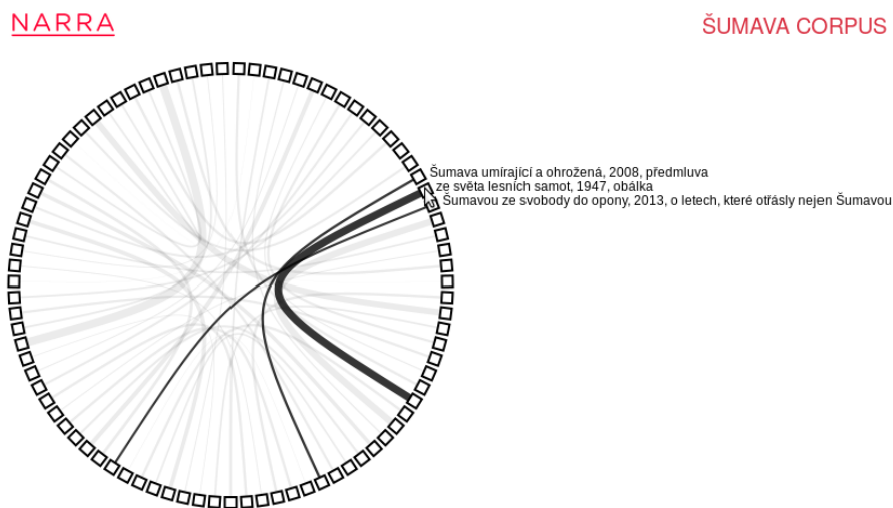
Similarly to projects such as the already mentioned pan.do/ra, NARRA was also envisioned as relatively thin metadata storage on top of the multimedia content. However, in contrast to such projects, the main goal of NARRA was not the annotation of the multimedia content itself but rather the creation of specific visual exploratory tools for multimedia projects. As such, NARRA can be used both internally during the editing process to discover related content and externally as a tool to present the set of branching narratives, allowing the spectator to choose where they want to continue with the narrative, based on a set of storylines.

For public displays, NARRA still relied on editors who authored several pathways through the available content to guide the viewer through the material. This decision was deliberate, as we want to preserve the narrative function of the multimedia project, representing the editor’s intents, rather than overloading the viewer with a large amount of automatically linked contextual information. However, the system could then use various data visualizations to represent the navigation possibilities, such as a (possibly directed) graph with different edge types connecting the corpus items.



As a somewhat ambitious goal, NARRA envisioned learning on the human-made annotations and associations to offer an algorithm for automatic construction of synthetic connections in the graph based on some properties of the content itself. Thus automatically proposing new possible narratives or other contextual connections. As a side-effect, the semantic properties of the content should be public and searchable to simplify work for the editors of the curated narratives.

However, even though such a vision looks simple on paper, the implementation of such an idea had several technical obstacles. We wanted to show all of the extracted features that participate in the association in a graphical user interface which implies that all of the extracted features have to carry high-level semantics that is easy to represent. However, such features are hard to obtain from a mixed dataset without reliable ground truth that we could use for training. On the other hand, low-level features are easy to extract but usually retain high dimensionality and cannot be easily transferred into the semantic features without extensive machine learning on much larger datasets than we had on our hands.



**Figure 1.2.** Visualization of text document similarity on the Šumava corpus [1].

A couple of diverse projects were selected to showcase the capabilities of NARRA, with media ranging from social media updates and book excerpts to recordings of a television broadcast and raw video footage. Therefore, extracting knowledge from such diverse items also requires different processing methods. For example, the news segment of the television broadcast usually has graphical overlays specific for the television channel that usually contain a summary of the topic discussed, possibly including the event’s location. On the other hand, the raw footage has no graphical overlays but may have the location information stored in its metadata as GPS coordinates.

On top of that, the creative process may create additional complications for indexing, as some of the content may have multiple versions, such as raw footage and several modified variants created by different editors (for example, with different color grading, cropping, stabilization, overlays). However, with no pre-defined naming convention or global identifier persisted in the file metadata, there is no simple way to automatically discover such variants or decide which version should be considered final. In addition to the individual clips, the archive can also contain the edited content created by joining the individual clips already stored in the archive. The association of individual raw clips and the final cut is stored in the editor project files during the editing. However, once the

edited version is rendered out, such association is lost. For such cases, NARRA allows connecting the individual clips from the corpora to the edited version by a different type of connection containing timestamps of the edits.

NARRA platform can import an edit decision list (EDL file) and make the connections automatically from raw footage to the edited result based on the filenames (which is by itself far from ideal, as the file names may be ambiguous). However, for most of the edited content, we had no access to such a file. Therefore, similarly to the other features, most annotations had to be added tediously by hand.

On top of the laboriousness, the annotations provided by a human have one significant pitfall – hard-to-maintain consistency. Even in the movie production, where most of the definitions seem to be well established (at least for an external observer), and the same understanding of such definitions is crucial for communication within the crew, the actual meaning of even the basic definitions such as shot sizes and angles may differ from one team to another. This ambiguity poses a significant issue for open narrative systems as they inherently assume independent work of multiple artists on the same data. Even though NARRA proposed limitations to some feature values, we finally allowed free text values to allow easy modifications for any project in the future.

For a more thorough reading on open narratives and the NARRA motivation, please refer to [2], as the rest of this thesis will focus mainly on the aspect of knowledge extraction.

## 1.4 From a Book to a Movie (and back)

To illustrate the variety of information involved in creating multimedia content (and what we may extract from it with some reverse operation), let me provide a simplified guide to transferring a written story to a finished movie in Fig. 1.3. Although such a detailed process reflects the creation of feature films, while smaller or live productions tend to skip some of the steps, we can theoretically take any video content and transfer it by hand to a structured text document that resembles a movie script afterward.

### Story

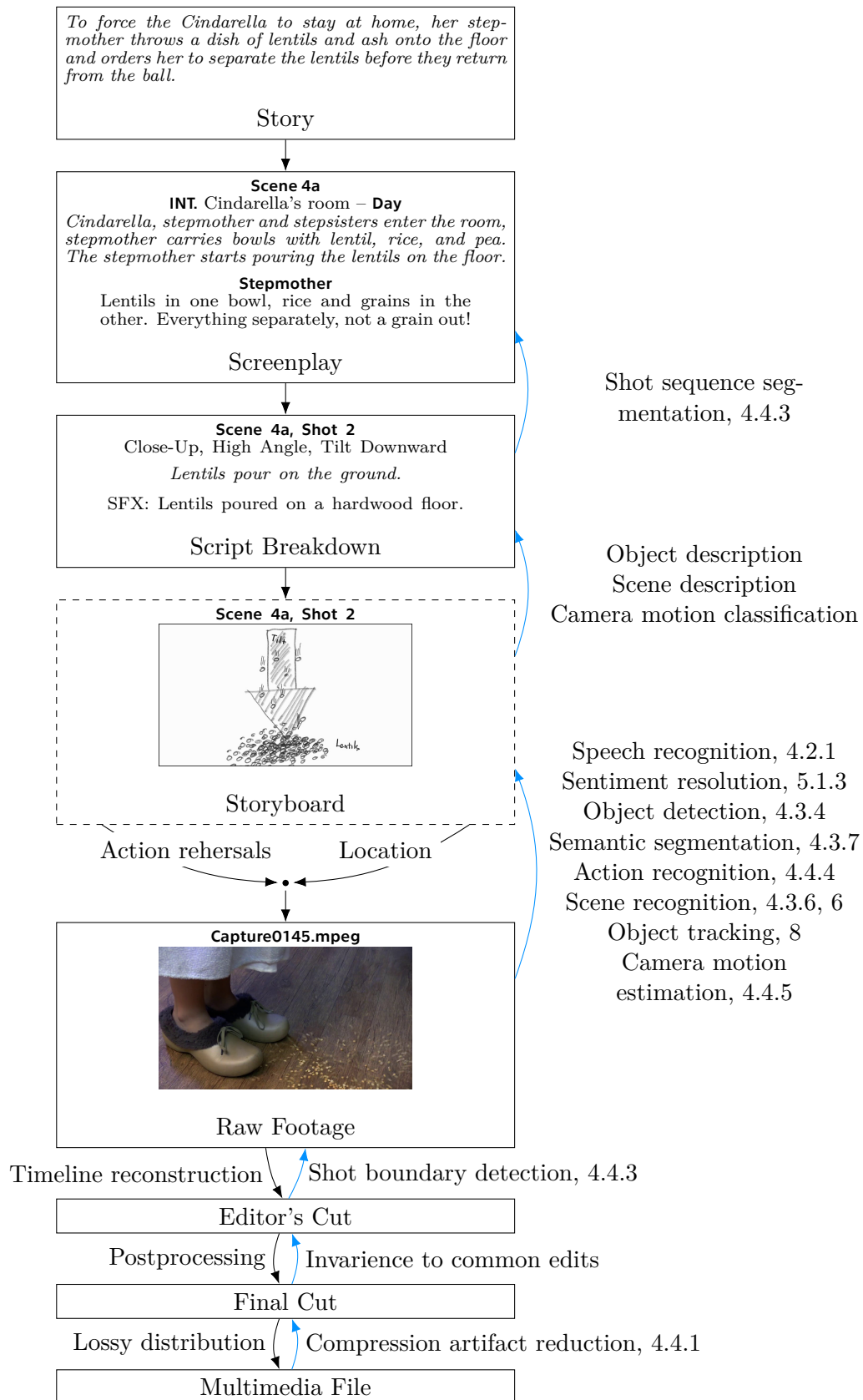
Many movies start with a written story. Its free text is usually loosely structured into chapters but mixes descriptions of the environment, objects, and characters with dialogues and actions.

Understanding the story and comparing it to the others allows us to, for example, recommend new stories or classify a genre. Moreover, in the context of knowledge extraction from multimedia, the transformation of the edited video back to free text is sometimes viewed as its ultimate goal. However, not only in NARRA, structured knowledge extraction is preferred as it allows more complex search methods in the given corpora.

### Screenplay

The original story is divided into scenes to allow more straightforward navigation in the text. Description of the environment is extracted and corrected against other parts of the story taking place in the same setting. The screenplay author may add some detail or change some aspects of the story; however, the description is still pretty much given as a free text. Only the most essential aspects of the environment, such as a decision on Day/Night and Interior/Exterior, are captured in designated screenplay fields.

The screenplay writer also transfers the free text into a set of explicit actions and dialogues. The automatic processing of such information still relies heavily on natural



**Figure 1.3.** Simplified process of creating a feature film (arrows from top to bottom) and some of the approaches to reverse the process and possibly extract semantic information with references to the relevant sections of this thesis (bottom to top).

language processing. However, the additional structure helps to distinguish individual elements of the text and process them differently. For example, we may correlate the individual dialogue lines with the direct speech segments of the original story.

### **Script Breakdown**

The scene is then further divided into individual shots. In addition to the description of the environment (which is usually given only once for the whole scene), we add explicit information on what should be captured by the camera in a given shot, such as: what is the timing of individual actions, list of used props, positions of individual actors in the scene, the position of the camera and its motion during the shot, size and angle of the shot, depth of field and other technical information.

Therefore, the script breakdown contains valuable structured information correlating with the semantic features we would like to store in NARRA. Based on such annotations, we may provide a possibility of discovery of certain content, search for illustration shots, and much more.

### **Storyboard**

To further prepare for the shooting and possibly provide some visual material for the special effects team, the script can be transformed into a set of sketches that provides more information on the actual composition, visual appearance, and motion of the actors, objects, and camera, while restricted to a selected aspect ratio. The detail captured in the sketches can vary significantly from simple stick figures to artistic drawings; some are just hand-drawn, some are constructed as collages or inpainted to photos of the proposed location.

The primary purpose of the storyboard is to capture some information on the visual layout of the shot with only high-level semantics and as few frames as possible to capture the action sufficiently. The last point partially correlates with one of our goals in knowledge extraction, as we consider the complete processing of tens of frames per second in sections with no fast-paced action as generally superfluous.

### **Raw Footage**

The next step is to transfer all the prepared text and image material to a set of video clips. For feature films, this usually involves many people of different professions: the locations team has to find an appropriate place where the shooting will take place; casting department has to find suitable actors; set department, construction, and lighting unit prepare the location to match the visual appearance to the visions of the authors; costume department may either modify or create bespoke clothing for the actors; grip prepares the structures for mounting and moving the camera around the scene; actors rehearse their part; camera operator starts recording; first assistant camera checks the focus of the camera; the second assistant camera shows a clapperboard to all rolling cameras; the sound unit prepares their microphones on long boom poles. Clap! Action!

A script supervisor checks if the actions conform to the script and if the shot preserves continuity. After each take, they write down notes into a shot log which allows a direct association of a piece of film material or a digital file to a scene in the script. As mistakes happen and some of the takes may be decided right during the shooting as unusable, the shot log significantly assists during the first rounds of editing. Some directors may even request two or more good takes in a row to be extra sure (and the digital recording allows them to do retakes relatively cheaply).

### **Editor's Cut**

Once the raw footage gets to the editor, the individual good takes are sorted according to the script, trimmed, and joined together. As a result, the sequence of the individual

shots reveals once again the whole story. Even though the editor's cut (sometimes called a rough cut) usually exceeds the designed runtime and is overall not pleasing to watch, this version may be used for the recording of the voice-overs, as well as sound design or visual effects preparations.

This stage is usually the last one that can be captured with a simple edit decision list, as it usually uses no intricate transitions, special effects, or overlays.

### **Sound Effects, Music and Speech**

Once the scene's mood is established in the script, appropriate sound effects and music can be selected from existing sound banks or recorded to match the edited video. The creation of new sound effects usually does not involve any extensive preparation and is up to the foley artists to come up with a combination of objects that will produce an effect that goes well with the already captured visual action.

On the other hand, music recording is usually based on a score or a sheet divided into individual instruments. Keeping an association from a music piece to its score would be very helpful for easy extraction of its structure, main motive, assessing similarities, or enabling easy search by example. However, such association is usually not retained, and just the mixed-down result is usually available. Furthermore, multiple instruments usually mix in each of the limited set of audio channels as if they represent the virtual placement of the instrument in the scene. Therefore, leaving no straightforward way how to separate the individual instruments from the recording.

On top of that, the narrative voice-overs and dialogues are mixed in, usually re-recorded in a process called revoicing or overdubbing. Similar to the instrument separation, separating the spoken word from the final downmix would be beneficial for automated speech recognition. However, in reality, clean separation is complicated.

### **Final Cut**

By making further edits to the editor's cut, mainly by cutting out sections that disrupted the flow of the story, color correcting each clip, and combining the footage with pre-mixed audio and rendered visual effects and graphics, the final cut is prepared for distribution. Most of the supporting or unused material gets discarded as the production starts working on another project.

In the end, with the feature films, we are left only with some copy of the final cut, usually heavily compressed or degraded by copying the material multiple times along the way, the free-text description of the distributor, possibly the film genre and the Motion Picture Association film rating that denotes suitability of the content for limited audiences.

With the digital multimedia shared on online platforms, we might be in an even worse situation. The platform usually offers a way to store the title and short description of the video. However, they may not entirely reflect the content of the video, possibly containing a misleading title, oversimplified annotation, unrelated text (such as links to social platforms of the author), or even nothing else than the name of the original file.

## **1.5 Goal of our Research**

Editing scripted multimedia content may seem like an easy task from the introduction above. However, even if we have a perfectly prepared script and there is no place left for improvisation in any way, all of the steps rely on correct annotations that allow exact matching of individual takes in the footage to the original script. If we violate these

assumptions in any way, editors may need to get manually through all the acquired material repeatedly, significantly slowing down the whole process. To mitigate such issues and allow collaboration on the annotations, the film *Slum Bombay* uses external deployment of tool `pan.do/ra` on the website `Pad.ma` to publish shots from the movie, and the NARRA tool was deployed experimentally in Czech VFX company R.U.R.

For documentaries and unscripted content, the situation is significantly more pronounced. For example, documentary authors have varying levels of access to an overwhelming amount of material they need to sieve through for creating a compilation sharing their view on the story with the viewer—material for which they may have just an internet link and no other provided annotation. On the other hand, unscripted content authors may have sparsely annotated material in their archives. Consequently, they would like to leverage the footage limitations (recorded in a limited number of places, with a limited set of people) to automatically provide annotations to the rest of the archive, similarly to photo management applications.

These scenarios share one common theme—they would significantly benefit from automatic annotations that would simplify search and discovery in the corpora. Therefore, the ultimate goal of our research was to enable the extraction of structured semantic information from a limited archive of unannotated multimedia content in an efficient manner.

To this end, the original vision was to use existing feature extraction algorithms to pre-process multimedia content. On top of such features, we wanted to provide a simplistic layer of classification algorithms that would provide the semantic annotation in the context of a small and poorly annotated multimedia archive. Multimedia content processing is currently a dynamic research field, with many novel neural network architectures introduced during the last couple of years. Many of these approaches can provide semantic annotation directly, even if on a deliberately limited set of media. Simultaneously, the computational power necessary to train such complex classifiers got more available, including specialized hardware for tensor computations.

On the other hand, the scalability of such an approach is limited in real-world applications, as long training on increasingly large datasets is required to provide satisfactory results. The overutilization of neural networks also harmed the explainability of the proposed systems and introduced new phenomena, such as shortcut learning.

Therefore, we slightly shifted our research focus to the better utilization of computation resources and explainability of the underlying computation. At the same time, we still experimented with some of the neural network architectures, but on a much smaller scale to imitate real-world scenarios with smaller private multimedia archives.

## 1.6 Contributions

As such, the contributions of our research are in four interconnected areas.

First, we directly collaborated with the Film and TV School of Academy of Performing Arts in Prague on project NARRA, partially by providing functionality to the open narrative system itself [B7, B6] and partially by researching approaches to the utilization of image processing and machine learning in the open narrative systems [A11].

Consequently, we wanted to speed up the knowledge extraction proposed for the collaborative systems. To this end, we wanted to exploit the repetition of information in the multimedia content across the individual modalities and in the temporal domain. Therefore, we started with a theoretical framework for utilizing meta-learning to extract

features efficiently [A10]. However, this framework inherently required a set of methods to extract easy-to-obtain knowledge from the multimedia file that we can consequently use to recommend more sophisticated (and expensive) knowledge extraction methods, and none of the existing methods seemed to be a good match out of the box.

Therefore, while exploring methods suitable for knowledge extraction from multimedia data, we experimented with several extraction methods ourselves.

As a minor side-project, we experimented with MPEG-7 audio descriptors to discover sentiment from utterances [A8] and recommend dancing style [B5] as a means of extracting mood from the multimedia content.

After that, we focused mainly on two areas that provide essential information about the scene and objects in the foreground with relatively low computational complexity in video signal processing. One branch was dedicated to scene detection from low-level statistical features [A6] and keyframes pre-processed by standard neural backbones [A1]. In the second branch, we experimented with knowledge extraction based on tracking of visual features [A9] utilized for foreground segmentation [A11, A7, A5] and visual tracking in general [A4].

We then significantly extend our research in visual feature tracking to tackle the problem of multiple object tracking. To this end, we propose an approach that satisfies our search for efficient knowledge extraction methods and is simultaneously directly comparable to existing methods on available datasets (opposed to the optimization problem of general knowledge extraction). Moreover, common approaches to multiple object tracking are mainly concerned with the extraction and prediction of the motion itself (which presents an essential feature for robotics, autonomous driving, and many other problems). Opposed to that, we showcase that image processing can be significantly less computationally demanding if we utilize a method of low-level optical feature tracking for the general tracking and only resolve some of the edge cases with the entire stack of neural-network-based object detection, description, and matching. Simultaneously, we are interested in reducing the costly object descriptor extraction as they may be re-used during the tracking of the object or updated only from time to time. As a side-product of our research in this area, we also present a novel unsupervised approach that can extract a dataset-specific training set for state-of-the-art object tracking approaches [A2].

## 1.7 Structure of the Thesis

In order to introduce the individual modalities involved in the multimedia content, the next chapter will summarize the properties of the audio and video modalities and their acquisition process. In the last section, we will also mention the use of metadata and its storage formats.

Chapter 3 will provide a common theoretical background for both the state-of-the-art knowledge extraction methods described in Chapter 4 and our approaches that will be described in the following chapters.

Chapter 5 will provide a theoretical framework for efficient knowledge extraction from multimodal data. We will provide a summary of the fundamental methods of combining classifier outputs, including the fusion of the final class labels and the combination of input data with similar semantics. Later in that chapter, we provide an overview of meta-learning and our theoretical take on an efficient iterative multimodal data classification framework based on the fundamentals of meta-learning.





# Chapter 2

## Multimedia Acquisition

As our research wanders on edge between theoretical informatics and art – and some technical properties of acquisition devices are closely related to the challenges we face – we briefly introduce digital audio and video capture and storage methods.

### 2.1 Audio

We can produce the audio signal in two fundamentally different ways – either by capturing sound with the help of various microphones or by taking the electrical signal directly from musical instruments and other devices.

While the direct capture of an electrical signal has to be checked just for the signal levels not to exceed the input rating of the capturing device (which would result in a signal distortion), nor be too low to hide in the noise, the microphone selection depends on the individual use case and may result in very different results. By physical means of operation, we commonly use three basic types of microphones:

- Dynamic microphones are the most robust, least sensitive, with narrower frequency response, but do not require external power to operate. The coil inside creates an electric current when a sound wave hits a connected membrane.
- Condenser microphones are more sensitive, commonly with designed directional characteristics, and require phantom power of usually +48V. The membrane acts as a second side of a charged condenser, changing the voltage.
- Contact microphones do not capture pressure waves in the air but rather the vibration of a surface on which we attach the microphone. These microphones usually utilize a piezoelectric effect, generating voltage on the deformation of the piezo crystal.

Based on the situation and the properties of captured sound, we can carefully select different microphone sizes and mounting options. For example, we will probably use a hand-held dynamic microphone for a singer on a music stage due to its lower sensitivity. For recording dialogues in a relatively quiet environment, we may use a lavalier attached to ones' clothes. In soundproofed studios, we will take the larger membrane microphones to capture a broader range of frequencies.

The position and direction of the microphone also play a significant role in the properties of the captured sound. For example, acoustic instruments tend to emit a specific range of frequencies or timbers to different directions from the instrument. Human speech captured near a throat mainly contains the lower frequencies, rendering the resulting sound illegible. Positioning the microphone directly in the trajectory of the exhaled air then makes the plosives and some fricative consonants (the [f], [p], [b], [Φ], [t], or [k] sounds) more pronounced or may lead to mechanical or electrical clipping, resulting in distortions.

Moreover, fricatives such as [s] and [ʃ] are close to different colored noises based on the position of the tongue (in these two cases, blue and white noise) and may cause internal resonance in the microphone, falsely amplifying such sounds.

Apart from the frequency properties of the microphone, we have to be aware that the signal from the microphone is of very low intensity, and we need to amplify it even before conversion to a digital signal. The amplification should not introduce a change in frequency spectrum or distort the sound; however, it will create a noise floor. The more power we provide for amplification and the worse the amplifier components are, the more pronounced the noise in the acquired signal will be.

Last but not least, we rarely record the sound in a completely isolated environment like a vibration-isolated anechoic chamber. Therefore, we must be aware that we record a mix of multiple sounds and that background noise or sound reflections are present. Therefore, to eliminate these issues, we tend to position the microphones as close as possible to the sound source with appropriate directional characteristics and wind protection.

### ■ 2.1.1 Digitisation

To be able to store the audio signal digitally, the signal has to be sampled and quantized. The most commonly used sampling frequencies are 44.1 kHz and 48 kHz – values selected with human hearing in mind, as we expect a healthy young person to fully perceive frequencies of up to 24kHz, although this threshold lowers significantly with age. To encode the highest possible frequency, we have to set the sampling frequency to double that according to the Nyquist–Shannon sampling theorem [3].

We may consider the somewhat odd sampling frequency of 44.1 kHz as a relict of the past when professional productions reused video equipment to record pulse-code modulated (PCM) audio and simultaneously keep the standard universal across PAL and NTSC regions (with 50 or respectively 60 fields per second). This method of recording audio onto a videotape was rendered obsolete soon after its introduction; however, the PCM encoding with 44.1 kHz sampling rate remained up until these days, especially in music industry.

For quantization, we usually use 16 or 24-bit samples. Higher precision is usually unnecessary, as the standard analog to digital converter itself cannot produce a significantly better signal-to-noise ratio.

### ■ 2.1.2 Lossy Compression

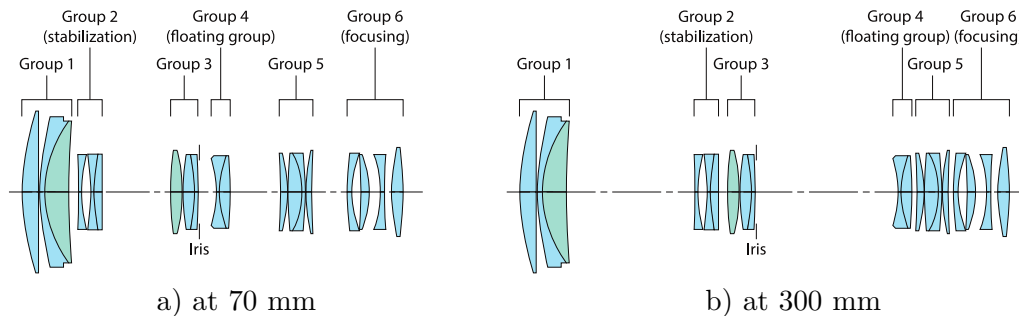
Due to the relatively low bitrate of PCM audio files (48kHz 24bit sound requires just 144 kB per second per channel), we can easily store the audio signal as uncompressed.

However, some lossy compression algorithm is commonly used for personal use, storage on mobile devices and transfer over cellular networks, as the compression algorithms allow a significant reduction in bitrate with a minimal loss in quality. For example, the MPEG-1 Audio Layer III encoding (better known just as MP3) uses psychoacoustic modeling on top of Fourier Transform and Modified Discrete Cosine Transform to remove parts of the signal masked by our perception in temporal or frequency domain, and therefore unnecessary in the resulting files. Moreover, the audio file may contain multiple very similar channels; thus, we may store only their differences in the encoded file. Also, only a narrow band of frequencies is required for some applications, such as telecommunication, to keep the human speech legible, whereas we may safely discard the rest of the spectrum.

## 2.2 Video

The acquisition of a digital video signal is significantly more complicated than the audio recording. Instead of converting pressure waves into electrical signals, we need to collimate light rays bouncing from the environment onto an extensive array of miniature light sensors and heavily process the data to enable its storage on non-volatile media.

### 2.2.1 Lens



**Figure 2.1.** Cross-section view of a 70 – 300 mm  $f/4 - 5.6$  optics.

The selection of a lens and lens setup significantly affects the properties of the captured image. Typically, the lens is composed of multiple segments, which can be moved around to change what distance is in focus. So-called “zoom” lenses, an example of such is shown in Figure 2.1, allow to change the focal length as well. The change in focal length has several effects on the final image. First, the angle of view decreases with the increased focal length. As a side effect, the objects captured with a lens having a long focal length appear larger (as they now cover a larger sensor area), and a visual separation of the foreground and background gets less pronounced. Also, the reduced field of view reduces the amount of light available for capture, resulting in a shallower depth of field and a requirement for higher sensor sensitivity.

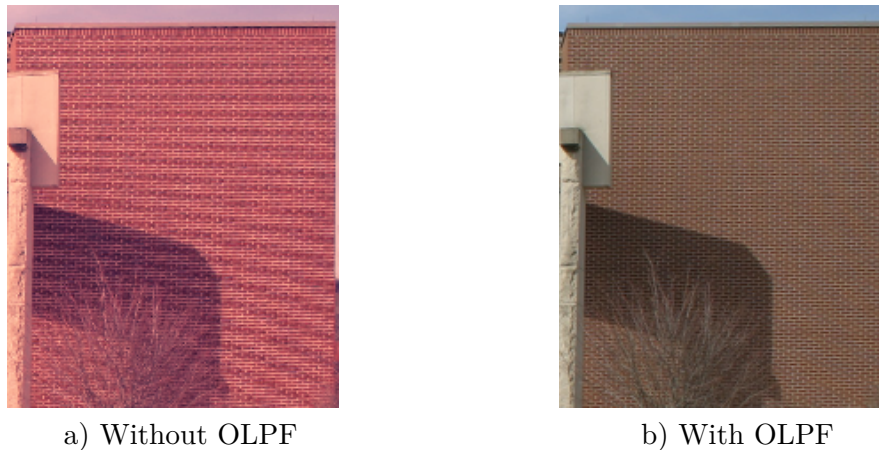
Extremely short focal lengths (of so-called fisheye lenses), on the other hand, have such a large field of view that they distort the image quite significantly.

In between these optical elements is an iris, which enables an additional control of the amount of light getting through the lens. Closing the iris then has a side effect of increased depth of field (distance between two objects in acceptable focus in front of the camera). The shape of the iris then changes a shape of bokeh – projection of out-of-focus light sources and reflections.

Usually, the lens aims to project the image as truthfully as possible on a straight optical axis, without significant distortions and aberrations. However, due to physical limitations, the final amount of light is usually not equal on all sensor cells, and so-called vignetting occurs when the image’s borders seem darker than the center, even on a constantly illuminated scene.

### 2.2.2 Filtering

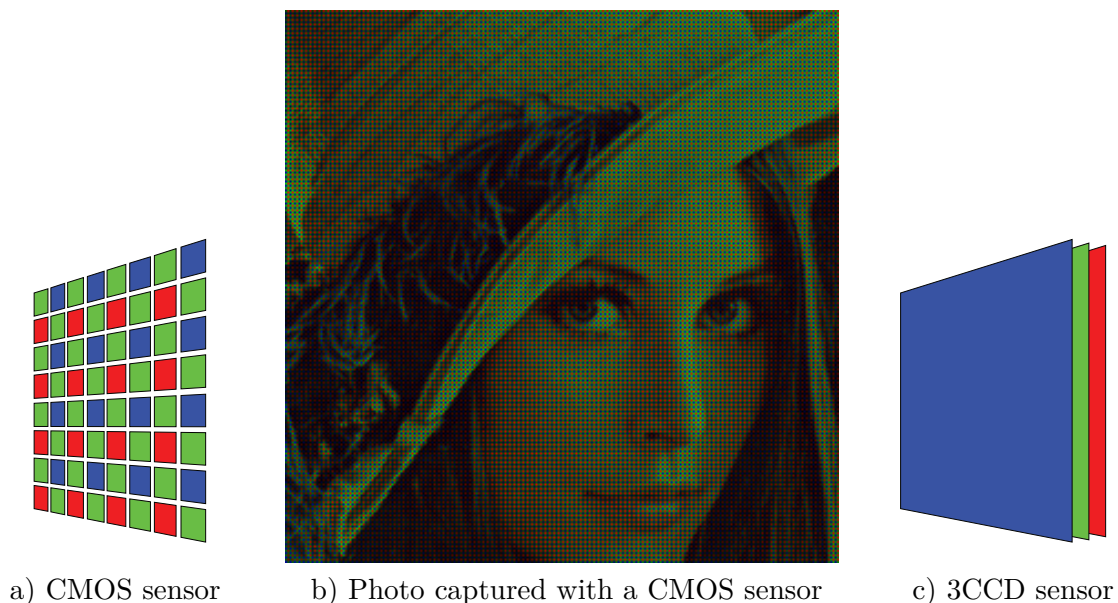
Until now, we just concentrated the beams of light on a plane of the chip, which means that unwanted frequencies in both the light spectrum and spatial distribution are still present (see Figure 2.2a). The first introduces significant disproportions in color balance, as some of the near-infrared light can be captured by the red cells of the sensor. The second causes the moire effect due to the spatial sampling on a sensor.



**Figure 2.2.** Effect of optical low-pass filter (OLPF).

Both issues are commonly removed by an optical low pass filter (OLPF) that blocks off the invisible parts of the spectra and slightly blurs the image to prevent the capture of higher spatial frequencies than the resolution of the chip allows.

### 2.2.3 Color Sensing



**Figure 2.3.** Different options for colour sensing.

As individual cells are sensitive to any wavelength and we would like to capture color images, we need to filter the incoming light further. To this end, the currently most popular CMOS (Complementary Metal–Oxide–Semiconductor) sensors use one of three color filters in front of the individual cells – red, green, and blue – arranged to a repeating pattern. The most commonly used is the Bayer pattern (illustrated in Figure 2.3a), which exploits our higher sensitivity to brightness changes in the green part of the spectra. However, as the resulting image resembles the example in Figure 2.3b, the internal processing of the camera has to interpolate the color information to determine the full RGB color of each image pixel.

Aside from this system, historically, three separate CCD (Charge-coupled device) sensors have been used to capture their independent piece of the color spectrum, as we

illustrate in Figure 2.3c. This approach required a selectively reflecting mirror system to route the filtered light in the camera, but on the other hand, this approach increased the maximum resolution achievable with CCD chips that suffered from overheating with increasing resolution.

#### ■ 2.2.4 Shutter and Gain

The readout of the data from the sensor may be of two basic types. The CCD technology uses a so-called “global shutter,” where all pixel information is pulled to a buffer simultaneously, mimicking film cameras’ mechanical shutter.

The currently more prevalent CMOS chips use a different approach, where the individual subpixel values are read one by one and pushed to a buffer as serial data, resulting in a “rolling shutter.” However, based on the readout speed, each pixel value is taken at a slightly different moment in time. As a result, low-end CMOS chips with a slow readout may produce distorted images if the camera or the captured objects move quickly. This rolling shutter effect is much less pronounced in newer high-end CMOS chips, as they can combine several fast readouts into a single frame producing results similar to the “global shutter.”

Another property of a sensor that can be changed is its sensitivity. In both film and digital photography and professional cinematography, the sensitivity is usually denoted by an ANSI ISO number. Since we are mainly concerned about digital cinematography, the amplifier gain measured in decibels seems to be more appropriate. If we use the amplifier in a way recommended by the manufacturer, the amplification gain is equal to 0 dB. Increasing gain to +6 dB doubles the voltage we are providing to the amplifiers in the sensor in return for its higher sensitivity. However, providing more power results in higher heat dissipation and creates more noise in the picture.

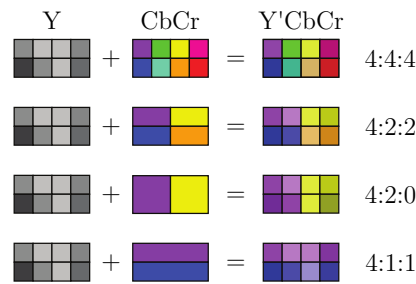
#### ■ 2.2.5 Lossy Compression

The amount of data acquired from the sensor is enormous. Current top-level camcorders, such as the RED Monstro 8K, can process continuously up to 75 frames per second of 8K full-frame ( $8192 \times 4320$  px) image with 12 bits per color channel. That totals to approximately 12 GB of raw data per second. This amount of data must be subsampled and compressed in real-time to allow storage on solid-state drives with a maximum data rate of circa 300 MB/s.

Similar to audio compression, we can first reduce the amount of information encoded because of some limits of human perception. As we are significantly more sensitive to changes in luminance than the color change, we can convert the image to a color model that separates luminance and the color differences components and subsample both of the color difference channels.

In practical applications, we use a color model denoted as Y’CbCr, therefore containing the luminance channel and color differences in blue and red channels and subsampling expressed by a ratio  $J:a:b$ , where for a block of  $J$  pixels wide and two pixels high  $a/J$  denotes the horizontal subsampling and  $b$  denotes the number of changes introduced by the second line of the block. See Figure 2.4 for a list of common chroma subsampling ratios.

In addition to the chroma subsampling, the video compression algorithms can leverage the fact that changes from one frame to the other in a sequence with a relatively high framerate can be minimal. In many cases, the changes from frame to frame can even be modeled by storing the visual information on a block of pixels just once (in a keyframe) and providing only the stream of transformations of such block in a couple of



**Figure 2.4.** Common chroma subsampling ratios.

subsequent frames. However, many professional video productions do not utilize such compression as it introduces motion artifacts which may pose an issue for special effects or other subsequent processing.

For compression of the individual frames (or, respectively, just the keyframes), the luminance and chrominance channels are processed separately by dividing the image into smaller blocks, applying some transformation method to a frequency domain, quantizing the coefficients, and storing only the first couple of them. The most commonly used algorithms are based on two-dimensional discrete cosine transform, similar to JPEG still image compression. Alternatively, the compression can be based on some other wavelet transformation, similarly to JPEG2000.

The main issue that we would like to pinpoint at the end of this section is that the compression methods are constructed with human perception limitations in mind. This approach saves a significant amount of resources during multimedia content storage and transfer. However, it usually complicates additional processing by video editors, archival of the content, and even machine learning.

## 2.3 Metadata

We may need to store additional data in the multimedia container during the multimedia acquisition and further content processing. To this end, we can utilize two standards: Exchangeable image file format (EXIF) for a couple of the most popular image formats and Extensible Metadata Platform (XMP) for any media content.

The metadata can accompany the multimedia file from recording or ingesting (transfer of multimedia content into the computer) to the final editing, and some of the metadata may even have high semantic value. For example, in the movie industry, some metadata fields may be used to associate an individual take (video file) with the script from the moment it is recorded in the camera.

Other metadata might be added during the ingestion to aid during editing. Many of the fields are also structured or limited to a predefined set of values. Such as the “Basic – Identifier” field should keep an array of unambiguous identifications strings, and “Dynamic Media – Alternate Timecode – Time Value” has to follow a specific format in compliance with the used framerate. Other subsections of the metadata fields should follow the DublinCore specification.

However, outside of the movie industry, the much more common use of metadata platforms is to record technical information required for proper playback or information that is easy to obtain automatically. For example, as the image sensor itself has no information about its orientation in space, all photos taken will be, by default, presented to the viewer in landscape mode. However, by adding a gyroscope to the device,

detecting one of the four possible orientations, and storing it in the metadata field, the viewer can be presented with a correctly rotated image.

In theory, some of these metadata fields may also be used to assist the consequent feature extraction and image processing. For example, we can compensate for the lens distortions and vignetting with a known camcorder model, type of lens, and focal distance. During a panorama stitching, we can, therefore, pre-process all of the images and avoid artifacts or darkened seams.

However, the sad truth is that such metadata is not propagated to the edited copies of the content and adequately modified to match the edits. And even if they are, they are usually removed by most of the content-sharing platforms.

Moreover, in video content, the technical metadata is usually stored only once at the beginning of the recording. So even though the technical metadata may be valid for the first frame of the video, many of the parameters (such as white balance, iris, shutter, gain, focal length) change during the recording. Therefore, we cannot depend on most of the technical information stored in the metadata for video pre-processing either.

# Chapter 3

## Theoretical Background

This chapter will briefly summarize some of the fundamental concepts and approaches used to extract descriptors from multimedia content and their follow-up processing into labels.

### 3.1 Dimensionality Reduction

Many machine learning approaches used later down the data processing pipeline require a significant feature dimensionality reduction. For example, even if we consider a simple modality, such as text documents, one-hot-encoding a single document will require a binary matrix with a shape equal to the vocabulary size times the number of tokens in the document, which may be hard to fit into memory, let alone to train a classifier on it. Therefore, when designing the dimensionality reduction, we need to select a subset of the original object's properties considered essential for the final task. The loss of other properties can be considered a first possibility to design invariance to the irrelevant properties and alleviate the need to design classifiers that can cope with the curse of dimensionality.

#### 3.1.1 Bag-of-words

One of the most straightforward approaches is to deliberately remove the information on the location of the data in a document. In such a setup, we can represent each document by just a single vector with a length equal to the size of the dictionary and values equal to the number of token occurrences in the document.

Although the bag-of-words method is the most popular for text documents, where we can construct the dictionary from  $n$ -grams or word lemmas, other modalities may adopt a similar dimensionality reduction approach. A notable example is the **bag-of-visual-words** method, which takes the salient point descriptors from an image, clusters them into visual words, and represents images by the number of individual visual words present in the picture. Such a representation can be then used for various tasks, such as scene classification [4] or land-use classification [5–6].

To increase the semantic information stored in the resulting vector, we can use a **term frequency – inverse document frequency** (tf-idf) weighting scheme [7]. In such cases, the values in the document description vector do not represent just the count of a particular word from the dictionary but rather the word's relevance to the document and is calculated as:

$$w_{t,d} = (n_{t,d} / \max_t(n_{t,d})) \cdot \log_2(n_D / n_{D,t}),$$

where  $n_{t,d}$  is the count of term  $t$  in document  $d$ ,  $n_D$  denotes number of documents in corpus, and  $n_{D,t}$  is count of documents from corpus that contain the term  $t$  at least once.



### 3.1.2 Bag-of-concepts

Even though the bag-of-words indexing removes the whole word-location dimension from the original one-hot-encoded data and the combination with tf-idf weighting provides the notion of the importance of the word in the context of a given document, the final length of the document descriptor still depends on the size of the dictionary; with the size usually in the order of tens to hundreds of thousands of words.

To reduce the dimensionality of the data and improve further processing, **Latent Semantic Indexing (LSI)** [8] can be performed. Matrices resulting from the LSI, which internally uses a Latent Semantic Analysis [9], can help us compare the overall content of documents because it does not match them word-by-word but with significantly shorter document concept vectors. Moreover, as these concepts internally exploit the co-occurrence of words in one document, we may consider them a partial solution for the disambiguation of synonyms and homonyms in documents without men-prepared lexical databases such as WordNet [10].

From the mathematical point of view, Latent Semantic Analysis uses a modified singular value decomposition:

$$A \approx A_k = U_k \Sigma_k V_k^T,$$

where  $A$  is the original term-by-document matrix,  $A_k$  is the low-rank approximation of  $A$ .  $U_k$  and  $V_k^T$  are orthogonal matrices representing the mapping of a term to concept and, respectively, from a concept to document.  $\Sigma_k$  is a diagonal matrix that corresponds to the scaling matrix in the singular value decomposition and that we truncate to only  $k$  largest singular values (corresponding to the most significant concepts) to be considered in the reconstruction of the original matrix  $A$ . We provide the graphical representation of the modified decomposition in Figure 3.1.

$$\begin{array}{c} \text{documents} \\ \text{terms} \left[ \begin{array}{c} A_k \\ m \times n \end{array} \right] = \begin{array}{c} \text{concepts} \\ \text{terms} \left[ \begin{array}{c} U_k \\ m \times k \end{array} \right] \times \begin{array}{c} \text{concept} \\ \text{weights} \\ \left[ \begin{array}{c} \Sigma_k \\ k \times k \end{array} \right] \times \begin{array}{c} \text{documents} \\ \text{concepts} \left[ \begin{array}{c} V_k^T \\ k \times n \end{array} \right] \end{array} \end{array}$$

**Figure 3.1.** Latent semantic analysis with scaling matrix  $\Sigma$  truncated to  $k$  highest singular values for limiting the number of considered concepts.

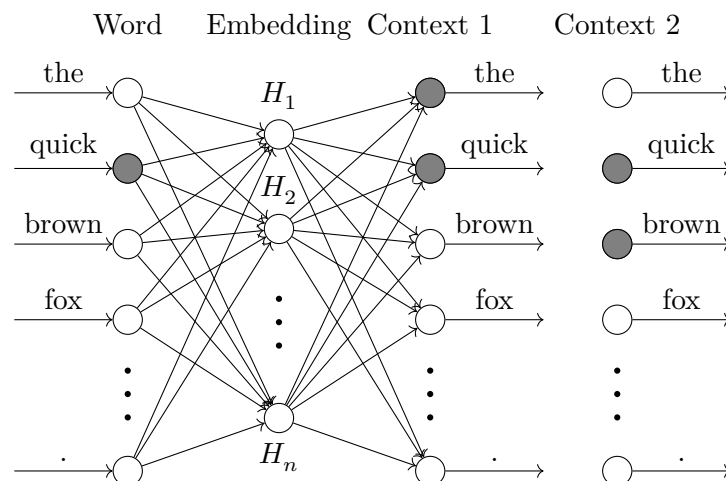
The number of concepts  $k$  can be selected either according to the capacity of the classification or indexing system, based on the experiments with the target domain, or empirically by detecting an “elbow” in the plot of singular values (where adding another concept does not bring much of additional precision to the reconstruction of the original matrix).

Dense concept-by-document matrix is then computed in accordance to [11] as:

$$C_k = \Sigma_k V_k^T.$$

### 3.1.3 Embeddings

A more generic method for extracting a vector description that retains the semantic of a word (we will stick to the text example) is to train an algorithm that constructs a mapping of the dictionary word into a vector in latent space of a limited dimension that



**Figure 3.2.** A simplified example of word2vec skip-gram model training on a sentence “The quick brown fox jumps over the lazy dog.”. In this example, with the context of length one, an input word “quick” generates two output contexts “(the, quick)” and “(quick, brown)” that we use as two data points for the training. Grey input and output nodes represent a value of one, white input and output nodes represent a zero.

either predicts the context of the input word or predicts a word from a given context. As the latent space is trained to encode all of the information needed to reconstruct the word’s context, we assume that words commonly occurring in a similar context will have a small distance also in the latent space.

In theory, the bag-of-concepts approach satisfies these properties as the individual entries in the word-by-concept matrix can also be considered embeddings from words to vectors in the concept space. However, the previously mentioned matrix decomposition approach considers the whole document for context.

To diminish the dependency of individual words on the whole documents, the approaches such as **GloVe** [12] utilize the information from the local context window and trains the embedding algorithm using a global word co-occurrence count matrix.

One of the other approaches to learning the semantic vector is to use the context directly and leverage the universal approximation power of neural networks. The most notable example of such neural network embedding training is the **word2vec** approach [13] which represents the most straightforward example of a fully connected network with just three layers, as illustrated in Figure 3.2:

- input layer with the one-hot-encoded word representation,
- one hidden layer with a limited width to capture the semantic vector, and
- an output layer with the one-hot-encoding of the context.

Word embeddings trained on large corpora also revealed one surprising property: we can resolve semantic relationships between words by vector arithmetic on its embeddings. The prototypical example being:

$$\textit{embedding}(\textit{king}) - \textit{embedding}(\textit{man}) + \textit{embedding}(\textit{woman}) \approx \textit{embedding}(\textit{queen}),$$

where the embedding can be provided either by skip-grams with negative sampling (word2vec) or GloVe.

The basic idea hidden behind this property is that we can consider some semantic relationship between words (“king” is to “man” as “woman” is to “queen”) as an inherent

feature of the training data. In other words, we can add the same vector to the word embedding of the word “man” and “woman” to get their royal counterparts (“king” and “queen” respectively) even though such semantic relationship is not defined explicitly by any dictionary.

A paper on understanding the origin of linear semantic relationships between words [14] discusses this property of the word embeddings in broader detail, including the proof on both noiseless skip-grams and GloVe.

### ■ 3.1.4 Histograms

Another straightforward yet effective dimensionality reduction technique is to transfer the numerical input data into histograms. This approach also removes the information on the data location in the document and introduces a binning strategy to reduce the data dimensionality further.

Traditionally, we use histograms to approximate the distribution of color intensities in image data or frequency bands in a sound recording. However, we can aggregate many other data properties in a histogram.

For example, the **Histogram of oriented gradients** (HoG) accumulates the orientation of the texture gradients (we will discuss the visual features more thoroughly in the next chapter) in the image divided into smaller sub-images, called “cells.” Further on, we can use the combined histogram entries as a feature vector describing the object in the scene. In combination with appropriate contrast normalization and histogram collection over a variable window, we can use a relatively simple classifier, such as linear Support Vector Machine, for pedestrian detection in static images [15].

For video content, **Histogram of oriented optical flow** combines the description gathered by the HoG from the first frame with optical flow obtained from consecutive image frames, sometimes simplified to a motion boundary. One of the first uses of such descriptor was again to detect pedestrians in the video stream from a static camera [16]. Later research used this descriptor in event detection [17–18] and other action classification tasks [19].

## ■ 3.2 Distance Measures

Because we pre-process most of the text and multimedia data we use in our research into vector or histogram representations, we need only a couple of basic distance measures.

### ■ 3.2.1 Euclidean Distance

Once we transfer the data to some  $n$ -dimensional Euclidean latent space, the Euclidean distance (also known as L2 distance) is the most commonly used distance measure. For  $n$ -dimensional vectors  $P$  and  $Q$ , the Euclidean distance is defined as:

$$d_{Euclidean}(P, Q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2},$$

where  $x_i$  represents the  $i$ -th element of vector  $X$ .

### ■ 3.2.2 Manhattan Distance

In many applications, the Manhattan distance (also known as L1 distance or City Block) can be used as an excellent alternative to the Euclidean distance. The main benefit of the Manhattan distance is a faster computation time and higher numerical stability in

high-dimensional spaces. The L1 norm may even contribute to higher accuracy in some cases. However, unlike the Euclidean distance, it may hide some semantic properties of the latent space. For  $n$ -dimensional vectors  $P$  and  $Q$ , the Manhattan distance is defined as:

$$d_{Manhattan}(P, Q) = \sum_{i=1}^n |p_i - q_i|.$$

### ■ 3.2.3 Cosine Distance

The cosine distance has rather specific semantics. Instead of comparing the absolute positions of the data points in the hyperspace, the cosine distance compares their orientation relative to the origin. Therefore, we commonly use the cosine similarity on low-level feature descriptors (such as bag-of-words document representations) that may not have high-level semantics of their own. Moreover, as the cosine distance represents just the angle between the vectors starting in origin and ending in the compared data point, the data do not have to be normalized for the document size beforehand. Instead, the cosine distance formula already includes such normalization (which we can omit if we normalize all data points to a unit hyperball). For vectors  $P$  and  $Q$ , the cosine distance is defined as:

$$d_{cosine}(P, Q) = 1 - \frac{P \cdot Q}{\|P\| \|Q\|},$$

where the centered dot ( $\cdot$ ) represents a scalar product of two vectors in the same Euclidean space and  $\|X\|$  denotes the Euclidean norm of the vector  $X$ .

### ■ 3.2.4 Hamming Distance

We can use a Hamming distance to compare strings or vectors of the same length by the number of elements that differ. However, in the context of this thesis, we will use this measure only to compare binary vectors. In such case, the definition of Hamming distance of two binary vectors  $P$  and  $Q$  is the very same as for the Manhattan distance:

$$d_{Hamming}(P, Q) = \sum_{i=1}^n |p_i - q_i|.$$

One of the significant advantages of the Hamming distance on binary strings is its simplicity. Practically, we can compute the distance as:

$$popcnt(P \oplus Q),$$

where  $\oplus$  operator represents the XOR operation on individual bits and *popcnt* (population count) function counts the number of bits set to 1. Both operations are available as native instructions on virtually any modern hardware.

### ■ 3.2.5 Dynamic Time Warping

As we deal with time series in our research, we need a distance measure to compare two data series. However, at the same time, we want to allow some flexibility in the alignment of the two signals before computing their difference. One of the most commonly used time series alignment and similarity computation measures is the Dynamic Time Warping algorithm (DTW) [20].

In a nutshell, we can calculate the dynamically time-warped distance of two series  $P$  and  $Q$  of lengths  $m$  and  $n$  by finding  $DTW_{m,n}$  with a recursive formula:

$$DTW_{i,j} = dist(p_i, q_j) + \min(DTW_{i-1,j}, DTW_{i,j-1}, DTW_{i-1,j-1}),$$

where  $dist$  can be any distance measure suitable for the data at hand and  $DTW_{i,j}$  for  $i$  or  $j$  smaller than zero is defined as zero.

### 3.3 Convolution

Many signal processing methods need to apply a specific filter over the incoming signal. Intuitively, we can provide this functionality by multiplying the original signal (a function of time) with a mask function across the whole time domain in a process called convolution. On continuous functions, the convolution is defined as:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt,$$

where  $f$  is the input signal function and  $g$  is the filter function reflected about the  $y$ -axis to stay consistent with the concept similar to moving weighted average.

In digital audio and video processing, where we have to deal with sampled discrete-time signals, we replace the integral with a sum. Also, as both the signal and the filter have limited support, we can replace the infinities with finite starting and stopping points of the summation:

$$(f * g)[n] = \sum_{m=-M}^M f[m]g[n-m].$$

The real-world applications of the convolution are then dependent on the type of the filter, in the discrete case usually called the convolution kernel.

One possibility is to use a well-known function to determine the kernel values. The convolution with such kernel then effectively executes some processing method on top of the original signal. For example, we can efficiently remove high-frequency bands from the signal by using the Gaussian convolution kernel.

The other commonly used approach is to set the convolution kernel to a specific pattern we want to search for in the incoming signal. For example, convolution with a kernel  $[-1, 1]$  and detection of positive response will provide us with the positions of rising edges in the incoming signal.

Last but not least, the convolutional layers in neural networks are based on the very same principle of pattern detection. However, the convolution kernel is part of the trainable parameters.

### 3.4 Deconvolution

A useful principle is also hidden in the reverse process. If the data acquisition is subjected to a deterioration that we can model purely with a convolution, such as certain classes of image blur, the reverse process may lead to a successful restoration.

However, in real-world scenarios, the acquisition process is more likely to be subject to noise:

$$h = (f * g) + \epsilon,$$

where  $h$  is the acquired signal,  $f$  is the actual signal,  $g$  is the convolution function modeling the issues in the acquisition process, and  $\varepsilon$  is the additive noise. However, in many cases, the precise convolution kernel is unknown, and we may derive only the type of the kernel, not its precise parameters.

### 3.5 Fourier Transform

Another approach to simplify many of the operations on the incoming signal is to translate it from a time (or space) domain to a frequency domain.

On a continuous signal, the Fourier transform of a function  $f$  is defined as:

$$\mathcal{F}\{f(x)\} = \hat{f}(u) = \int_{-\infty}^{\infty} f(x)e^{-2\pi iux} dx,$$

and its inverse:

$$\mathcal{F}^{-1}\{\hat{f}(u)\} = f(x) = \int_{-\infty}^{\infty} \hat{f}(u)e^{2\pi iux} du,$$

which can be, similarly to the convolution, converted to operations on discrete sequence:

$$F[k] = \sum_{n=0}^{N-1} f[n]e^{-2\pi ikn/N},$$

$$f[n] = \frac{1}{N} \sum_{k=0}^{N-1} F[k]e^{2\pi ikn/N}.$$

One of the additional benefits of working in the frequency domain, known as the Convolution theorem, is that convolutions in a time (or space) domain correspond to ordinary multiplication in the frequency domain:

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}.$$

Another exciting concept connected to the Fourier transform is called *cepstrum*. This concept is sometimes denoted as a spectrum of a spectrum and, therefore, we can use it to investigate periodic structures in the frequency spectrum of the original signal. The concept's name is a wordplay on the word "spectrum." Similarly, the results are not presented in the frequency domain but in the "quefrequency" domain.

The original definition of the power cepstrum is as follows:

$$|\mathcal{F}^{-1}\{\log(|\mathcal{F}\{f(x)\}|^2)\}|^2,$$

however, many variations are used. Usually, the common factor is that the signal is transformed to a frequency domain, modified with a non-linear function, and either transformed back to the time domain or processed by the Fourier transform once again (leading to a similar result, just differently scaled, as noted in Section 5.3.3. of [21]).

One of the most notable examples is the Mel Frequency Cepstra, such as in [22], which adds a rescaling of the first frequency spectra by a mel scale, equalizing human-perceived distances between the frequency bands.

## 3.6 Discrete Cosine Transform

As an alternative to the Discrete Fourier Transform (which is a transformation between sequences of complex numbers and uses the complex exponential function to construct the function base), we can use a base consisting of only cosine functions to transform sequences of real numbers to sequences of real numbers.

There are several types of discrete cosine transform, however, the most commonly used is probably the DCT-II:

$$F[k] = \sum_{n=0}^{N-1} f[n] \cos\left(\frac{\pi}{N}(n + 1/2)k\right).$$

## Chapter 4

# State-of-the-art Multimedia Processing Methods

To establish a common ground for our research, we will provide an overview of some methods used to process multimedia signals that we build upon in this thesis.

### 4.1 Text Extraction and Processing

The destination modality that is the easiest to process is inarguably the text data. However, we usually do not store much of the direct text information in multimedia files. In the case of content published on some multimedia sharing platform, we usually get only a short title and possibly some description that may be only partially related to the content of the multimedia clip. In specific cases where the authors wanted to (or had to) make their content more accessible, human-made closed captions may be available. However, there is usually not even a short description or semantic file title in many private archives.

On the other hand, multimedia files commonly contain an image of a written text or sound of a human speech, which can be transcribed into text documents without significant loss of semantics. However, as we need to introduce some state-of-the-art sound and image processing methods first, please refer to Sections 4.2.1 and 4.4.2 for more information.

### 4.2 Audio Signal Processing

Similar to the processing of text documents, where the individual letters do not carry much semantic information and have to be combined into words, sentences, and paragraphs, the audio signal cannot be processed as individual samples.

To this end, one of the most commonly used multimedia description standards, the MPEG-7 (thoroughly described in a book [23]), defines 17 low-level audio feature descriptors that can be used for various tasks in audio processing—features ranging from audio power and spectral centroids to its harmonicity.

Some approaches then benefit from building their classifiers on top of the complete spectral information instead of just a couple of statistical properties.

#### 4.2.1 Automated Speech Recognition

One of the most effective audio processing methods for multimedia indexing is the extraction of speech and transcription to a text document that takes significantly less storage space and can be searched much more efficiently.

In theory, automated speech recognition (ASR) can be carried out by detecting specific frequency patterns in the audio signal, classifying them into a set of phonemes, and converting a sequence of phonemes into the most probable word. However, it is still



challenging due to different speakers, environmental noise, poor audio signal acquisition, and many others, such as discussed in [24].

The usual approach to mitigate issues with detecting phonemes in a noisy sound is to use some frequency spectrum or possibly its envelope or cepstrum (spectrum of a spectrum, see Section 3.5), mapped on a mel scale, such as in [20]. However, as discussed in [25], the Mel Frequency Cepstral Coefficients may not be the best choice, even though it is the most common. Instead, they propose to use frequency filter banks that seem to be more suitable for speech representation. However, with the advent of deep neural networks, such as [26], the preprocessing of the signal is getting less dependent on engineered features.

Hidden Markov Models are the traditional approach [27] to transfer the sequence of phonemes into words. However, as the flexible time dependency can also be modeled with specific neural network layers, approaches based on recurrent neural networks, such as proposed in [28], are starting to pave a way to end-to-end speech recognition with neural networks.

### ■ 4.2.2 Audio Fingerprinting

To enable efficient search in the audio database by providing an example, we have to turn to feature extraction methods that produce compact descriptors with high discrimination power.

Shazam is one of the prominent examples of a service that provides search in a database of popular music by a low-quality recording of the audio signal from a hand-held device. According to [29], the service uses overlapping calculations of Fourier transform in regular intervals and keeps only such time-frequency bins with a higher power level than the surrounding bins. The resulting spectrogram peaks of a possibly noisy signal are then matched against spectrogram peaks from the noiseless music signal of each stored song. Shazam internally implements significant search optimizations by introducing a hashing algorithm, but this is out of the scope of this thesis.

One of the papers [30] associated with the authors of YouTube Content ID reveals a possibility that their system uses Gaussian mixture acoustic models in combination with weighted finite-state transducers to capture song structure to produce the fingerprints. Such an approach would be less sensitive to time drift in the audio signal and (unlike Shazam) enable to match mashups and covers of the original music. This feature, in turn, allows the Content ID system to be used for content management and copyright protection of it.

Other fingerprinting approaches may leverage the fact that almost all contemporary music contains some melody that can be transcribed into musical notation. One of the possibilities is to index the sheet music directly and allow the user to enter a fragment of the song in musical notation or by humming [31], which is transcribed to the musical notation by methods of the central melody extraction such as [32]. Another approach is to collect the data in the same types of representations as the users will use for the query. To stay with the music example, Midomi [33] allows to search by humming the main melody and comparing to records of other users humming.

To assist with the extraction of the main melody, we may use a method of music source separation such as Open-Unmix [34]. This method takes mixed-down audio and separates some of the audio tracks – namely vocals, percussion, and bass – with a bi-directional long short-term memory (LSTM) neural network.

## 4.3 Still Image Processing

Although we are primarily interested in video sequence processing, we can transfer many approaches from the research area of still image processing. After all, we may consider the video sequence as nothing more than a rapid series of still frames.

Again, the individual pixel values do not carry much of a semantic value and, therefore, we need to extract some features from the image.

### 4.3.1 Edge Detection

One of the essential features is a visual edge, as it may correspond to an object's edge, change of material, or other vital semantic aspects of a part of an image.

As we are primarily interested in high-frequency edges that correspond to object edges, a naïve approach would be to filter just the high frequencies by subtracting a blurred image from the original one. However, such a simple approach would amplify the noise, possibly making the detection useless.

A commonly used approach to edge detection is to convolute the input image with a small gradient-detecting kernel such as Roberts cross [35], Prewitt operator [36] or Sobel operator [37], as showcased in Table 4.1. The size of the kernel is deliberately selected to be small to provide the strongest response on the pixels with a high gradient in their closest neighborhood. As we use a separate kernel to detect the gradient in the horizontal and vertical direction, we combine the magnitude of the two responses as  $G = \sqrt{G_x^2 + G_y^2}$ .

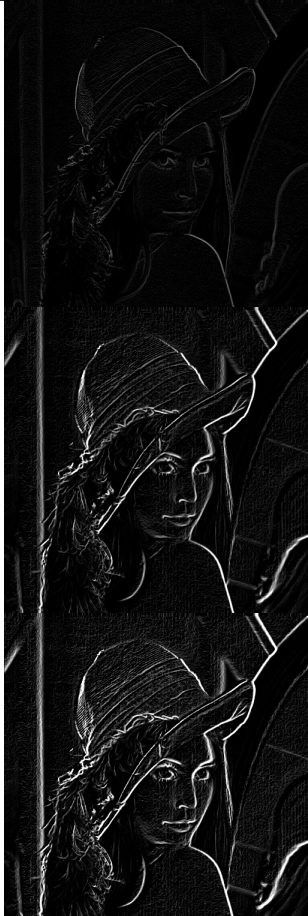
However, these gradient detectors are still susceptible to noise, and we have to use relatively high thresholds to obtain edge-like structures. Therefore, more sophisticated edge detection methods such as Canny [38] and Deriche [39] use an image smoothing operator first to remove the noise and then use gradient detection to propose edge candidates, filtered to obtain just a set of strong edges. One of the possible outcomes of the Canny edge detection algorithm is shown in Figure 4.1.



**Figure 4.1.** Result of Canny edge detector,  $1\sigma$  Gaussian blur and 10 – 30% threshold.

We can use the detected edges, for example, in a naïve object detection from a silhouette, where the edge is coarsely discretized, approximated by a B-spline curve, or encoded in a radial function. As an artifact of the early image digitization with a meager resolution, the outline of the captured object may be captured in a Chain code – a set of instructions for recreating the approximation of the outline in a discrete

**Table 4.1.** Comparison of discrete gradient detectors.

Name	$G_x$	$G_y$	Resulting image
Roberts cross	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	
Prewitt	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	

grid. Even though a Hamming distance may be theoretically used on such a set of instructions to assess the similarity of object outlines, significant research effort was dedicated to reversing such encoding [40].

We illustrate some of the encoding options in Table 4.2. These descriptors are usually constructed to be at least scale-invariant (in polygonal approximation and chain code, the grid size is pre-selected) or scale and rotation invariant (shape vector and shape matrix, the first value is the one of highest magnitude) to enable simple comparison of the detected object outlines.

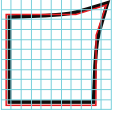
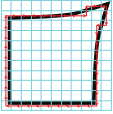
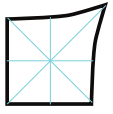
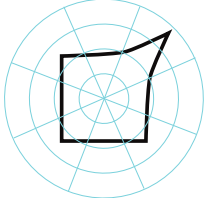
However, in real-world applications, a more sophisticated approach to assessing the object outline similarity is based on a set of two-dimensional moment invariants, such as the seven Hu Moments [41]. These moments are invariant to translation, scale, rotation, and reflection if we omit the seventh invariant (encoding the skew of the image).

The theory behind the sets of rotation moment invariants is then developed further in the work of Jan Flusser [42], providing an extended set of independent and complete invariants.

### 4.3.2 Interest Point Detection and Description

Another important low-level feature that we can extract from the image is a set of positions in the image that are visually distinct from their surroundings. For example, positions where two edges cross or a single edge makes a sharp turn.

**Table 4.2.** Edge information encoding.

Name	Illustration	Data
Polygonal approximation		$[0,1], [7,1], [10,0], [9,3], [9,10], [0,10]$
Chain code		$2^*D, L, 8^*D, 9^*L, 9^*U, 8^*R, U, 2^*R$
Shape vector		$(107, 59, 82, 58, 82, 58, 82, 59)$
Shape matrix		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

Although the interest points are technically not limited just to visual corners, the nomenclature can sometimes be mixed up. Therefore, for the purpose of this thesis, we will consider the names “visual corners”, “interest points”, “salient points” and “keypoints” as practically equivalent. Please keep in mind that in the context of image processing, they may also be called just “features.”

In this subsection, we will describe three of the most popular interest point detectors and descriptors: the Scale-invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), and a combination of Features from Accelerated Segment Test (FAST) interest point detector combined with a Binary Robust Independent Elementary Features (BRIEF) descriptor known as Oriented FAST and Rotated BRIEF (ORB).

**SIFT** is an algorithm for detection and description of local visual features in images, published by David Lowe in 1999 [43]. This algorithm takes the input image, creates a scale space of four octaves (halving the resolution) with five scales. Each has a different Gaussian blur, starting with  $\sigma = 1.6$  and  $k = \sqrt{2}$ , transferring the middle value to next octave. On three subsequent layers of these differences of Gaussians, we then search for local extrema. This way, we gather not only information about interest point position but also the scale.

The resulting extrema points are refined, the ones with very low intensities or location on image edges are removed. Final interest points are then assigned a majority orientation based on gradients sampled from their neighborhood.



**Figure 4.2.** Comparison of SIFT, SURF and ORB interest points. Circles denote the scale of the interest point and the inscribed line the direction of the feature.

Description of the individual interest points is then provided by creating an 8-binned gradient-direction histogram of  $16 \times 16$  surrounding blocks, collected on  $4 \times 4$  sub-blocks. Therefore, SIFT creates a vector of 128 real numbers for each detected interest point. Euclidean distance is then most commonly used to compare the resulting SIFT descriptors.

**SURF** [44] is a mere enhancement of the SIFT descriptor. The Laplacian of Gaussian used by SIFT for the interest point proposal is approximated with a Box filter in SURF, and both orientation assignment and feature description are gathered from wavelet responses. In the case of SURF, we consider  $4 \times 4$  sub-regions around the salient point, each being described by four coefficients of the wavelet responses. SURF, therefore, creates by default a vector of 64 values for each interest point. Due to the similar data semantics to SIFT, Euclidean distance is also commonly used to compare the SURF descriptors.

A relatively new interest point descriptor called **ORB** (Oriented FAST and Rotated BRIEF) [45] uses a modified FAST algorithm for interest point detection. Their approach searches for many point proposals with a significant difference in intensity to pixels in a circular neighborhood in several layers of a Gaussian pyramid. Such proposals are then ordered according to the Harris corner measure to select only the best points.

Similar to the other interest point extraction algorithm, the point is assigned a direction from the detected point to its intensity centroid, and the feature size is deduced from the currently processed level of the Gaussian pyramid.

BRIEF (Binary Robust Independent Elementary Features) [46] descriptor is then used to describe the neighborhood of the selected interest point. However, it is modified to take into account the interest point direction. This process is called “steering”. As this descriptor produces binary vectors in the default setup, we should use a Hamming distance for comparing the descriptions.

In addition to that, the ORB visual features have one attractive property. Based on the research of [45], two points can be considered visually similar if the Hamming distance is smaller than 64. Thanks to this property, we do not need to apply additional tests on the proposed matches, opposed to the example of the ratio test proposed in [43] for the SIFT detector.

### 4.3.3 Fiducial Marker Detection

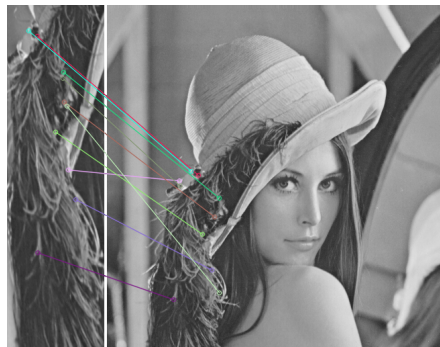
Fiducials are distinct markers (usually holding some additional information for identification) used especially in robotics and motion capture for estimating distances and

orientation. Their detection is a task somewhat similar to visual corner detection. However, instead of detecting single pixels with large gradients in their vicinity, detectors such as [47] use various techniques to detect structures specific for a given type of fiducial marker. For example, in the case of ARTags or AprilTags, the tag is constructed from black and white squares and, therefore, the detector is trying to search for long edges with high contrast to fit a quadrangle and read the information stored in the fiducial.

However, the principles of fiducial detection can also be used without specific physical markers. For example, the face detection algorithms usually provide an additional set of points marking the positions of the most important facial features [48]. These points can then be used either to normalize the image of the face for consequent tasks in image recognition or to be directly used as numerical descriptors of the face. For more information on facial fiducials, please refer to [49].

#### 4.3.4 Object Detection

Another step in the complexity of the detection algorithms is the detection of objects. Searching for an exact copy of the query image in the destination image would be straightforward and solvable with a single convolution. However, any transformation of the destination image would break such a fragile approach. Therefore, to enable robust object detection, many invariants are employed.

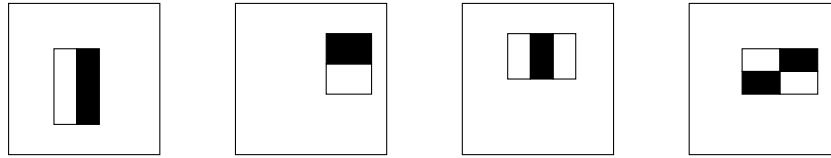


**Figure 4.3.** Visualization of the ten closest visual matches from the query image on the left to the destination.

Staying with the example of detecting a visually identical object to the query image, we may use the **registration of interest points** we discussed earlier in this section. By extracting the descriptions of the interest points from both the query and the target image and matching them to one another, we are able to obtain some correspondences, such as in Figure 4.3. Based on that and on the assumption that the visual patch is deformed only by a projective transformation, we can directly register the patch onto the target image (with filtering of the outliers, for example, by Random Sample Consensus [50]) and, therefore, return the detected object's location.

However, in many cases, we want to search for all objects of some type; rather than a specific object. To this end, we need to use a classifier that decides if the object type of our interest is present in the target image or not. Moreover, to allow better invariance to scale and position, the classifier is usually executed on multiple levels of the Gaussian pyramid and in a moving window.

One of the earlier approaches was to train a **cascade of Haar-like features**. The computation of a Haar-like feature is straightforward – it is just a difference of sums of



**Figure 4.4.** Example Haar-like features relative to the enclosing detection window. The sum of pixels in white rectangles is subtracted from the pixels in black rectangles. From left to right: vertical edge, horizontal edge, vertical line, and four-rectangle feature.

adjacent areas in the picture (examples in Figure 4.4). Using these features, we may then capture some fundamental concepts hidden in the image, such as that on a picture of a face, the horizontal stripe around the eyes is usually darker than the forehead and cheeks. However, the hard part that we left for the classifier to learn is selecting the most helpful features. To improve the performance, the classification step is then divided into stages where the most discriminative Haar-like features are tested first, and only if we have some suspicion that the object may be present, we start the next stage of the cascade. For more information on Haar cascade object detectors, please refer to [51].

With the advances of neural networks and the availability of computational power, the engineering of such visual features almost stopped, and most of the effort moved to the tuning of **deep (recurrent) convolutional neural networks**.

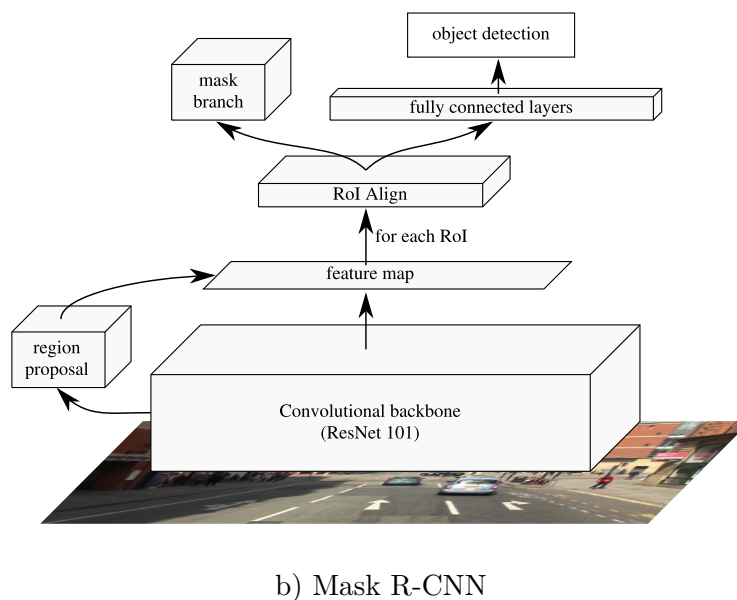
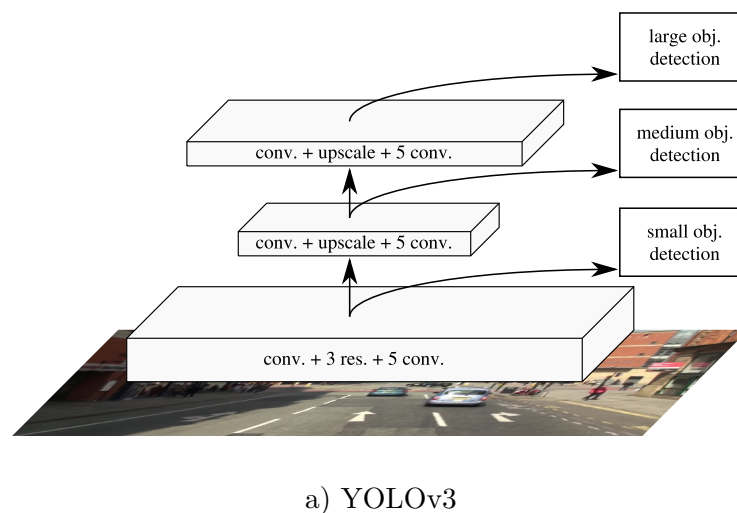
**You only look once** (YOLO) [52] is an advanced, state-of-the-art real-time object detector that uses a single network evaluation for all object predictions.

A significant difference from other object detectors is that YOLO does not rely on a low-level feature extraction backbone network and does not use a separate region of interest proposal and a follow-up object classification network. Moreover, the core of YOLO also consists of convolutional layers and residual blocks only.

With such a simple architecture, YOLO provides excellent throughput for both training and inference. However, as object detection inherently uses a classification of bounding boxes and selection of the ones with the highest confidence, direct extensions to object segmentation are not feasible.

On the somewhat opposite side of the spectrum of convolutional neural networks for object detection is a method called **R-CNN** – Regions with CNN features. This approach uses an independent algorithm for the proposal of regions in the picture, warping each proposed region to a fixed size and pursuing the classification on each warped region. The original R-CNN [53] works with the external proposal of 2000 regions and uses a convolution neural network and SVM classifier on the original image data, separately for every proposed region. The repeated convolution on different yet overlapping areas of the picture makes this approach very slow. In the later iterations on the R-CNN approach, Fast R-CNN [54], and Faster R-CNN [55], the so-called convolutional backbone is executed once on the whole frame. The feature map extracted by the convolution is then sliced by independent region proposal (in Fast R-CNN) or with the help of region proposal network (in Faster R-CNN) to create a pool of regions of interest (RoIs) that are classified just by fully connected layers.

Although inherently slower than YOLO due to a more complex architecture, this approach allows connecting other image processing branches on the level of proposed RoIs. Starting with the Fast R-CNN, the regions of interest consist of features extracted from the convolutional backbone and are, therefore, directly suitable for object segmentation by inferring a mask of the detected object with an additional detection branch. This approach is known as Mask R-CNN [56].



**Figure 4.5.** Simplified block diagram of neural network architectures for object detection.

The major drawback of R-CNN approaches is that the inference speed and model size is still unsuitable for real-time video processing deployment. Both Faster R-CNN and Mask R-CNN can process only five frames per second on a K40 GPU.

### 4.3.5 Image Classification

Traditionally, image classification was also a domain of engineered features and robust classifiers. Earlier in this chapter, we already discussed features based on edge descriptors or interest points. These features can be used to train various classifiers, such as Support Vector Machines, possibly with nonlinear kernels, like Radial Basis Functions. However, in theory, even the raw pixel values may be used as features with a small enough resolution and a vast collection of samples, such as the MNIST dataset.

However, the recent approaches in image classification also shifted towards end-to-end training where the network gets the pixel data on input and is trained to produce the output label directly.



The **very deep convolutional network for large-scale image recognition** [57] known as VGG, can be hardly considered very deep by today's standards. The most commonly used VGG-16 network has only 16 convolutional layers with kernels of size  $3 \times 3$  for image preprocessing and three fully connected layers for the classification. However, the first fully connected layer (of size  $25088 \times 4096$ ) contributes to a large number of trainable parameters (over 138 million). Even with this rather simplistic architecture, the VGG-16 achieved top-1 accuracy of 74.4 % and top-5 accuracy of 91.9 % on ImageNet.

The most commonly used variant of the deep residual network **ResNet-101** [58] that was released two years later improved the top-1 accuracy to 78.25 % and the top-5 accuracy to 93.95 % on ImageNet. And although the depth of the network increased to 101 layers, the number of trainable parameters dropped to 43 million.

The field of neural-network image classification is still rapidly evolving. Novel approaches, such as EfficientNet [59], improve the classification performance even more, despite introducing limitations to network hyper-parameters, and the currently best image classification approaches are based on Vision Transformers [60]. This development proves the extreme flexibility of neural networks as universal approximators. On the other hand, as the neural network structure becomes increasingly complicated, there is usually no possibility to investigate the reason for activations in hidden layers and re-use them for other tasks at hand. Therefore, the VGG and ResNet backbones are still prevalent in many fields of computer vision.

#### ■ 4.3.6 Scene Recognition

In the scope of knowledge extraction approaches, we may consider scene recognition as a particular case of image classification, with the only significant difference that image classification is mainly concerned about the object in the foreground. In contrast, foreground objects usually present a distractor for scene recognition methods.

The primary task of scene recognition is to get a label describing the general properties of the scene based on the image or set of images from the same setting.

In the domain of still images, the research in scene detection is relatively vivid, ranging from approaches that derive scenes from detected objects [61] or interest points [62]. Again, more approaches used deep neural network classification to process the images directly [63–65].

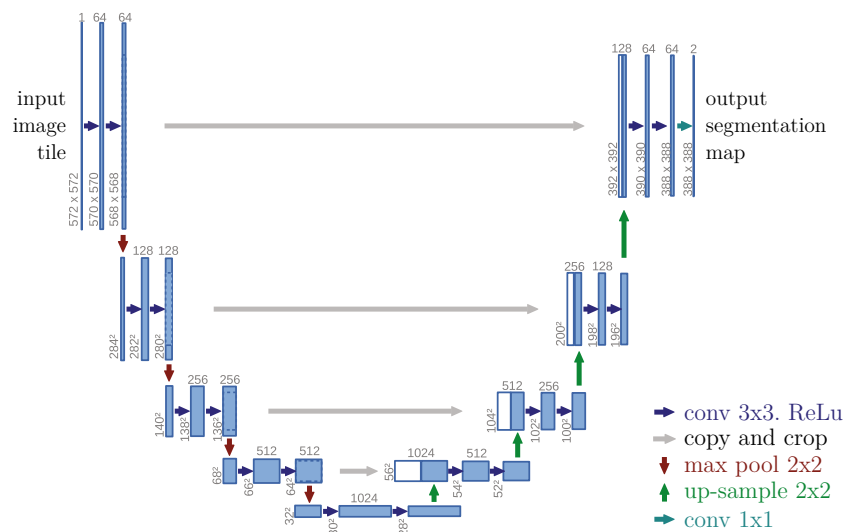
#### ■ 4.3.7 Image Segmentation

The task of image segmentation can be, in theory, considered an extension of the image classification that provides the target class not to the whole image but rather to a set of pixels in the image even though the traditional feature extraction methods (sometimes deliberately) drop the association of individual pixel location to the extracted feature.

Therefore, the traditional pixel-wise segmentation could be done separately, only after obtaining seeds from some classification or clustering algorithm. In many cases, we even need help from some image structure extraction (detection of edges or connected components) and possibly need to resolve min-cut / max-flow graph cutting problem (for example, with [66]) for the colliding sections of the image.

With neural network architectures, such as Mask R-CNN [56], U-net [67], LinkNet [68], or PSPNet [69], the training of a semantic segmentation does not have to rely on any of such processing anymore.

The basic concept of such architectures is that they introduce a counterpart to the classification (an encoder made with convolutions and poolings) based on more convolutions and a trainable up-sampling (a decoder). Data resulting from the convolutions



**Figure 4.6.** Visualization of U-net architecture, adapted from [67].

are copied over before pooling to the corresponding layer of the decoding process (via so-called “skip connections”) to improve associations with the original pixel space.

During our research, we conducted experiments only with some of the approaches. For example, with Ondřej Pírko, we hypothesized that semantic segmentation of satellite imagery into EuroSAT land-use classes with modified VGG16 classifier might provide additional information for the refinement of the digital elevation model in [B2]. With Jan Kostecký, we experimented with a detector of dams and embankments on the same satellite imagery with a modified U-Net in [B1].

## 4.4 Video Processing

The processing of video content can be, in theory, carried by the processing of the individual frames and reducing just the acquired knowledge in time. However, such processing would be wasteful in many cases as the same information will be extracted repeatedly. In other cases, the methods may benefit from execution on top of a frame sequence.

### 4.4.1 Individual Frame Reconstruction

The first step of any processing is actually to reconstruct individual frames from the video stream. Although this might seem like a task just for the video decoder, the major disadvantage of the video codecs is that they are tailored to human perception, and the knowledge extraction may depend on features deemed unnecessary for viewers.

As a specific example, video codecs with interframe compression typically cannot cope with many tiny, randomly moving objects at once (such as glitters, confetti, or bouncy balls). The compression tries desperately to focus on at least some of them, but as the movement is too random, some objects can stay in the same spot, disappear or move unpredictably until we decode a new keyframe (compressed with just the intraframe method).

There are some efforts to remove such compression artifacts, such as [70]. However, their actual effect on consequent machine learning is yet to be confirmed.

### ■ 4.4.2 Optical Character Recognition

To support our claim on wasteful repeated processing, we would like to present a specific example of text recognition from a lecture recording. In the simplest case, only the presentation slides are recorded; however, they are recorded in a video to capture the lecturer's voice.

To avoid repeated processing of the same slide dozens of times per second, it is, therefore, advisable to detect transitions in the video signal, and split the video signal back into a set of static slides. Several methods of such slide transition detections in various scenarios are discussed in the Bachelor's thesis of Richard Burkoň [B3].

Once we have the individual images, the optical character recognition can be divided into several fundamental steps. First, we need to find the lines by detecting connected components, filtering the resulting blobs, and fitting text baselines, as discussed in [71]. Then, based on the edges of the connected components, we have to separate individual characters and match their polygonal approximations against a classifier trained on multiple typefaces. The whole process is well documented on an example of optical character recognition system Tesseract [72].

### ■ 4.4.3 Video Shot Detection and Shot Sequence Segmentation

Similar to the task of slide detection, the processing of generic edited video content consisting of multiple shots also benefits from the detection of shot boundaries and separate processing of the individual shots. However, the motivation is slightly different, as we primarily want to limit the consequent video processing to semantically coherent units. In addition to that, the detection method is also different; please refer to [73] for a survey.

For example, separation of an edited recording of dialogue into individual shots would allow us to assign names of the speakers to the individual separated shots (in the ideal case by detecting the speaker's face just once), as well as to provide adequate labels to the additional content, such as illustrative shots, that would otherwise confuse the detection algorithms used for speaker detection.

The resulting sequence of semantically described clips can then be used as an additional source of high-level information. In line with the example mentioned above, a detected sequence of shots alternating between two people can be assigned with a label for a dialogue. Moreover, extraction of such semantic descriptors does not necessarily depend on the explicit metadata, as the concept of alternating shots can be discovered by clustering the shots, such as described in [74].

Many other aspects of shot detection and sequence segmentation are concisely summarized in [75]. And in many of our experiments, we use the automatic shot segmentation provided by PySceneDetect [76], specifically we obtained the best results with the content-aware detector.

### ■ 4.4.4 Frame Registration and Action Recognition

Once we have the video signal separated into individual shots, we may relatively safely assume that the individual frames in the given shot can be registered one onto the other. Under this assumption, we can construct a motion field that estimates the motion between the individual frames.

One of the possible applications for image registration is motion stabilization or smoothing [77], which tries to find an optimal (possibly non-rigid) transformation from one frame to the other, eliminating unwanted camera motion.

During our experimentation with stabilization of timelapse footage recorded from an unstable pole, we obtained the best result with brute force matching of ORB salient points by Hamming distance, prefiltering obvious outliers by distance, and estimating 2D affine transformation with four degrees of freedom (limited to translation, rotation, and uniform scaling) with random sample consensus (RANSAC).

A different use-case for the registration may be an introduction of super-resolution. If the object moves or the camera shakes, the object is registered on slightly different pixel points of the sensor. As some image registration approaches have a sub-pixel precision, this additional information can be used to reconstruct the recorded object in a higher resolution [78].

Another possible application for such an extracted optical flow description is the recognition of human action. A naïve approach is to represent the optical flow of each frame pair by a histogram binned by size and angle and later classified into a set of actions. However, such a rudimentary approach fails miserably when the camera starts to move around the scene. Therefore, much more sophisticated approaches, such as [79], are needed.

#### ■ 4.4.5 Simultaneous Localization and Mapping

Instead of registering whole frames with an estimation of a single transformation matrix, we may segment the image into several facets and estimate the transformation only on the subset of salient points. In theory, this allows tracking of multiple rigid objects in the scene and simultaneously estimating the camera's motion in the scene. However, the tracking approaches have to make several trade-offs as the camera motion introduces new degrees of freedom.

The Simultaneous Localisation and Mapping (SLAM) approach, originating from the computer vision and robotics research [80], is aimed at rapid scene reconstruction that allows using high-level scene and object information for successive description. Also, such object description does not have to be carried out on each frame, as opposed to convolutional neural networks, such as [52]. In an ideal case, the reconstructed objects can be described once and simply tracked throughout the following frames.

To this end, SLAM proposes a pipeline consisting of motion tracking, motion segmentation, and 3D reconstruction to obtain the visual representation of individual objects from consecutive frames and map them into a 3D space.

While SLAM in robotics may utilize information from a wide variety of additional hardware sensors (such as depth cameras [81–82] and inertial measurement units [83–84]), visual SLAM has only the information provided by the motion tracking, which has to be, therefore, as precise as possible.

Major challenges arise when we deploy visual SLAM in dynamic environments. In the motion segmentation step, we need to account for the fact that not only the camera is moving through the environment but also that the individual objects are moving independently. As a result, neither the approach based on 8-point correspondency under epipolar geometry discussed in [85] nor the 5-point relative pose resolver [86] can be used directly. They may be utilized to derive the motion of the camera (ego-motion) only when we are confident that the segmented objects are static.

### ■ 4.5 Knowledge Extraction as a Tool

We believe that knowledge extraction should not be considered the final goal. Instead, we consider knowledge extraction a tool that allows better utilization of the data stored

in the media archives. The benefits of the extracted knowledge can be directly used by the professionals and with suitable representations even by the general public.

For example, professional content editors may utilize the extracted semantic information to organize their archives, search for material suitable for their projects, remove duplicities, or even protect their footage from unauthorized re-use.

For the consumers of the multimedia content, elaborate recommendation systems may take advantage of the information extracted directly from the clip instead of relying on the short text description attached to the video, movie ratings, and creation of user profiles.

### ■ 4.5.1 Multimedia (re)Discovery

One of the primary use cases for knowledge (feature) extraction and motivations for constructing an index on top of such features is to allow fast and reliable search in the vast amounts of data. Traditionally, we construct indexes on text data. Therefore, transitively, we commonly label the multimedia content with a free-text title and description and construct the index only on top of these text fields.

Without processing the actual media files themselves, the indexing algorithm of sharing platforms may utilize the additional information on the uploader of the content (which may be different from the actual author) and upload date.

However, searching only in an unstructured text description and a few metadata fields is inconvenient. It relies on providing an accurate description of the content by a set of words. Also, the query needs to be as similar as the description of the authors of the index. However, the authors have a more thorough knowledge of the content or context and may describe it only by a precise name, shot number, or other generally unknown technical properties. The description of the content is also subjective and is, therefore, prone to a possibility of different characterizations by other users or index creators.

#### **Structured Metadata Required on Upload**

One possible approach is to enable the uploader of the content to provide some structured information with a limited selection of values. For example, a location field with a value limited to one of the predefined classes, such as in the Places dataset [65]. While this annotation approach may be suitable for curated archives, many content creators would not bother to add such structured information by hand.

To mitigate the lack of human annotation, we may use feature extraction algorithms that take the content and classifiers that directly return some class labels. However, this approach is usually not as straightforward as we discussed in this chapter.

#### **Search in the Original Domain**

Another approach to a more convenient discovery in unannotated data is to search by an example. To this end, we need a feature extractor that can provide a (possibly high-dimensional) numerical description of both the objects in the archive and the object used as a query. However, we have to design such features to encode only the information that we want to base our search on and, at the same time, to be invariant to changes that we consider insignificant.

For example, photographs of a landmark may be captured from different angles and with different camera rotations, but we would like to index all photos of the same landmark very close to each other, such as evaluated in [87]. DJs commonly use short audio samples from other already published music, possibly slightly changed by pitch shifting or added sound effects, but still recognizable by applications such as Shazam [29].

However, the search by example presents an issue if the query example is not easy to obtain, as we may retain only some memory of the content but not possess an actual example in the same modality that we want to search in. For example, we may remember just the song's main melody but cannot recall the lyrics, so we cannot use text input. In such cases, we need to introduce some proxy representation of the indexed content in a different domain that we can use for index construction and the user can use for querying.

### Search in a Different Domain

We can also use a similar proxy content approach for search in visual content. Again, the most straightforward approach is to index and query in the same type of content, such as a sketch. To recognize symbols with a given order of individual strokes, we may use a simplistic approach of comparing the relative changes in the stroke distance and angle using methods such as Dynamic time warping [88]. However, such methods are not suitable for processing more complex user input, such as cursive handwriting and object sketches. To this end, we may use various neural networks, especially those that incorporate the temporal information, such as time-delay neural networks and recurrent neural networks, as discussed in [89] or [90].

The real-world application of the aforementioned approaches depends on extracting the proxy from the content stored in our archives. For music, we might be able to use melody extraction, such as [32], possibly combined with a method of separating some instruments of the mixed-down music track, such as in [34]. Some approaches, such as [91], aspire to extract vector descriptions or contours from static images. However, [92] points out that the artistic representation of the sketch may pose significant issues and provides some experiments on converting face photos into sketches to be matched.

Some of the sketch-based approaches then extend the search capabilities even to video. For example, the approach of [93] uses the sketches to define a visual appearance of the scene captured on video. As another possible use, the work of [94] illustrates the usage of input sketches as annotations of actions and motion during a sports event and allows interactive analysis of rugby matches top-down recordings.

For more thorough material on Example-based Search, please refer to Section 1.4 of our book on Classification methods for Internet Applications [C1].

## 4.5.2 Content Deduplication and Re-use Detection

Another use case that relies heavily on feature extraction is the archive deduplication and detection of content re-use. However, the significant difference is that the extracted information does not need to retain the semantic information required for interaction with a human trying to search for some content. Instead, the “content identifiers” should deal with a broader range of modifications that may result from the creative process or even attempts to circumvent copyright infringement detection systems.

The multimedia content deduplication process is not as easy as it may seem at first glance. Due to the sheer size of the raw data, the multimedia content has to undergo lossy compression before distribution or even storage. The original content may also be transcoded several times into newer formats that aim to preserve the quality at a significantly lower bit rate during the archival process. Therefore, we might end up with multiple versions of virtually the same content, yet vastly different data representation, which renders deduplication based on file hashing unusable.

Moreover, many versions of the content may be created during the creative process. During the recording, the archive collects not only several takes of the raw audio tracks of individual instruments or video clips but usually also several edited versions.

For example, even when processing individual audio tracks, we commonly trim down only some sections of the original recording, use denoising filters, add reverberation to introduce space to the parts recorded in low-echo studios, or distort the signal to get a completely different signature. Therefore, the direct comparison (even of the decoded signals) would provide mediocre results. By introducing a feature invariant to many such operations and a suitable similarity measurement, such as in [22], we can detect the duplicates at last. To scale up the detection of similar content for copyright protection purposes, services such as YouTube had to further improve the detection algorithm's efficacy in [30]. However, the basic building block of the suitable feature extraction persisted.

The video recording is usually subject to color corrections, cropping, trimming, motion stabilization, and other edits that do not usually change the overall layout of the video frame. However, coming up with a suitable approach to similarity measurement of video content is complicated. One of the possible theoretical approaches is to represent each video shot by a video signature constructed from a randomly sampled set of frames, such as in [95]. The major downfall of this approach is that the image features are based only on a simple 178-bin color histogram; the method discards all of the motion and shape properties. YouTube Content ID video protection probably uses a revised approach, but it is not publicly disclosed to the best of our knowledge.

### ■ 4.5.3 Recommender systems

The third major use case we will discuss in this chapter deals with the amount of available content in slightly different way. Instead of constructing an index to allow an efficient search or limiting content volume by discovering duplicates or close derivatives, the recommender systems aim to select items that might interest the current user at a given time.

Traditionally, most recommender systems use some variation of collaborative filtering, which collects data from the behavior of many users in connection with the considered items. This way, the recommendations are given to the current user based mainly on their past actions and actions of similar users.

In the online environment, we can easily collect ratings of the individual items, either as a numerical score or some mark of approval, and their association to user profiles. As proposed in [96], we can then construct a recommender system that assesses a similarity to the other platform users via Pearson correlation coefficient, predicts a rating of items unseen by the user, and returns the ones with the highest score.

To deal with the insufficient information on new users with an insufficient number of item ratings or rating overlaps with the other users, we may extend the recommender system to build a model of user preferences. Such models usually represent the user preferences as all possible ordered lists of all objects in the dataset, and then we resolve the similarities between the complete preference lists of different users by methods such as [97].

Another possible approach to mitigate a dependency on the intersecting user ratings is to deduce some similarity just on the user's interactions with the content. For example, the YouTube Video Recommendation System presented in [98] introduces a concept of related video content based on the pairs of videos co-watched in the same session by the same user and proposes a connection of them by association rule mining.

The most notable example of a dataset and recommender system based on the collaborative filtering of movie ratings is MovieLens [99]. The Netflix prize is an excellent

resource for novel approaches to recommendation systems design, such as a very successful matrix factorization approach [100].

However, for new content, we have to combine the recommender system with some approaches to content filtering to mitigate the issues connected to a cold start. One possible approach is to define some similarity measurement of the content based on the available metadata, such as the uploader of the content, its title, or description. However, as we discussed earlier, such metadata may not be available for all of the items. Therefore, according to the use cases before, we need to devise methods of extracting semantic features from multimedia content and their similarity assessment.

For more reading on Recommender Systems, please refer to Section 1.2 of our book on Classification methods for Internet Applications [C1]. Particularly for examples of machine learning methods used for collaborative filtering, please refer to Sections 3.2.2, 3.3.5, 3.5.5, 4.3.4, 5.1.6, 5.3.4, or 6.2.4.

#### ■ 4.5.4 Online Annotation

Another use case that we want to keep in mind is the knowledge extraction from multimedia content in real-time. Progress in computational power and accelerated hardware allowed this relatively novel area, and the applications extend from assistive technologies to security and autonomous vehicles.

In practice, the fundamentals of the knowledge extraction methods usually do not differ significantly from the traditional indexing methods. However, the implementation has to be highly optimized to enable processing of the incoming signal in real-time and with a minimum possible delay on limited hardware.



# Chapter 5

## Efficient Multimodal Classification

With the plethora of multimedia processing and knowledge extraction methods presented in the last chapter, we are presented with two difficult decisions:

- how to combine the outputs of individual classifiers executed on different modalities,
- and which processing methods should be used to efficiently describe a given media file.

To formalize our problem in an oversimplified manner, the ultimate task is to find a classifier  $\phi$ :

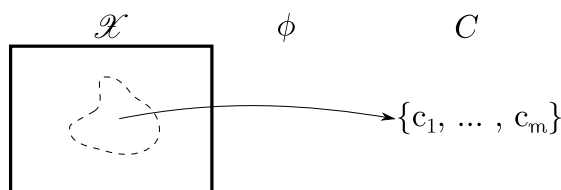
$$\mathcal{X} \rightarrow C = \{c_1, \dots, c_m\}$$

that creates a mapping from the feature space of the media files  $\mathcal{X}$  to one of the classes from a set  $C$  providing some information about a specific aspect of the media file in question.

### 5.1 Classifier Fusion and Meta-learning

As an easy example to follow, let us assume that we are presented with a short video of a smiling person with an audible laugh and words of joy, accompanied with a text description “Having a good time.” As a result, we are looking for the classification of sentiment as a selection from classes  $\{c_{positive}, c_{neutral}, c_{negative}\}$ .

#### 5.1.1 Monolithic Classifier



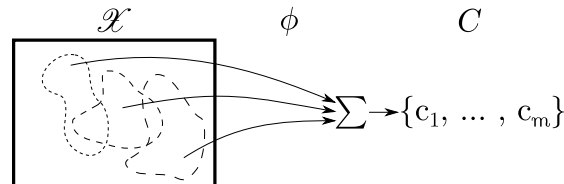
**Figure 5.1.** Schematic representation of feature-space coverage by a single classifier.

One of the theoretically possible yet technically challenging approaches is to design a monolithic end-to-end classifier that would take the whole multimedia stream, including the text description, and return a single resulting class as demonstrated in Figure 5.1.

Considering even our toy example, we soon conclude that this approach is not viable even with significant dimensionality reduction effort on all modalities. Even if we use just the 30 basic facial details described in [101], 17 low-level MPEG-7 Audio features, and one-hot-encoded occurrences of several tens of sentiment-bearing words, we would need a fully labeled dataset of hundreds to thousands of samples to achieve good generalization. Moreover, as the number of features is not proportional, and the features do not have the same type and semantics, we will have a complicated job of selecting a single appropriate classifier.

### 5.1.2 Classifier Fusion

One of the traditional approaches is to cover different parts of the feature space with different (and separately trained) classifiers and combine their outputs to provide the final class. Figure 5.2 represents a general case where the areas of the feature space that are well-covered by a given classifier may overlap; however, in the case of multi-modal classification, it is common to have a different classifier (or a set of classifiers) separated by the modality. The only limiting factor is that the class set of individual members has to be shared among all used classifiers.



**Figure 5.2.** Schematic representation of feature-space coverage by an ensemble of classifiers.

In the case of our toy example, we can use some pre-trained facial feature sentiment classifier, such as [102] that internally uses the distance between some of the facial characteristic points and returns one of seven classes:

$$\{c_{surprise}, c_{joy}, c_{sadness}, c_{anger}, c_{fear}, c_{disgust}, c_{neutral}\}.$$

These classes have to be then mapped to our three classes to participate in the ensemble.

A similar approach can be taken to assess the sentiment of the accompanying text. By pre-training a classifier described in [103] on a dataset such as the VADER Sentiment Lexicon [104] using tools from the Python Natural Language Toolkit, we can easily obtain a probability of positive description. For more information, please refer to Chapter 1.3 in our book [C1]. The positivity of the text can be transferred to the three target classes with a simple set of thresholds.

As another side-project, we studied in collaboration with Jiří Kožuszník the possibility of sentiment analysis from a recording of a human speech [A8]. We compared the performance of several classification methods on top of a combination of MPEG-7 features and some statistical features from Mel Frequency Cepstral Coefficients, Specific Loudness, and Sensation Coefficients. Our collaborative research concluded that Support Vector Machines and Multi-layered Perceptrons are the most suitable for such a given task. A full paper is provided as a part of the digital attachment.

Once we collect the votes from individual classifiers (responsible for their part of the domain), we can decide on the voting mechanism. The most commonly used approach is majority voting, where the class with the most votes is considered the final one. Modifications (including the use of class distribution instead of sharp votes) are extensively explained in the book by Ludmila Kuncheva [105].

### 5.1.3 Double Fusion

In addition to the traditional classifier fusion approach, we experimented with the approach of the double fusion framework proposed in [106]. This framework is based on the combination of pre- and post-classification data aggregation, called early and late fusion.

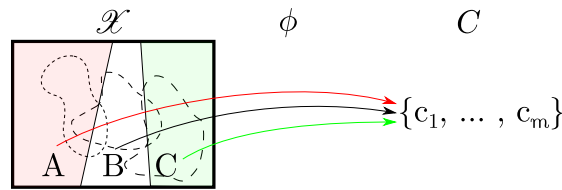
Early fusion merges the selected features to create a multi-modal data source for classification. During our experiments covered mainly in [A13], we have used this approach

to create a modality encapsulating textual data from OCR and speech recognition into a single synthetic feature that should, in theory, exploit the data overlap in these modalities.

To create such joined modality from the dataset described in Appendix A, we collected the text returned from the speech to text processing based on the at-the-time commercially available Google Speech API according to the use-case described in [107] and optical character recognition of text in recognized slides by Tesseract OCR [72]. To reduce the number of lousy word detections, we pre-filtered the set of words by matching them against the Aspell dictionary [108], extracting only the stems of individual words by a Snowball stemmer [109] from the NLTK library and removing some of the stop words that carry no semantic context. As described in Section 3.1.1, we then transposed the filtered text data into term-by-document matrices separately for each source and the concatenation of the text data from both sources and reduced the dimensionality by converting the sparse term-by-document matrices into dense bags of concepts as described in Section 3.1.2.

Late fusion is synonymous with the above-mentioned traditional classifier fusion, as this step aims to merge labels from individual classifiers. During our research in [A13], we use the majority voting to aggregate the results of the text and visual classifiers. The main reason for this decision is the ease of use and a simple analogy for a combination of classes based on posterior probabilities.

#### 5.1.4 Meta-learning



**Figure 5.3.** Schematic representation of feature-space division by meta-learning.

Instead of executing all classifiers simultaneously and combining their results, we may use a fundamentally different approach—to propose a recommender that will select the most suitable classification methods for the incoming data. Schematically we represent such feature-space coverage in Figure 5.3 with an additional division of the input space into additional areas  $A$ ,  $B$  and  $C$ . Upon the arrival of the new data point, we can check into which of these areas this point belongs and decide to use only the classification methods  $\phi_x$  (where  $x$  is from the set  $\{A, B, C\}$ ) that previously gave the best classification results or are otherwise most relevant.

This approach is traditionally used in the cases of a single modality where we have access to a wide range of data processing methods, yet usually with no guidelines or recommendations of which methods to use for a given problem. In such cases, the proposal of a suitable classifier for a given problem instance involves training many methods and selecting the most precise, the most robust, or the fastest one.

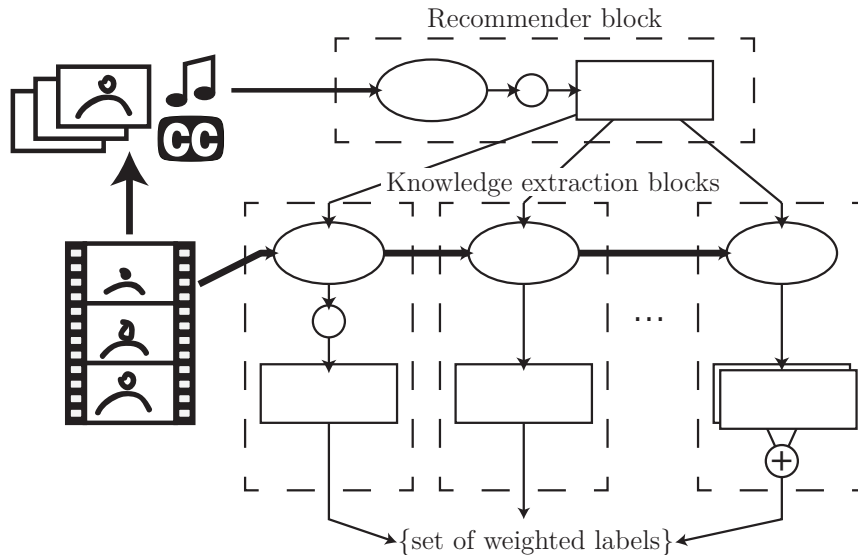
On the other hand, Meta-learning approaches can accumulate experience from applying the same processing methods over time and several processing tasks. A somewhat complicated selection of a processing method or its parameters may be reduced onto a search in the history of processing method executions. As this will consequently lead to “learning how to learn,” such process is called a meta-learning, and properties of input data upon which the meta-learning decide are called meta-features [110].

## 5.2 Meta-learning Framework for Multimedia Data

In a setting with multiple modalities, the meta-learning approach has an added benefit in the possibility of eliminating the processing methods that are not relevant for the given modality or its content altogether. For example, we should not execute classification methods on top of the subtitle track if no closed captioning is present in the multimedia file. On the level of the provided modality, it also makes no sense to find slide boundaries and transcribe text from the resulting images in other videos than lecture recordings. To confirm that the provided video is a lecture recording, we may utilize detection of a picture-in-picture video layout where one of the sources has a significantly lower apparent framerate. On the feature extraction level, it also does not make sense, for example, to search for and recognize faces if there are no people present in the considered shot. Although we would probably only waste some computational resources in this case, no misleading feature extraction should occur.

Such filtering of the processing methods is also crucial for the actual computation feasibility of the whole processing pipeline. However, as the traditional meta-learning calls for automatic selection of classification methods on the arrival of a new data point, all methods have to be executed anyway during the training, which is again time-consuming and wasteful.

To address this very issue, we proposed in [A10] a hierarchical meta-learning approach where the first layer of meta-learning classification will consider hand-picked features and simplistic classifiers (focused mainly on the processing speed) that are expected to have the biggest impact on the later selection of the best-performing classifiers in given part of the input data landscape.

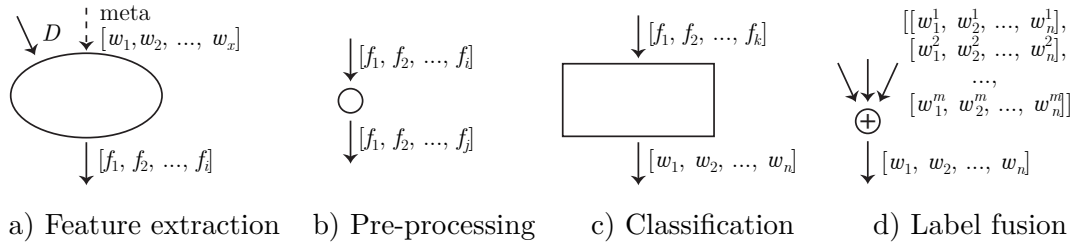


**Figure 5.4.** Block diagram of the hierarchical meta-learning framework.

The block diagram in Figure 5.4 presents the overall structure of our meta-learning framework and the data flow involved in the processing. We start with the original multimedia file (represented in the Figure by a film strip) from which we extract a set of individual frames, proxy video in lower resolution, cumulative image histogram, audio descriptors, closed captions, or other low-level features (depending on the task at hand). The main benefit of such proxy data is that such a limited amount can be easily stored locally inside a given multimedia platform (such as NARRA or Pan.do/ra) for

fast query, while the large original media files can be stored even offline and queried only when necessary. The proxy data are then handed over to the recommender block and used solely to propose a set of knowledge extraction blocks and possibly their hyper-parameters if needed.

The selected knowledge extraction blocks then correspond to the stand-alone knowledge extraction methods, such as we discussed in Chapter 4, with their feature extraction and pre-processing methods, classifiers, and label fusion. The knowledge extraction blocks have full access to the original multimedia file.



**Figure 5.5.** Building blocks of the proposed meta-learning framework.

Figure 5.5 provides the breakdown of the individual building blocks. Within the recommender block, the feature extraction takes some proxy of the incoming data or the data itself ( $D$ ) and produces a high-dimensional feature vector ( $[f_1, f_2, \dots, f_i]$ ). To provide an oversimplified example of meta-learning in scene classification, we compute a histogram of each incoming image. Real-world indoor/outdoor classifiers, such as [111], consider the textural features as well.

In many cases, the extracted features are too verbose for the classifier, and we need to reduce the dimensionality even further (from  $i$  to  $j$ ). In our example, we can average the histogram over time and increase the histogram bin size to gain just the basic color tonality of the shot.

The classifier then takes the set of features ( $[f_1, f_2, \dots, f_j]$ ) and returns class weights ( $[w_1, w_2, \dots, w_n]$ ). The classifier in the recommender block can be pre-trained to discriminate between indoor and outdoor shots to follow our example.

Because a set of classifiers with the same pre-processing but different model parameters may be executed on the same features (or their overlapping subsets), we may execute the processing more efficiently by running the classifiers in parallel. Later, we can aggregate the class weights with some label fusion which takes a class weight matrix and aggregates it into a single class weight vector.

Once we obtain the information from the recommender block, we can choose among the set of prepared knowledge-extraction blocks and pass the meta-information to its feature extractor. In the example of scene classification, the indoor class may invoke a room type classifier that utilizes visual information on furniture placed in the considered room [112]. Or start the relatively expensive process of visual simultaneous localization and mapping (see 4.4.5) for better object recognition [113] as the indoor environment is more suitable for the task of camera visual localization.

If the outdoor class is detected, we can start another range of processing methods specific to the outdoor setup. Such as determining color constancy, subtraction of background independently of illumination, determining scene geometry from the illumination changes and casted shadows, or even geo-locating the camera as models such as [114] provide access to the angular velocity of the sun and angle of the casted shadow.

The methods of scene classification that are pre-trained on datasets such as Places [65] can classify both indoor and outdoor scenes. Therefore, we can execute them

regardless of the indoor/outdoor classifier result, but such additional information can be used inside the knowledge extraction block to fine-tune the classification process or reduce misclassifications.

To address one of the edge cases, if we have access to the unmodified metadata of the original recorded content and they contain the GPS coordinates, we can directly use the location as an additional source of truth. We may still need to classify if the shot was made inside or outside if the location points to an urban area. We also have to keep in mind that the location is recorded just once at the start of the recording. However, in many cases, metadata processing may simplify scene classification tremendously.

The framework will be executed repeatedly in practice, as different aspects of the multimedia content and the selection of processing methods may rely on the already extracted knowledge. However, as this external classifier hierarchy is outside the meta-learning scope, we do not include it in our diagram.

### 5.3 Search for Structure in Multimedia Data

In collaboration with Michal Kopp, we proposed in [A12] a use of self-organizing maps combined with hierarchical clustering to discover structure in a multimedia dataset. The discovery of an inherent structure in complex high-dimensional data is of our particular interest, as the structure itself may be utilized for a direct projection of an otherwise complex similarity function into a low-dimensional space. As a second aspect, we can leverage the discovery of such structure in search for classification and other knowledge extraction methods that can be used to distinguish diverse types of the multimedia material (corresponding to the knowledge extraction recommender in the meta-learning framework presented above), or that are specific for a given cluster (leading to the definition of the expert knowledge extraction blocks).

The traditional methods of **hierarchical clustering** are a great option to discover a structure in relatively low-dimensional data. However, albeit significant dimensionality reduction by using bag-of-concepts and other methods, our multi-modal dataset described in Appendix A contained 5 800 numerical features after conversion of each presentation slide and accompanying video and audio into a single vector. Computing distances between all pairs of almost 16 000 data points is not feasible.

A different method of data structure discovery is based on the use of **self-organizing maps** (SOMs) [115]. The main idea behind SOM is that it can preserve topological relations of the input space, which is typically high-dimensional, in a low-dimensional map (typically two-dimensional). That means the data that are similar in their original high-dimensional input space are also close to each other in the resulting map. Due to this characteristic, the relationships of high-dimensional data points are much simpler to visualize.

SOM is constructed as a neural network with a single layer of neurons organized in a regular topology. Most common is the grid, hexagonal and random topology. Also, a distance function, which measures some distance between the neurons, needs to be selected. Common distance functions are Euclidean distance, Manhattan distance, or the length of the shortest path between neurons.

Before the training, we assign each neuron in the map with a weight vector of the same dimension as the input vectors. We may either use samples from the input data or small random values to initialize the vectors.

Training the SOM is an iterative process. In each iteration, one sample vector  $X$  is presented to the map, and similarity between the selected feature vector and all weight

vectors in the map is calculated, in our case, with a Euclidean distance. The neuron that has the most similar weight vector is selected as a winning neuron  $c$ . More recent versions of self-organizing maps may be presented with batches on input to speed up the process, as described in [116].

When the winning neuron is found, the weight vectors of the winning neuron and its neighbors are updated such that the neighborhood is attracted to the presented input vector with decreasing magnitude of changes according to the growing distance from the winning neuron. The update rule for a neuron  $i$  with weight vector  $W_i(t)$  is:

$$W_i(t+1) = W_i(t) + \alpha(t)\theta(c, i, t)(X(t) - W_i(t)),$$

where  $\alpha(t)$  is learning rate from interval  $(0, 1)$  and is decreasing with increasing iterations, and  $\theta(c, i, t)$  is the neighborhood function which depends on the distance between the winning neuron  $c$  and the considered neuron  $i$  (and which may also depend on the iteration counter  $t$ ).

As a prominent example of using SOM on multimedia data, PocketSOM [117] uses self-organizing maps for mapping favorite songs for a given user. Internally, the songs are characterized by acoustic attributes – a high dimensional vector of features representing the original song. The primary purpose of the PocketSOM is to assist with building custom playlists by providing the song similarity in a simple visualization. As a result, the user can choose the songs by drawing a path on the organized map. The songs mapped to the nodes (neurons) on the path are then returned to the user to decide which songs will be added to the final playlist. Even though this approach uses relatively high-dimensional data, the number of items is limited. The performance of the self-organizing map is, therefore, not hindered.

Another example is the Music Map [118], which utilizes SOM for the music playlist generation from a much more extensive collection of songs. This system takes the list of songs that should be included in the playlist on the input, transforms it to an optimization criterion for the path planning, and traverses the pre-generated map along an optimal path to generate a new playlist. Although the user is not involved in the Music Map playlist generation as much as in the PocketSOM playlist generation, the main idea remains the same – to provide the user with a possibility to create a playlist that corresponds to their demands. However, in contrast to our requirements for the structure discovery in multimedia, the Music Map uses only nine features, whereas our dataset contains thousands of features.

To accommodate the high dimensionality of our dataset and retain the benefits of hierarchical clustering in data discovery with no prior estimate on cluster count, Michal Kopp proposed a clustering algorithm that computes the average within-cluster distance from the trained SOM.

To this end, we first train a relatively small self-organizing map on the full dataset. Due to the number of samples and their dimensionality, the largest map in our experiments has  $20 \times 20$  neurons connected in a hexagon lattice, but we also experimented with the sizes of  $8 \times 8$ ,  $12 \times 12$  and  $16 \times 16$ .

The divisive hierarchical clustering algorithm then uses the Euclidean distance between the target nodes in the map to assess the distance between the samples, thus effectively limiting the dimensionality on which the hierarchical clustering is made to just two. To measure the quality of each clustering solution obtained by cutting the cluster hierarchy, we calculate the average within-cluster distance by mapping each pair of input vectors from each cluster to the neurons in the map and considering the dis-

tance of the neurons instead of the original data points, similarly to the cluster splitting step.

To provide an illustration of the self-organizing map and the quality of the hierarchical clustering approach that uses the distance measurement from the pre-computed SOM on our dataset, we select the smallest  $8 \times 8$  map and present the results in Figure 5.6.

From the distribution of feature vectors shown in Figure 5.6a) we can see that from the total number of 15 953 data points, there are 1 854 mapped to a single neuron. That indicates that there are many quite similar entries in the data. This is even more evident when we look at the neighborhood of the neuron, which is also populated with quite a lot of data points, opposed to the lower-left part of the map.

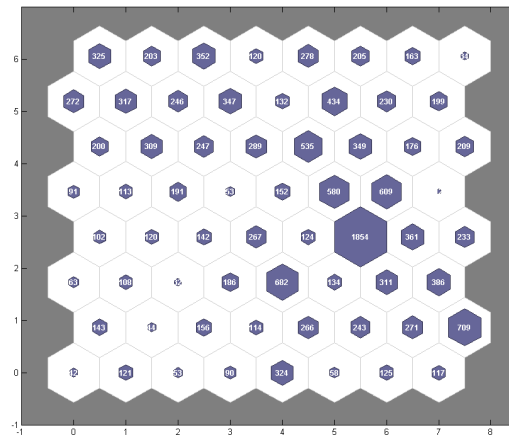
Upon closer inspection of the data points that are mapped to these neurons, we found out that these sections of the recorded lectures provide limited information in the extracted features. Namely, nearly half of input data mapped to this neuron seems to be slides created from video playback, or they are just images without text. So, even though these slides look different visually, the OCR produces little to no text, resulting in greater similarity among them. Many of the input data that populated these neurons also had little to no recognized speech.

A quite exciting property of the acquired map is that even though we manually found whole sequences of similar input data from one lecture that should be in theory mapped to a single neuron, we found many of the input samples in the neighboring nodes of the map. This leads us to the idea that large clusters (with significantly more samples) tend to spread over more neighboring neurons, but this notion is yet to be formally proved.

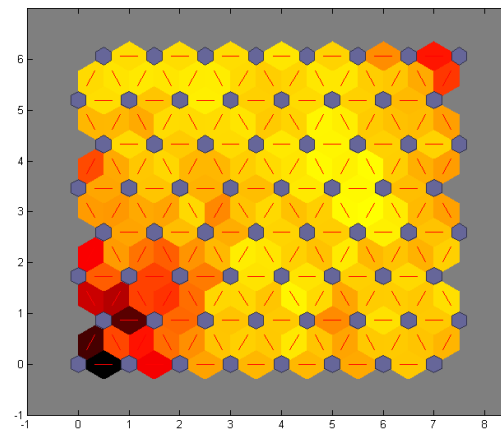
Figure 5.6b) shows the distances between neighboring neurons. We observe that the shortest distances are between the neurons with the highest number of vectors. Conversely, neurons in the lower-left corner are the most distant from each other. When we look at the input data, we find that the segments of video recordings and associated slides projected during that segment differ substantially from the remaining segments of all recorded lectures. First, the camcorder was set to shoot only the projected slide during the segment in question instead of the whole lecture hall. Therefore, the resulting set of visual features and histogram was significantly different. Second, in most cases, the projected slides were web pages, including the browser's interface. Moreover, in 35 out of 42 segments mapped to the neuron in the bottom-left corner, the captured web page was taken from Wikipedia, so they contained a significant volume of text instead of only a few lines of text we encounter on a typical slide. Therefore, the text documents transcribed from the captured images were also significantly different.

Figure 5.6c) shows that in the map of size  $8 \times 8$  neurons, an average within-cluster distance drops to its local minimum when the divisive hierarchical clustering is stopped at about a hundred clusters. This means that the optimal number of clusters is higher than the current number of neurons in the map – 64. Therefore we continued with our experiments on larger self-organizing maps, up to a dimension of  $20 \times 20$  where we concluded that the clustering to 400 – 500 clusters finally provides the lowest average within-cluster distance (similar to the size of the self-organizing map) and that it does not make any sense to make the maps any larger. Moreover, we have to keep in mind that the explainability of large maps is much harder, as the populated clusters spread over larger neighborhoods and individual neurons get much smaller support.

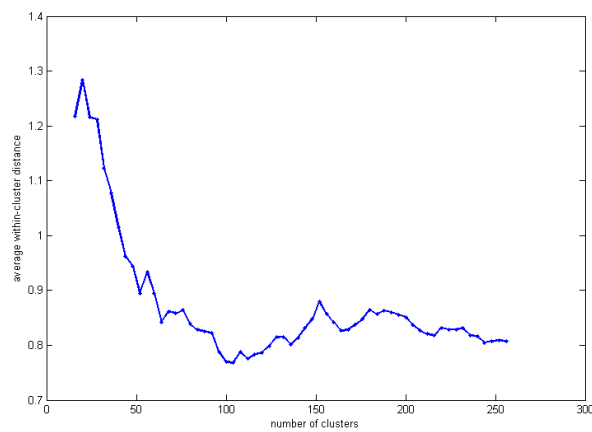




a) Relative distribution of the input feature vectors with the total numbers of vectors mapped to each neuron.



b) Distances between neighboring nodes in the hexagonal lattice, darker color means the greater distance between the weight vectors.



c) A within-cluster distance, measured as the length of the shortest path between neurons, is dependent on the number of clusters produced from the hierarchical clustering.

**Figure 5.6.** Results of our application of an  $8 \times 8$  hexagonal self-organizing map and hierarchical clustering for structure discovery in our dataset.

## 5.4 Rule Extraction from Multimedia Data

The comprehensibility of a classification algorithm (or any other data processing method) is crucial in many areas as it provides a deeper insight into the employed algorithms or final decisions and allows direct tuning of the system by a human observer if needed. Therefore, we dedicated a whole chapter to the classifier comprehensibility in our book on Classification Methods for Internet Applications [C1].

Most of the approaches revolve around rule extraction, as the communication of a decision by some formal logic seems to be the most universally accepted and simultaneously is relatively easy to modify. Because the condition is commonly stored as a conjunction of threshold checks covering some part of the input data space, and the consequent is simply the mapping to one of the classes, we can hand-pick only the rules that have good support or are backed by some other evidence or external knowledge. The possibility to modify the individual thresholds then provides an additional possibility of fine-tuning the rules by the domain expert.

Induction of **decision trees**, such as described in [119–121], is one of the machine learning strategies that are suitable for the direct creation of such sets of rules. We can leverage that the decision tree is constructed by recursive partitioning of the feature space such that the samples with the same labels stay together, and this decision is made on a single dimension at a time. As such, we can directly take all paths from the root of the tree to the leaves, collect all the splits with their respective thresholds as a set of conditions and the class or value in the leaf as the consequent.

However, learning of a decision tree is not feasible on high-dimensional problems, such as we have on our hands. Therefore, we need to take a slight detour and use other machine learning methods that are not as sensitive to the problem’s dimensionality.

In collaboration with Matěj Fanta, we provide in [A3] a survey of methods proposed for **extraction of rules from artificial neural networks** that are much more suitable for the processing of high-dimensional data, such as our dataset described in Appendix A. However, their universal approximation property has always been hindered by the very low human comprehensibility of the purely numerical representation that neural networks use to describe relationships and dependencies.

TODO

Since the 1980s, many rule extraction methods from neural networks have been developed; a good overview can be obtained from the survey papers [122–125] and the monograph [126]. Due to that diversity and a close connection of the semantics of extracted rules to the respective application domain, no rules extraction methods have ever become a standard, and it is always necessary to select a suitable method for the considered domain.

The main factor in the comparison of the rule extraction methods is the quality of the extracted rules. The survey paper [123] introduced four quality measures from which we will use:

$$\text{accuracy: } Acc = \frac{\#\{C(x) == C_R(x)\}}{\#X}, \text{ and fidelity: } Fid = \frac{\#\{C_{NN}(x) == C_R(x)\}}{\#X},$$

where  $C(x)$  denotes the ground truth class of the data  $x$ , the result of classifying  $x$  by a neural network is denoted as  $C_{NN}(x)$ , the classification by the extracted rules as  $C_R(x)$ , and  $\#X$  denotes the number of input samples.

The last quality measure is comprehensibility, which is task-specific and may contain the number of the extracted rules, the number of antecedents per rule, and other properties of the result.

For our experiments we selected three rule extraction methods: artificial neural-network decision tree (ANN-DT) [127], DeepRED [127], and HypInv [128].

The **ANN-DT** is one of the simplest methods taking into account only the input-output mapping learned by the network. It builds a regular decision tree with axis-parallel splits, but it obtains the input-output pairs for its training by generating the output by the neural network in question instead of using just the ground truth.

Nowadays, the most promising decompositional method seems to be **DeepRED** that can be applied to multilayer perceptrons of any depth. It was first published in [127], but the master thesis of Jan Zilke [129] describes it in more detail. The core idea of the algorithm is to build decision trees on the activations of the considered neural network as the input and compose these trees, or rules derived from them, into more complex ones.

The training of a neural network can also be considered a definition of subspaces of the input space belonging to the same class. A classical decision tree then approximates the boundaries between these areas by hyperplanes parallel to axes. As an alternative **HypInv** takes general hyper-planes for the boundary approximation. For this purpose, the derivative of the mapping computed by the neural network is used.

We applied these rule extraction methods to neural networks trained on the text modalities of a multimedia dataset:  $n$ -grams (with  $n$  set to 1 – 3 and concatenated on feature level) from text recognized in the slide images by OCR and bag of words from speech transcribed with ASR system, both of them with different subsets of features.

To be more precise, the neural network architecture for classification of text extracted from audio and converted to a tf-idf-weighted bag of word stems with no additional feature selection was experimentally set to 80 and 20 fully connected neurons in the hidden layers. We refer to this network as `audio_stem_all`.

To reduce the dimensionality but retain the explainability of retained features, we trained an ANN-DT on the individual stems from the audio transcript and used the internal metrics of the decision tree to select the features that contribute the most to the classification. We chose 1 000 of the attributes that provide the split with the highest entropy decrease on one of the selected networks. This network will be denoted as `audio_stem_fsen_1000`.

As we trained the decision tree for both the audio stems and OCR  $n$ -grams, we also experimentally selected the subset of features that appeared in any of the decisions at least once. This reduction leads to dimensionality under 1 500, and the resulting networks will be denoted by `audio_stem_fsdt_0` and `ocr_fsdt_0`.

After the four networks were trained concerning the precision of the resulting classifier, we subjected them to rule extraction methods with various settings with results presented in Table 5.1.

Experiments have shown that the rule extraction methods differ significantly in their performance on networks trained with different data. In addition, for methods that inherently use decision trees, i.e., ANN-DT and DeepRED, it is crucial to have data on which the decision tree itself achieves satisfactory results.

The most consistent performance was observed for the HypInv method. However, as the extracted rule is in the form of a linear combination of many features, its expressive power is generally not as good as that of other studied methods. On the other side of the spectrum, DeepRED results were the least consistent. We attribute this failure

**Table 5.1.** Comparison of the implemented methods on the binarized datasets. Averages and standard deviations are computed by 10-fold cross-validation. Cells in the column “rule\_count” where the number of rules is marked with \* are intervals in which the value occurs. Columns denote fidelity, validation accuracy, validation fidelity, the number of rules, the number of attributes used by the whole set of rules, the mean rule length, and the mean number of attributes occurring in each rule.

Network	Method	Settings	fidelity (%)	val_fidelity (%)	val_accuracy (%)	rule_count	index_count	rule_len	rule_index
audio_stem fsdt_0	ANN-DT	absvar-var	53.18 ± 0.42	53.14 ± 1.93	50.11 ± 2.09	14.9 ± 27	12.9 ± 25	2.06 ± 3.43	2.06 ± 3.4
		entropy	57.23 ± 1.38	56.78 ± 2.13	54.93 ± 1.59	10.6 ± 6.3	9.40 ± 6.2	4.05 ± 1.72	4.01 ± 1.7
		fidgain	58.31 ± 1.31	57.07 ± 2.96	54.33 ± 2.76	26.5 ± 4.7	22.8 ± 4.7	5.80 ± 0.36	5.76 ± 0.39
		var	57.01 ± 1.26	56.37 ± 2.14	54.71 ± 2.02	11.0 ± 4.2	9.40 ± 3.6	3.94 ± 1.10	3.94 ± 1.1
	DeepRED	build_first	54.18 ± 1.87	53.25 ± 3.58	52.55 ± 3.03	2 - 503*	7.80 ± 6.1	4.90 ± 3.61	4.17 ± 2.7
		del_last & build_first	52.86 ± 2.62	53.34 ± 3.48	53.37 ± 3.03	57 - 6060*	22.7 ± 8.1	12.01 ± 2.87	10.7 ± 2.4
HypInV	20-6	80.39 ± 1.86	80.76 ± 3.00	77.07 ± 2.53	7.30 ± 3.6	1533 ± 1.9	3.44 ± 0.95	1533 ± 1.9	
audio_stem fsen_1000	ANN-DT	absvar-var	52.01 ± 0.59	51.92 ± 1.88	50.11 ± 1.70	17.3 ± 22	15.3 ± 21	2.99 ± 3.86	2.98 ± 3.9
		entropy	58.55 ± 1.30	57.93 ± 1.77	55.25 ± 1.77	6.50 ± 2.4	5.40 ± 2.3	3.06 ± 0.62	3.06 ± 0.62
		fidgain	56.02 ± 1.23	54.78 ± 2.04	53.05 ± 1.57	24.7 ± 7.6	21.8 ± 6.2	5.82 ± 0.54	5.72 ± 0.52
		var	58.83 ± 1.10	58.12 ± 1.78	55.41 ± 1.96	11.3 ± 5.7	10.1 ± 5.5	4.49 ± 1.68	4.49 ± 1.7
	DeepRED	build_first	52.05 ± 3.56	51.49 ± 3.16	52.38 ± 2.04	9 - 1361*	14.7 ± 8.0	8.50 ± 4.24	7.38 ± 3.4
		del_last & build_first	49.78 ± 1.65	49.93 ± 1.14	51.76 ± 1.26	2 - 13278*	23.9 ± 11	12.44 ± 5.00	10.6 ± 4.2
HypInV	20-6	77.74 ± 2.54	77.76 ± 2.61	73.14 ± 1.82	11.6 ± 6.5	999 ± 0.3	3.90 ± 1.22	999 ± 0.32	
audio_stem all	ANN-DT	absvar-var	51.15 ± 0.93	50.87 ± 2.70	49.71 ± 2.16	2.20 ± 2.7	1.20 ± 2.7	0.74 ± 1.60	0.74 ± 1.6
		entropy	59.78 ± 0.90	59.59 ± 2.63	55.84 ± 2.07	6.60 ± 2.9	5.40 ± 2.8	3.37 ± 1.28	3.33 ± 1.3
		fidgain	58.42 ± 2.90	56.63 ± 3.64	55.73 ± 3.97	27.5 ± 11	24.2 ± 9.7	5.75 ± 0.92	5.71 ± 0.87
		var	59.82 ± 0.88	59.69 ± 2.72	56.01 ± 2.21	8.10 ± 5.6	7.00 ± 5.3	3.45 ± 1.51	3.45 ± 1.5
	DeepRED	build_first	51.54 ± 1.95	52.16 ± 2.68	51.93 ± 3.13	3 - 75*	7.60 ± 3.1	4.95 ± 1.81	4.37 ± 1.5
		del_last & build_first	51.89 ± 2.01	51.66 ± 3.06	51.17 ± 2.48	3 - 1019*	11.8 ± 9.9	6.67 ± 4.23	5.95 ± 3.8
HypInV	20-6	88.28 ± 5.45	87.17 ± 5.99	83.37 ± 3.84	3.00 ± 2.1	24437 ± 161	1.57 ± 1.20	24437 ± 161	
ocr fsdt_0	ANN-DT	absvar-var	59.94 ± 4.12	59.20 ± 4.65	54.32 ± 4.43	46.2 ± 30	42.0 ± 27	6.45 ± 2.56	6.38 ± 2.5
		entropy	81.17 ± 2.02	78.02 ± 1.12	70.88 ± 2.61	105 ± 26	86.4 ± 19	8.31 ± 0.35	8.17 ± 0.41
		fidgain	79.75 ± 3.01	75.74 ± 1.84	70.38 ± 5.77	118 ± 48	97.4 ± 37	7.63 ± 1.17	7.50 ± 1.1
		var	70.65 ± 6.96	69.68 ± 6.56	66.52 ± 4.88	49.7 ± 25	45.9 ± 24	6.93 ± 2.29	6.91 ± 2.3
	DeepRED	build_first	62.36 ± 7.09	62.03 ± 6.32	60.18 ± 3.95	108 - 50763576*	37.7 ± 11	25.55 ± 8.94	18.0 ± 5.4
		del_last & build_first	65.70 ± 5.08	65.65 ± 5.63	61.62 ± 4.91	82 - 3324312*	34.2 ± 16	17.80 ± 6.88	13.2 ± 4.1
HypInV	20-6	73.14 ± 3.11	72.91 ± 3.68	69.09 ± 7.35	19.8 ± 5.2	711 ± 213	4.87 ± 0.22	711 ± 213	

to the fact that it internally generates decision-directed acyclic graphs with random depths and, therefore, many of the extracted rules are excessively long.

## 5.5 Steps Towards Efficient Multimedia Classification

All our following research stems from the theoretical principles provided in this chapter, and that can be summarized with the following set of recommendations:

- Try to extract basic knowledge with computationally inexpensive approaches first.
- Make use of the local proxy data in these essential knowledge extractors, but be aware of the limitations of such data.
- Propose the execution of consequent knowledge extraction methods iteratively using the already available knowledge.
- Exploit the temporal properties of multimedia content and prefer simple methods for knowledge transfer from one frame to the other while executing the expensive single-frame extraction methods only when necessary.
- If possible, attempt the knowledge extraction from the most accessible modality first. Resort to extraction from other modalities if uncertain, and properly align and correlate the extracted data.
- To allow efficient modifications to the knowledge extraction pipeline, prefer explainable methods and search for inherent structure in the extracted data before executing any additional processing.

In the following two chapters, we will summarize our research in the area of scene detection, extraction of optical flow, and multiple object tracking that follow these recommendations.

## Chapter 6

# Video Scene Recognition

During the design of the meta-learning framework, we mainly relied on already developed knowledge extraction methods. However, many such approaches were originally designed to process individual frames with rather computation-heavy methods and provide the final class-based on majority voting. The processing of individual frames is also far from efficient due to the redundancy in the video signal.

Recognition of a scene from a video shot was one of the first areas that we took particular interest in, as the extraction of such knowledge can provide a lot of valuable information for selecting consequent processing methods. Such as we mentioned earlier, it does not make much sense to execute some of the other expensive knowledge extraction methods in the context of some scenes, for example, the estimation of camera location from the landscape [130] if the video clip is classified as captured in a living room.

To further limit our scope, we consider the generic scene classification into a small set of generic classes too generic for our approach. In personal archives and video repositories of media houses, we are much more interested in the problem of scene recognition – describing in which exact scene the video clip was taken. As an oversimplified example, we want to extract the information if the shot was taken in my living room or a living room of a friend, not just the information that it was taken in any living room.

This requirement corresponds with the ultimate goal of (shooting) script reconstruction from a video clip, where the shot location is specified to the smallest of details, opposed to a screenplay where only the type of environment is specified.

At the same time, we want to take into account the limited annotation of many personal archives, which poses a significant issue for the traditional machine learning methods.

To this end, we prepared a custom dataset from a popular television series, *The Big Bang Theory*, of which we annotated the first series by hand with a hierarchical descriptor of the scene, such that we captured the information on the building, room, part of the room and closer specification of the shot. The hierarchical descriptor has a significant benefit in that it can be truncated to a certain level if we need better support for some of the classes and kept longer in situations where the more detailed description of the shot is better suited for the description of the visual appearance.

For the given shot, we also provide in the dataset a single representative frame and a set of differently binned color histograms that are accumulated over the run of the clip. More detailed description of the dataset is provided in Appendix B. The dataset itself is a part of the digital attachment.

## 6.1 Video Scene Classification from Statistical Features

One of the approaches to scene classification that we used with Tomáš Šabata in [A6] was to train a simplistic classifier of (potentially partially spatially aware) histograms to a pre-selected level of the provided scene labels.

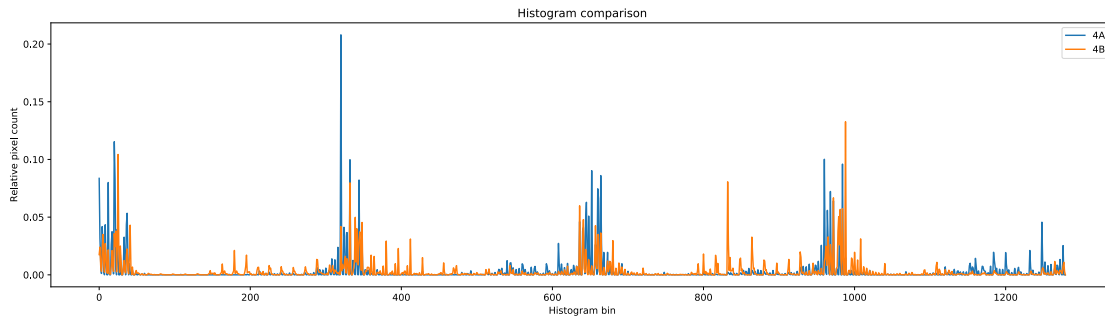
The motivation for using the color histogram is two-fold. First, the color histogram is a straightforward data structure that can be collected online during the playback of the video clip and requires a minimal amount of both memory and processing power. Moreover, collecting the histogram over the run of the whole clip then eliminates the effect of moving foreground objects and camera motion, as the histogram averages all such visual changes. The other motivation stems from an assumption that the color tonality of the particular scene can be directly used for its description if a suitable colorspace is used. We demonstrate the basis for such assumption in Figure 6.1.



a) Room 4A



b) Room 4B

c) Comparison of HSV histogram on  $2 \times 2$  grid.

**Figure 6.1.** Representative frames from two distinct living rooms and comparison of the proposed histograms. Although both of these pictures depict a living room, the distribution of colours is different. Source images courtesy of CBS Entertainment.

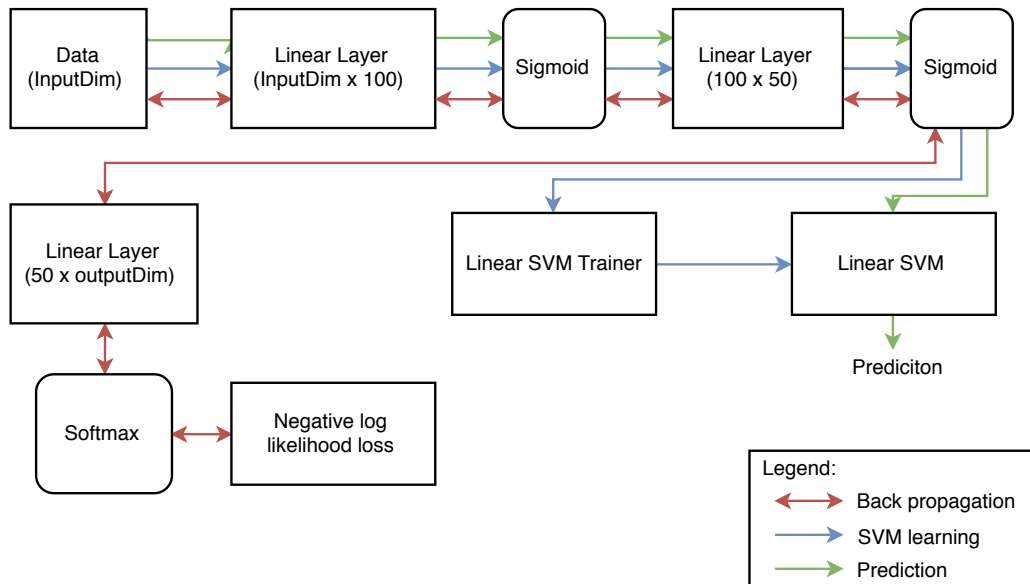
Concerning the selection of colorspace, RGB (red, green, and blue components) is the primary color space of multimedia acquisition and processing. However, it does not directly encode the quality of the color perceived by a human. To this end, we consider using the HSV encoding (hue, saturation, and grayscale value, i.e., the overall lightness of the color). In compliance with the state-of-the-art methods such as [131–132], we also sample the hue with higher precision (narrower bins in histogram approaches) than the saturation and value.

We then choose a shallow feed-forward neural network as the base scene classifier from the histograms in question. In particular, we use a network with two hidden layers of 100 and 50 neurons and logistic sigmoid as activation function. The output layer uses the softmax activation function. The network is trained using backpropagation with a

negative log-likelihood loss function and a stochastic gradient descent optimizer. The network topology, activation function, and optimizer were selected through a simple grid search. We also considered other activation functions such as ReLU or hyperbolic tangent and different optimizers based on adaptive estimates of first and second moments of the gradients, as discussed in [133].

We then use the trained neural network for the scene classification task. However, we introduce an improvement inspired by transfer learning. Transfer learning is usually used in deep convolution neural nets where the convergence of all parameters is slower [134]. However, we would like to demonstrate that transfer learning can bring a substantial benefit also in shallow neural networks, especially in combination with a support vector machine (SVM) classifier.

In our scenario, we freeze the parameters of the first layers and use the network as a feature extractor. The original softmax layer is then replaced with a linear support vector machine for the classification stage. This modification brings us a relatively small but consistent improvement in the final accuracy. For an overall structure of our proposed network, please refer to Figure 6.2.



**Figure 6.2.** The architecture of the proposed neural net. Red arrows represent the first learning phase in which parameters of the net are found using a backpropagation. Blue arrows represent the second phase – transfer learning. Here, the first two layers of the already trained neural net are used for training dataset generation. After that, a linear SVM classifier is trained. Green arrows represent the prediction of new samples.

Finally, the model performance was improved by using a combination of semi-supervised and adaptive learning by choosing a combination of uncertainty sampling with pseudo-labeling through self-training.

In each iteration,  $n$  samples with the lowest utility function were queried to be annotated. At the same time, samples with the utility function higher than a threshold were predicted using the current version of the model, and these predictions were then used to train the next version of the model. Utility functions were calculated from the output of the softmax layer of the neural net. The number of samples  $n$  was chosen

to be 5 in each iteration. The threshold value was tuned to keep the number of wrong labels getting into training data as low as possible.

To evaluate our approach, we compared the accuracy of our system with the traditional classifiers such as linear Support Vector Machines (Linear SVM),  $k$ -nearest neighbor classifier ( $k$ -NN), naïve Bayes (NB), and the feed-forward neural network without the linear SVM layer (FNN) on several collected histogram types.

As a baseline, we computed a flattened cross-channel histogram in the RGB color spectrum, reduced to 8 bins per channel, thus resulting in 512 features (RGB 8x8x8). Then we experimented with a histogram of just the hue, reduced to 180 bins (H), and a concatenated global histogram of hue, saturation, and value (HSV). To incorporate some spatial information into the feature vector, we introduced a flattened cross-channel histogram of 20 bins in hue times four bins in saturation, and four bins in value over the video sequence divided spatially in half horizontally and vertically. We refer to the resulting data as HSV 20x4x4 2\*2.

Comparison of the accuracy across the extracted data and selected classifier is presented in Table 6.1. It is noticeable that HSV 20x4x4 2\*2 feature dominates over all other variants. Therefore, we were using HSV 20x4x4 2\*2 in the subsequent experiments. On the same note, replacing the last layer of the FNN with an SVM classifier brings a relatively small but consistent improvement across all extracted histograms.

**Table 6.1.** Accuracy of combining the considered four kinds of histograms with selected classifiers. The highest accuracy for the given histogram is in italics. The highest accuracy for the given classifier is in bold.

Accuracy [%]	Linear SVM	$k$ -NN	NB	FNN	FNN+SVM
RGB 8x8x8	18.1	32.42	26.1	54.5	<b>60.0</b>
H	12.4	30.7	26.1	56.0	<b>58.9</b>
HSV	14.1	32.6	32.4	63.3	<b>65.7</b>
HSV 20x4x4 2*2	<i>46.0</i>	<i>45.4</i>	<i>33.9</i>	<i>77.2</i>	<b>78.8</b>

To further improve the classification accuracy, we propose the addition of semi-supervised (SL) and active learning (AL) to the system. The initial dataset contained only 5 315 labeled samples. In addition to that, we extracted an additional unlabeled dataset with 26 528 samples and used it for both active and semi-supervised learning. A human annotator was asked only five queries at each of ten iterations. The result of our experiment is presented in Table 6.2. Even though the improvement is rather small, we successfully demonstrated the viability of such improvements. We predict even better results with a more sophisticated combination of such approaches.

**Table 6.2.** Final achieved accuracy, weighted accuracy, precision, recall, and F1 score with the HSV 20x4x4 2\*2 histogram. The highest value among the considered classifiers is bold for each classifier performance measure.

	Acc	Weighted acc	Precision	Recall	F1
FNN	0.7723	0.8518	0.7813	0.7590	0.7578
FNN-SVM	0.7883	<b>0.8626</b>	0.8026	0.7837	0.7842
FNN-SVM with SL+AL	<b>0.7895</b>	0.8617	<b>0.8037</b>	<b>0.8022</b>	<b>0.7978</b>

## 6.2 Video Scene Classification with Neural Networks

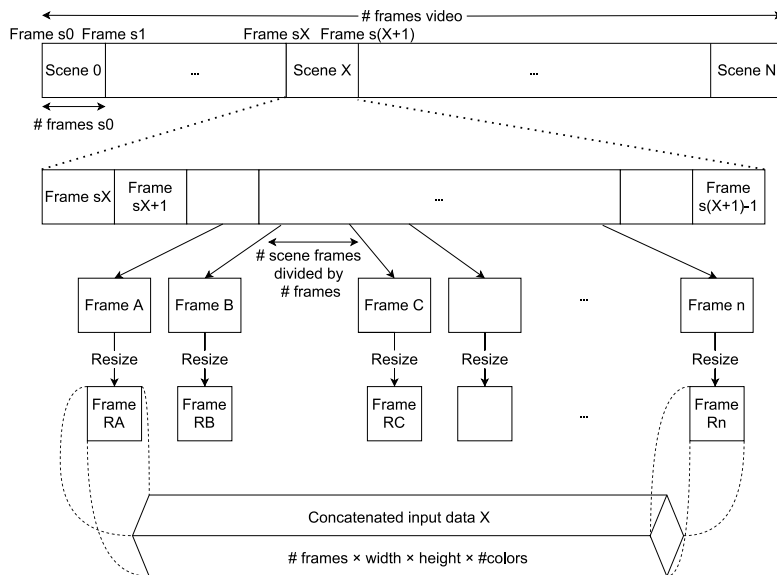
The second approach for scene classification (and quite the opposite to the one above) that we took with Lukáš Korel and Jiří Tumpach in [A1] is based on a pre-processing of



several frames from the video clip with a state-of-the-art pre-trained image processing neural network. Our approach then combines some of their internal weight vectors (that should correspond to some embedding) and classifies only the merged representation. Our paper with an exploratory work in this field is also a part of the digital attachment.

The basic idea behind this approach is that if an image processing backbone such as ResNet50 [58], AlexNet [135] or VGGnet [57] is suitable for classification of the scene on its own from a single picture (such as presented in the results of the Places dataset [65]), there should be a possibility to adapt the scene classification to the video source. Because we had limited computational capacity at the time, we selected just the best-performing network on the Places dataset – VGG. More precisely, we use the VGG19 variant.

Our primary goal of combining a description of several frames is to eliminate the effect of selecting a single frame from the sequence that is too obstructed by a foreground object, blurred, or otherwise unsuitable for the scene classification. This assumption does not hold too well if the shot is static and the foreground object takes a significant portion of the frame, such as a sequence containing close-up dialogue shots. However, we intentionally opted out of any engineering approach that would solve just this issue in favor of exploring the approaches to combinations of the frame representations.



**Figure 6.3.** Preparation of video sequence for scene classification, considering  $n$  equidistant frames from each shot and resizing each frame to  $224 \times 224$  pixels.

To pre-process the video sequence, we first split the video into individual shots (as we already did in the preparation of the dataset), select  $n$  equidistant frames from each shot, and resize them to a shape suitable for the direct processing by VGG ( $224 \times 224$  pixels  $\times 3$  color channels); see Figure 6.3. Each frame is then taken through the TensorFlow `preprocess_input` function and processed by a VGG19 network pre-trained on the ImageNet dataset [136]. We selected ImageNet on purpose, as we do not want to introduce the concept of a scene into the descriptor before the actual classification and simultaneously want to simulate the real-world scenario of a limited training dataset unsuitable for training any image-processing network from scratch.

To obtain the representations of individual frames instead of the final classification, we remove the last Softmax layer from the VGG19 network, resulting in a one-dimensional vector of 4 096 values for each considered frame. On top of such repre-

sentation, we experiment with several approaches that combine the information from individual frames into a single global representation and consequently the class distribution with the following trainable layers, where  $n$  is the number of considered frames and  $c$  is the number of output classes:

- Product: layer that computes a product of all values in the dimension of time, resulting in a single  $c$ -dimensional vector.
- Flatten: flattening to a vector of size  $n * c$ .
- Average: average-pooling in the time dimension with kernel size  $n \times 1$ , thus resulting in an average function of each feature over time.
- Max: max-pooling in the time dimension with kernel size  $n \times 1$ , thus resulting in a maximum function of each feature over time.
- LSTM: long short-term memory layer with default settings and  $c$  units.
- BiDi-LSTM: the same long short-term memory layer as above, encapsulated in a bidirectional wrapper.

All combination layers need several additional layers in their vicinity to function correctly. After the product layer, we added two extra dense layers with 1 096 units, followed by a layer with  $c$  units and a softmax activation layer. Flatten is followed by a single dense layer with  $c$  units and a softmax layer. Layers computing the average and max for each feature use alpha dropout with regularisation set to 0.1 on a dense layer with 2 048 units before the combination and another dense layer with 1 024 units after, followed by another dense layer with  $c$  units and a softmax layer. Both of the LSTM-based approaches add just a softmax layer after the combination.

During our experiments, we fixed the number of frames extracted from each shot ( $n$ ) to 20, and the number of output classes ( $c$ ) is 52, as we limited the hierarchical scene descriptor to only three levels.

For evaluating the individual combination approaches, we considered each episode a separate training set (to model the effect of limited training data) and tested the trained feature combination approaches on the rest of the episodes. This way, we also gained a possibility to compare our approaches in different setups and gain a better insight into the combination’s overall quality. To reduce the effect of random initialization, we repeated each experiment at least seven times.

**Table 6.3.** Comparison of the approaches to combine features extracted from a set of frames for scene classification. Values correspond to the count of datasets on which the model in the row has higher average accuracy than the model in the column. Value is in italic if the difference in accuracy is not significant according to the Wilcoxon test. If the difference is significant, then the value is presented in bold.

↓ wins over →	Product	Flatten	Average	Max	LSTM	BiDi-LSTM	Score
Product		<b>16</b>	<i>6</i>	<b>16</b>	<i>5</i>	1	44
Flatten	1		0	<i>10</i>	0	0	11
Average	<i>11</i>	<b>17</b>		<b>17</b>	3	1	49
Max	1	6	0		0	0	7
LSTM	<i>12</i>	<b>17</b>	14	<b>17</b>		3	63
BiDi-LSTM	<b>16</b>	<b>17</b>	<b>15</b>	<b>17</b>	<i>14</i>		79

To summarize our experiments, the model with a max-pooling provided the worst results with overall accuracy consistently around ten percent (mean 9.3 %). This method also provided better results only over seven datasets and combination methods.

The model with the flatten layer (that behaves much like a combination of just dense layers but discards the temporal structure of the data as well) was only slightly better. Only on some of the datasets and by pure chance, the approach provided an excellent prediction accuracy; some runs even resulted in a 100 % accuracy. The overall mean accuracy was, however, just 23.6 %.

Averaging the values of the descriptor over time resulted in better classification accuracy in 49 dataset and feature combination approaches and is significantly better than the models with the flatten or max-pooling layer. The mean of the prediction accuracy landed on just 32.2 %, but the standard deviation is by far the smallest from the usable models, making the average-pooling model the most stable one.

Surprisingly, the product model (that just multiplies the individual values of the image descriptor over time) has an overall mean accuracy of 43.7 %, even higher than the models based on the LSTM and the peek accuracy over 80 % for multiple datasets. However, the standard deviation of the accuracy is the highest from the tested approaches. Thus, the product model wins in only 44 combinations of dataset and model comparison, although again significantly better than the models with the flatten or max-pooling layer.

As we somewhat expected, LSTM models provided the best overall results. They can retain some of the context information from the time domain, therefore enabling to shift the attention in favor of the frames that provide information most beneficial for the final classification while retaining the data structure. Even though the overall mean accuracy of the uni-directional model is only 40.7 %, it still wins in 63 combinations (over 74 %!) of the dataset and model.

The bidirectional LSTM then adds a possibility to collect the information from both sides of the temporal context. This improves the mean predictive accuracy to 47.8 %, while the improvement is consistent over all datasets. As a result, we can state that the bi-directional LSTM is significantly better than all other models, except uni-directional LSTM, where we also observe an improvement, just not a significant one.

## 6.3 Challenges in Video Scene Classification

During our experiments with the scene classification (or, more precisely, recognition, as we are mainly interested in approaches that provide an as-specific-as-possible identification of a given scene), we encountered several issues that struck our interest but remained mostly open from our side.

### 6.3.1 Color Invariance

The color tonality of a clip recorded in daylight depends on the camera's white balance setting (which may change over time), time of the day, and current weather conditions. For example, in the case of bad weather, all the colors are somewhat faded and dull, and in the case of a clear sky and correct white balance and exposure setting, the color is more saturated. When a shadow is casted from an object in the scene or a cloud, the value component of the color changes. Furthermore, as the daytime changes, the hue also changes a little.

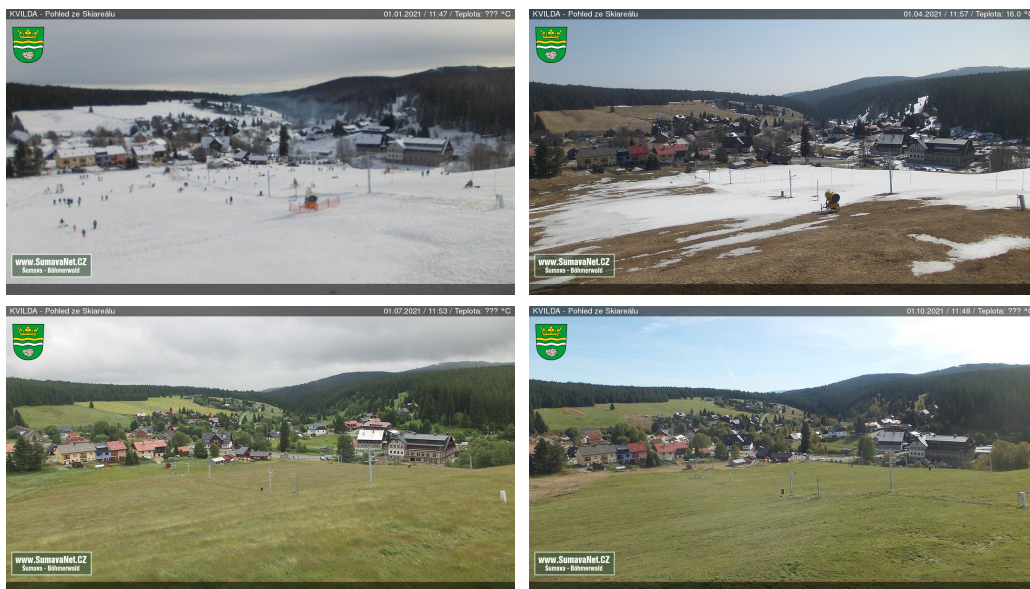
In our experiments with Tomáš Šabata, we used the hue-saturation-value (HSV) color representation precisely for this purpose, but we overestimated the discrimination power of just the Hue component.

Articles, such as [137], present methods of extracting the color information invariant to the sunlight by computing the relationship of a red/blue ratio and some power of green/blue ratio. However, as no such color re-mapping is commonly used within the computer vision community, we have not considered it in our research, although it might be an attractive pre-processing option.

### 6.3.2 Incorporating Time Insensitivity

Another exciting area that came to mind when processing time-lapses captured over multiple years is the stability of scene classification over time. More precisely, for our research, the insensitivity to seasonal changes in the scene and changes in the structure of the scene caused by natural processes or human activity.

For example, for the NARRA project that documents long-term changes in the Šumava national park, we helped collect multiple years of images from several webcams in the area. Figure 6.4 illustrates how much the scene may change over an extended time, even though we still want to classify all of these images with the same label.



**Figure 6.4.** Illustration of changes in the visual appearance of the scene over a longer period due to weather seasonality and changes introduced by a human.

In general, we have two distinct solutions. As one possible approach, we can engineer a subset of features that are invariant to the changes in mind, such as considering not the color tonality of the scene but rather the placement of distinct objects in the scene that are not supposed to change over some time, such as the color of facades. Another approach that seems to be used more over time as the training of neural networks became more available is to let the neural network decide which subset of features is the best for the task at hand. However, the later approach hinders the explainability of such an approach.

### 6.3.3 Partial Scene Matching

Connected to the challenge above is a possible solution of acquiring the similarity of given scene representations based on matching limited sections of the considered im-



**Figure 6.5.** Illustration of changes in the frame’s content over a longer period due to human activity.

ages. For example, let us consider two frames from another timelapse that captured a reconstruction of a shopping mall in Figure 6.5.

As a human observer, we can easily decide that the two images were captured from approximately the same location, although with a slightly different camera position and setup. As clues for the decision on the location, we can theoretically use the parking lot’s layout and Prague’s panorama in the distance. However, it is far from easy to extract such engineered features automatically.

The video content processing methods, which we are primarily interested in, might provide some additional and helpful information. If the camera moves around the scene, we consider a theoretical possibility of segmenting the captured frames into planar approximates of object faces and index (and match) the images by the color-invariant texture of such object representations. However, during our investigation of approaches viable to tackle the segmentation of the individual frames of the video in such a manner that the individual segments can be stitched into a (possibly hyper-resolution) textures of the objects, we returned to the problem of visual simultaneous localization and mapping. We discuss the issues of visual simultaneous localization and mapping in more extensive detail in Section 4.4.5 and can conclude that there is no silver bullet in this area so far, and further investigation is required.

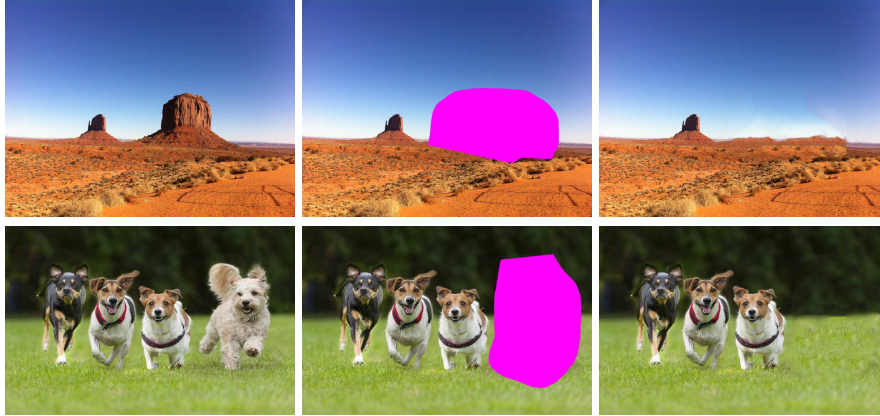
### ■ 6.3.4 Foreground Object Removal

One of the areas that we have just lightly touched here (and we will discuss in some detail in the following chapters) is the presence of foreground objects obscuring the background and the possibility of minimizing their influence over the scene classification.

First, we need to detect such objects, ideally with a mask rather than a bounding box, and remove them from the image information passed along for further processing. One possibility for filtering the data out is to pass the background mask through the processing and mark the pixels that belong to the foreground objects as invalid. Another possibility of still photo processing (that does not require significant modifications to the classification architecture) is to fill the void in a content-aware manner, such as discussed in [138] and illustrated in Figure 6.6.

In contrast to processing a single photo, the processing of video sequence may again benefit from the fact that moving foreground object (and camera) may reveal different background parts in different frames, and the inpainting can be, therefore, based on the actual appearance of the background.

For one of our experiments, we took a video sequence of two people walking across a hallway, captured by a camera moving on a dolly (camera rails), therefore maintaining a parallel position to the wall in the background, and followed the actors as they took the stairs up to the next floor.



**Figure 6.6.** Removal of an object in the foreground by providing its mask and inpainting the background with multi-scale neural patch syntehsis. Adapted from [138].

To obtain the information on the scene, we mapped the frames onto one another with the help of visual features. Then we assumed that the background pixel value would be visible in more frames than that the pixel would be covered with an object in the foreground. This assumption reveals a possibility to obtain the visual information on the background by computing a median pixel value without any expensive method of foreground object detection.



**Figure 6.7.** Scene background extracted by transposing frames of a clip into a common pixel space and selecting a median value.

The result of this particular experiment is presented in Figure 6.7. Although this seems like a very promising result, the memory requirements for the computation of the median value are enormous. Also, the registration of the frames is a relatively expensive task that can be combined with the tracking of objects in the foreground if a proper visual feature tracking approach is used. Consequently, a tracklet clustering approach can also be employed in the foreground objects' segmentation to obtain a good background representation. Therefore, in our following research, we focused on the issue of efficient matching of the visual features.

# Chapter 7

## Local Visual Features in Multimedia Processing

A very different approach to image processing is based on the detection and registration of local visual features (we also refer to them as interest points or salient points in our publications). In the context of our work and following the Section 4.3.2, we define the local visual feature as an image pixel with specific surroundings in the image function that we expect to be sufficiently similar in consecutive frames of the video to enable mapping of the points onto one another. In practice, the visual features are extracted on visual corners (positions with a significant gradient in at least two directions) and filtered by additional conditions and spatial restrictions.

### 7.1 Methods of Visual Feature Matching

To recall the commonly used methods of visual feature matching and allow us to build upon them, we provide a short overview of the two fundamentally different approaches. After that, we provide a detailed description of our feature matching algorithm that takes inspiration from both approaches and combines them into a single system suitable for video signal processing.

#### 7.1.1 Kanade-Lucas-Tomasi

In the context of the video signal and the assumption that we are processing a continuous clip without any edits, we can only detect visual features as an initial step. Then we can follow by a search for the most similar patch in the vicinity of the original position in the next frame.

This patch-mapping approach is a basis of the Lucas-Kanade algorithm [139], commonly combined with feature point selection of Shi-Tomasi [140], and jointly referred to as **KLT** or **Kanade-Lucas-Tomasi**. For the computation of the new feature point position, a corresponding spatial intensity gradient of a window around a given point of interest (typically  $31 \times 31$  pixels) is found. The iteratively discovered displacement of the intensity gradients is then considered the actual optical flow vector and translated to estimated motion vectors of the nearby points of interest.

Lucas-Kanade algorithm uses a pyramidal approach to ensure partial scale invariance and faster detection in a larger neighborhood, where the optical flow is computed on the coarsest level of the pyramid and then refined on the lower levels up to the full resolution. To assess the error of the point tracking, KLT uses backward tracking, where the flow detection is executed again in the backward direction, and pixel distance from the original point is calculated. Feature points with a bidirectional error of more than 3 pixels are usually considered lost and not considered in further tracking. Therefore, an occasional refresh of the tracked features is needed.

#### 7.1.2 Descriptor Distance Matching

A faster and spatially unrestricted visual feature matching can be achieved by the extraction of the feature descriptors (such as SIFT or SURF) from both of the frames that

should be registered, and **comparing the values of the descriptor** instead of matching the pixel values. This way, the matcher can propose point matches even for a video feed with significantly reduced framerate (such as one frame per hour in timelapse videos) and unstable camera position.

On the other hand, such spatially unrestricted visual feature matching produces many false candidates that need to be filtered out. This issue is caused primarily by the limited discriminative power of the visual feature descriptor with direct dependence of its dimension to a gradient in some part of the image function and its rotational invariance.

A universally applicable filter is the **Lowe's ratio test** [43], where we select two closest visual feature descriptors for each query feature and append the match of the query to the closest match only if:

$$dist_1 < dist_2 * c,$$

where  $dist_1$  is the distance to the closest match by the visual feature descriptor,  $dist_2$  is the distance to the second closest, and  $c$  is a multiplication factor between 0 and 1 that determines the required ratio between the distance of the closest and second closest match, commonly set to 0.7.

The main idea behind this filter is that if the first two matches have very similar distances, the query point itself may have been improperly selected, and its feature descriptor is not descriptive enough to deduce a unique match of high quality.

Other filters exploit the assumption of the continuous video stream with a high-enough framerate, therefore proposing a **limit on motion vector length**. This filter may break the frame registration in cases of extreme magnification and sudden motion. However, due to imposed motion blur, the successful registration may lead to false object reconstruction and other issues down the processing pipeline.

We can impose another limitation to the visual feature matching in the form of discarding all matches that do not **follow a common transformation**. Sadly, as we do not have any ground-truth information on the feature distance from the camera, we cannot simply enforce a 3D transformation matrix. However, under the relatively strong assumption that most of the captured features belong to the background (and that we will move towards in the next chapter), we can approximate the effect of camera motion by constructing a homography matrix with a Random Sample Consensus algorithm (RANSAC) [50]. Therefore, even though there is an abrupt motion across the whole frame caused by the rapid change in camera position or angle, we can transform the pixel data to a shared space and use traditional image processing methods that would otherwise suffer from such abrupt change.

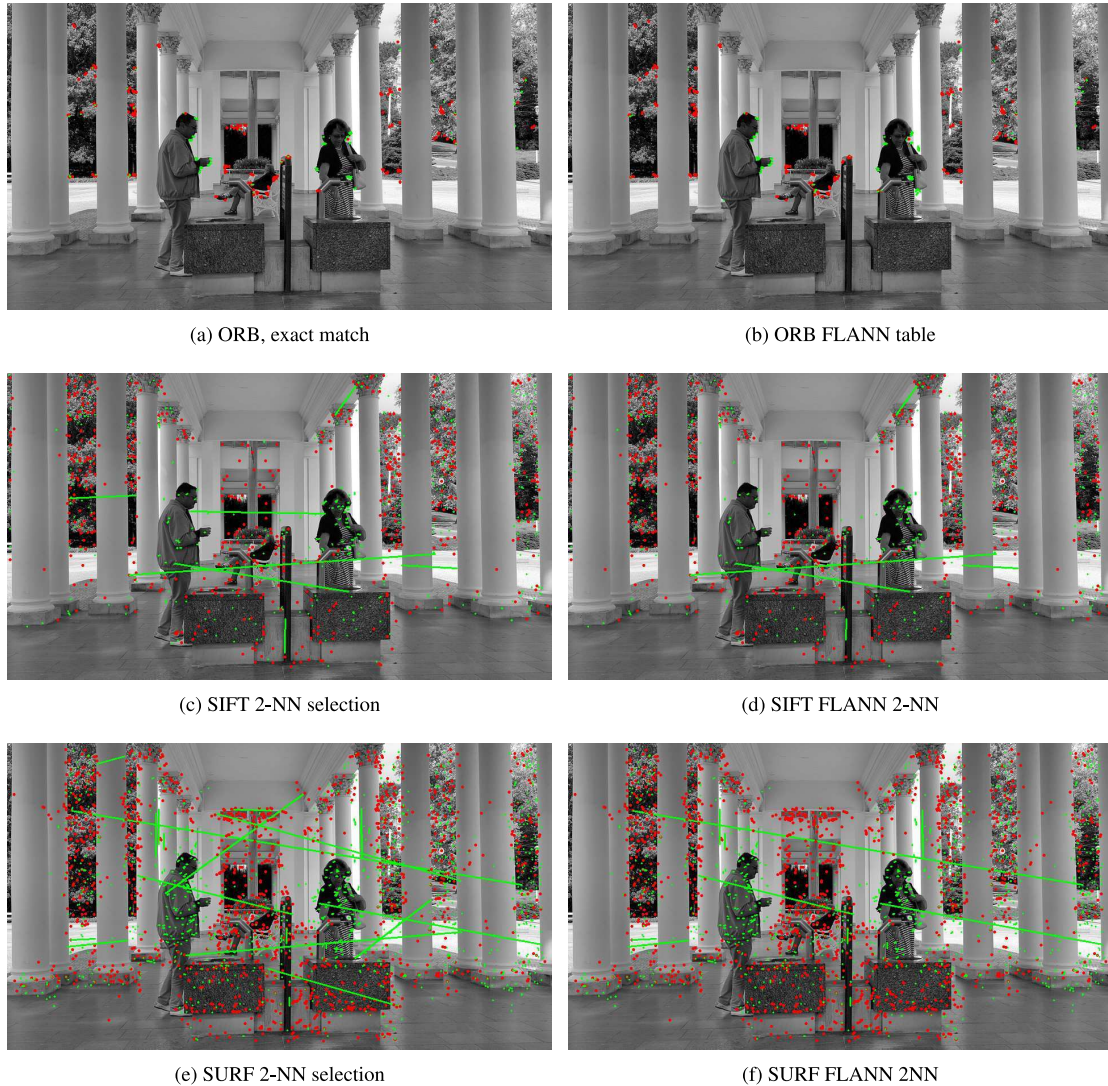
As an additional feature of the BRIEF feature descriptor [46] utilized by the ORB feature detection and description system [45], we can **directly assess the similarity of the descriptor** with the Hamming distance to obtain an actual similarity of the point neighborhood (opposed to the SIFT and SURF, where the descriptor is not constructed to support such use-case).

To decide on the feature detection and description algorithm to use in our follow-up research, we conducted a series of experiments in [A11]. First, we considered the overall quality of the matches proposed by the individual detectors in their usual setup (with the match to the feature with the smallest descriptor distance in ORB, or selecting the two nearest neighbors and deciding on the match with Lowe's ratio test in SIFT and SURF). We conducted such an experiment with a brute-force matching, where the visual feature descriptors are scanned linearly for each query, and with Fast Library



for Approximate Nearest Neighbours (FLANN) [141] that should provide a significant speedup in the search phase.

Our key takeaway from this experiment, with the visual comparison provided in Figure 7.1, is that ORB does provide significantly fewer visual features that stick much better to the areas with more pronounced and thus more stable gradient change (such as edges of objects), whereas SIFT tends to search for structure within the objects, and SURF features (as they are based on blob detection) are present inside the object structures even more.



**Figure 7.1.** Visual comparison of motion vectors detected on an image with a resolution of  $960 \times 540$  pixels with different local image descriptors and matching algorithms. Red dots represent no motion; green lines represent a detected motion vector to the next frame.

We sadly observed no speedup during this particular experiment if using FLANN, but we do use it extensively in our further research and with good results. Interestingly, the use of FLANN also leads to some improvement in the quality of visual feature matches. However, these are likely just a side-effect of our setup.

However, the visual feature matching performance depends significantly on the selected feature detectors and descriptors. During this initial experiment, ORB took an impressive lead by processing 50 frames of Ultra-HD video (with a resolution of

3840 × 2160 pixels) in two minutes, whereas SURF needed almost 2.5 hours, and SIFT needed almost 7 hours to complete. Even if we consider that ORB returned only 13 740 matches and SIFT produced over 1.5 million matches, the cost is still in favor of ORB.

In a follow-up article [A9], we experimented with a novel approach of joining the detected motion into longer tracklets. Simultaneously we tested the performance of the Kanade-Lucas-Tomasi and a custom matching algorithm for ORB features that limits the number of points against which the Hamming distance of the descriptor is computed to a spatial neighborhood of the query position, thus simulating partially the approach of KLT. To speed up the search in the spatial domain, we indexed the visual features in a k-d tree [142] by their position and used a radius search function of library `nanoflann` [143].

As a result, our approach is now able to achieve performance comparable to KLT on image resolution under HD (720 vertical lines), and simultaneously we introduced a tradeoff in the speed matching two consecutive frames that depend on picture resolution and the number of detected visual features, as we illustrate in Table 7.1.

**Table 7.1.** Comparison of elapsed time in [ms] to process a single pair of frames. Darker green represents processing time suitable for frame rate above 30 fps, lighter green above 25fps.

a) Kanade-Lucas matcher with Shi-Tomasi detector executed on each frame

Points	Vertical resolution											
	180	360	540	720	900	1080	1260	1440	1620	1800	1980	2160
100	2.86	8.0	16.01	27.8	43.03	62.79	84.3	106.87	137.56	167.6	203.23	246.07
200	3.27	8.47	16.83	28.25	43.98	63.57	84.61	107.06	133.76	164.99	201.39	245.36
300	3.6	8.78	16.86	28.62	43.94	63.03	84.8	109.33	135.98	165.48	201.85	246.09
400	3.58	9.16	17.53	28.98	44.72	64.06	84.6	109.88	134.55	169.59	202.4	245.68
500	3.56	9.59	17.93	30.23	45.35	64.08	83.47	108.63	135.25	164.1	205.1	242.84
600	3.56	9.95	18.3	30.38	45.46	64.67	84.81	109.08	139.97	167.74	201.41	243.62
700	3.57	9.03	18.4	30.41	45.02	64.77	87.1	107.72	136.34	167.45	202.32	248.48
800	3.58	9.03	19.08	30.64	46.46	65.12	86.88	108.62	136.68	167.77	207.38	244.43
900	3.59	9.06	17.36	30.77	46.62	65.5	86.66	111.48	138.7	168.93	206.52	247.6
1000	3.56	9.05	17.57	31.71	46.34	66.9	87.77	112.43	135.08	171.98	203.19	253.53

b) Our visual feature matching algorithm with the detection of visual features by ORB

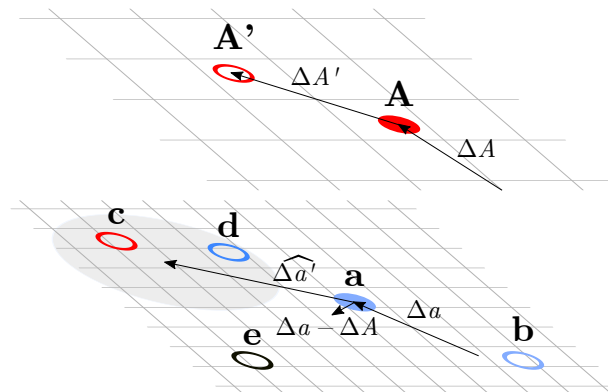
Points	Vertical resolution											
	180	360	540	720	900	1080	1260	1440	1620	1800	1980	2160
100	2.42	4.54	7.62	11.43	18.24	24.44	32.67	39.83	45.92	55.08	69.95	77.39
200	2.65	5.74	9.61	14.18	19.4	26.92	32.86	41.63	51.48	60.62	68.93	78.53
300	2.87	6.75	11.93	17.32	23.53	30.86	38.1	47.13	54.38	65.07	77.42	84.01
400	3.02	8.14	14.48	20.27	28.07	34.91	40.86	51.47	60.89	68.53	80.15	86.53
500	3.0	9.21	17.19	23.95	33.3	39.2	46.42	57.09	67.17	78.61	89.89	95.32
600	3.01	10.18	19.85	28.85	39.74	46.02	52.78	64.24	75.12	87.28	95.61	103.02
700	2.99	11.25	22.48	32.84	42.71	52.23	59.47	70.97	84.52	95.67	106.25	116.92
800	2.99	11.5	24.23	36.37	49.08	57.02	66.23	78.34	95.32	104.19	121.68	126.52
900	3.0	11.92	26.53	40.93	54.26	65.53	74.12	82.9	97.13	114.92	130.15	146.47
1000	2.99	12.2	28.65	46.52	58.58	72.27	84.01	92.25	111.85	127.17	144.6	152.49

## 7.2 Hierarchical Visual Feature Matching

So far, we assumed that the position of the visual feature does not change significantly over time. Simultaneously, we considered the visual features detected on all image scales during the matching. This approach has major issues, as it does not consider changes in the image function induced by abrupt camera motion, nor does it predict the motion of objects in the scene to approximate the new position of the visual feature before searching for the matching visual feature.

To tackle these issues, we introduce a novel method of hierarchical visual feature tracking [A4]. Inspired by the Kanade-Lucas-Tomasi optical flow estimation method,

we leverage the fact that the visual features are already detected on a Gaussian pyramid and match the features on the highest level of such pyramid first. This initial estimation is fast, thanks to the limited resolution of the layer, and results in a limited number of visual features but provides spatially-dependent information on the estimated optical flow across the frame. Points detected on lower levels of the pyramid can leverage such information to predict the new position of the visual feature better and can, therefore, eliminate matches with similar visual features that are located elsewhere in the frame while enabling the matcher to keep up with moving objects. The matching of a single visual feature is illustrated in Figure 7.2.



**Figure 7.2.** Method of hierarchical visual feature tracking. The upper section with a coarser grid represents a higher octave of the Gaussian pyramid with already paired features. The lower section depicts estimation of the new position of the feature  $\mathbf{a}$  ( $\widehat{\Delta a'}$ ) in the lower octave of the incoming frame based on the already detected motion of the feature  $\mathbf{A}$  ( $\Delta A'$ ) in combination with the previous motion of the feature  $\mathbf{a}$  relative to the higher octave ( $\Delta a - \Delta A$ ). Features from the previous frame are shown as filled circles; hollow circles represent detected features in the incoming frame. The Colour of the circles encodes the feature description. The grey area represents a neighborhood considered in feature matching.

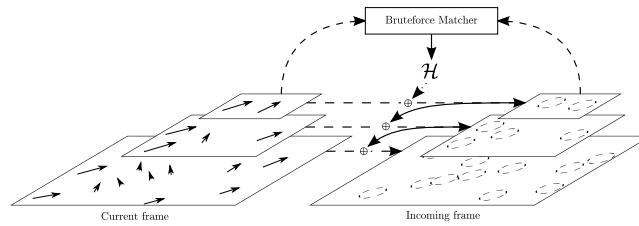
Furthermore, to eliminate the cold initialization of the visual feature matching in the first layer, we added an additional prediction layer on top that matches the features using a brute-force matcher without any spatial restrictions on the position of the matched point. To make this prediction layer more stable, we keep only a homography matrix extracted from the confirmed matches for the predictions in the following layer. Figure 7.3 presents an overview of this approach. This way, we both account for the possibility of high global optical flow induced by the rapid changes in the camera position and simultaneously limit the number of visual feature descriptor comparisons that need to be carried out thanks to the new position estimation.

Although this approach should be theoretically applicable to any visual feature detection and description method, we carried out our experiments on the ORB visual features, as they provided us with the ability to assess the match of the visual features just by computing the descriptor distance and does not necessarily require any additional validation.

The implementation source code in C++ is publicly available at: [https://github.com/petrpulc/gpu\\_orb\\_tracker](https://github.com/petrpulc/gpu_orb_tracker).

To confirm our hierarchical matching approach, we conducted a set of experiments on a Multiple Object Tracking dataset [144], where we verified how the tracklets con-



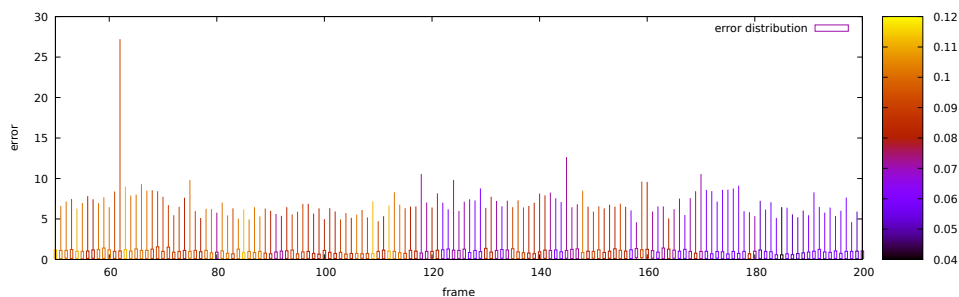


**Figure 7.3.** Overview of our hierarchical ORB tracker. The dashed arrows represent the flow of detected features (points with descriptors), the full lines represent the flow of matched features (relative motion vectors to assist prediction).

structured by our visual feature matching approach retain their association to the same object throughout the whole video clip and what is the average distance from the expected point position considering rigid objects. The visual confirmation of our approach is presented in Figure 7.4, the motion tracking error over time in Figure 7.5.



**Figure 7.4.** Resulting tracklets from sequence MOT17-04, valid in the displayed frame (some are already lost). Ground truth bounding boxes are shown in blue.



**Figure 7.5.** Distribution of motion tracking error within each bounding box for individual frames of sequence MOT17-04. Boxes represent first and third quartile, whiskers absolute minimum and maximum. Colour represents the fraction of lost objects (no features detected in their bounding box).

## 7.3 Foreground Segmentation based on Estimated Motion

As a common thread in our research, we experimented with the segmentation of the video sequence into background and foreground objects representation.

The exploratory work conducted in [A11] considered the non-hierarchical feature matches in two consecutive frames and used the angle and length of the estimated motion vectors in hierarchical clustering. This approach provided us with a set of visual features that we expect to belong to the foreground (and, respectively, to the background).

However, we would like to segment the original frames on the level of individual pixels to allow visual information processing in consequent parts of the processing pipeline – such as stitching and inpainting of the background and providing detailed information on the objects in the foreground. To this end, we utilize a Grid Cut algorithm [66] that can efficiently compute minimal-energy cuts in a 2D lattice by solving a min-cut (also known as max-flow) problem in a given graph. To adapt the algorithm for real-world photographs and video recordings, we need to filter the image with a Laplacian of Gaussian kernel first, obtaining the information on local lightness change in the neighborhood of a given pixel.



**Figure 7.6.** Result of frame segmentation based on clustering of estimated motion vectors according to their angle and size and providing the clustered visual features as inlets of the Grid Cut algorithm.

As a preliminary result, we could segment the objects in the foreground relatively cleanly, as illustrated in Figure 7.6. However, the objects were not captured whole, and computation of the clustering, based only on the information from a single pair of frames, was relatively expensive and unstable.

We continued improving the method with Oliver Keruř-Kmec in [A7] and [A5]. However, our experiments provided rather unsatisfactory results for use in the meta-learning framework.

Therefore, we conclude that although segmentation of a video signal based only on the extraction of visual features would be an excellent addition to the multimedia processing toolbox (and we had partial successes in this field), it is somewhat impractical and, in the end, more computationally demanding than the existing methods of image segmentation.

Therefore, our focus shifted slightly into providing improvements to already existing methods of multiple object tracking, which we will present in the next chapter.

# Chapter 8

## Multiple Object Tracking

So far, we were mainly concerned with tracking individual visual features in the video clip and then possibly combining such visual features and their tracklets into the detections of objects in the scene. However, considering the MOT challenge [144] results, a rather opposite approach became a de-facto standard in multiple object tracking, which we refer to as “tracking by detection.”

### 8.1 Tracking by Detection

This approach usually employs a detector that localizes an object that we want to track in the incoming video frame, describes its low-level visual appearance, and searches for a close match in the next frame by some representation of the visual appearance.

To detect an object, we may, for example, utilize a “You Only Look Once” detector (YOLO) [52]—an advanced, state-of-the-art real-time object detector that uses a neural network with just a couple of convolutional and residual layers, and evaluates all object predictions simultaneously. However, this network is sadly insufficient for us, as it does not provide the object description, nor does it segment the object for further semantic processing.

In the case of a static camera, one possible approach is to segment and match objects based on their contours and histograms [145]. A more generic method may propose a new contour from the previous frames and fit it over superpixels in the incoming frame [146]. However, these approaches are sensitive to the propagation of errors in detected object contours.

Another possible approach is to train a two-branch convolutional neural network that detects the object contour and computes an object appearance vector simultaneously [147] or directly provides a spatiotemporal representation of the object [148].

However, we do not need to extract contours explicitly. One of the significant benefits of deep neural networks is that we do not need to explicitly care about feature extraction if we provide the network with a sufficiently large dataset, such as Common Objects in Context (COCO) [149].

The YOLO object detector uses just the object bounding boxes for training, whereas the Mask R-CNN network [56] also utilizes the object contours from the dataset and (aside from class and bounding box prediction branches) contains a branch for inferring the association of the pixel data to the detected object—a mask.

Once we detect the set of objects in each frame of the video sequence, we need to develop an approach of associating the detected objects from one frame to the other.

A possible solution is to devise an object re-identification approach by adding another branch to the Mask R-CNN network, which produces descriptions of the detected object suitable for re-identification. A MaskTrackRCNN [150] is one of the prominent examples based on ResNet-50, pre-trained on COCO and trained on yet another large dataset: YouTube-VOS [151]. Alternatively, we may ditch the preprocessing network altogether and train a new network end-to-end, such as FairMOT [152] did.

However, proposing a match of two objects based solely on their visual appearance may be misleading. For example, what if two nearly identical objects appear on the scene? With the currently deployed approaches, we usually have no visibility into the decisions of the matching algorithm as the descriptions bear no semantic meaning.

One of the commonly used approaches to mitigate obvious mismatches is to verify the proposed match based on the previous motion of the object. However, they, in turn, depend on the correct association of the objects in the previous frames, thus creating a closed feedback loop.

## 8.2 Feature Tracking and Object Detection

To mitigate the issues with detection mismatches, we propose in [A2] a combination of our unsupervised tracking algorithm based on matching semantically rich visual features with the object detection algorithms. This way, we leverage the high precision of the state-of-the-art object detection approaches but do not have to rely only on any object description to match the detected objects.

In a nutshell, we take the mask of each detected object in the last registered frame and collect visual features with location within such mask. On the arrival of a new video frame, we execute both the detection of the objects and the detection and mapping of the visual features and connect the object detections that share most of the mapped visual features. As this translates to a problem of optimal pairing, we utilize the Hungarian algorithm as described in [153].



**Figure 8.1.** Top: Visualization of the Mask R-CNN object detections for one of the objects (one in ten frames) connected by the tracklets from our hierarchical visual feature tracker. Bottom: The extracted visual representation of the tracked object (one in four frames).

Consequently, in combination with a suitable object detector, the above-proposed tracker can match objects from one frame to the other quickly and without additional training, as presented in Figure 8.1. However, as we rely entirely on visual feature tracking for object association, this version cannot cope with complete object occlusions. To this end, we have to resort to some object re-identification approach pre-trained on

a suitable dataset for these cases. However, as the complete occlusions are relatively rare and the training datasets such as MARS (Motion Analysis and Re-identification Set) [154] may not be entirely suitable for the task at hand, we experimented with the creation of a custom training dataset for object re-identification based on these limited matches and consequent fine-tuning of a re-identification model.

### 8.3 Fine-tuning a Re-identification Model on a Custom Dataset

The preparation of the dataset for fine-tuning of the re-identification model from the Multiple Object Tracking Dataset is relatively straightforward and follows closely the procedure described above. The sequence fragmentation does not pose a significant issue here, as we close the group of representations connected to a single object once we lose a connection by the tracked visual features and start a new object group, even if the object is visually similar.

For object detection, we use the Mask R-CNN network with the provided model trained on COCO. We include the model in our source code repository, as we needed to update the source code to the newer version of TensorFlow. The model weights can be downloaded from [https://github.com/matterport/Mask\\_RCNN/releases/tag/v2.0](https://github.com/matterport/Mask_RCNN/releases/tag/v2.0).

Our algorithm of visual feature tracking is slightly modified from the C++ version presented in the last chapter and is rewritten to Python to enable easier interfacing with the neural networks. One of the more significant changes is that only the area of the frame that is not covered by any of the object masks is considered background and used to compute the homography matrix in the top-most layer of the hierarchical tracker.

One significant difference from the datasets such as MARS is that we are using an object detector with built-in mask inference and, therefore, we can include the mask in the alpha channel of the exported images. However, to make our dataset otherwise compatible with MARS, we do not null the RGB values in the transparent pixels and store the resulting object representation images in the same resolution of  $128 \times 256$  pixels.

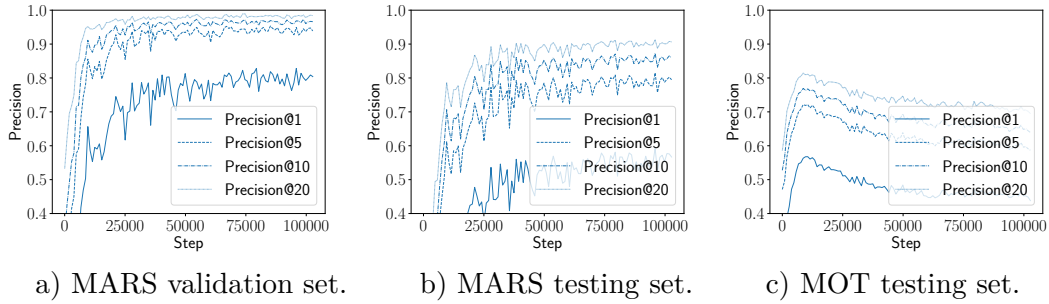
As a result, our dataset extraction approach results in a relatively large number of short object tracklets. On both training and testing MOT16 video data, we extracted over ten thousand tracklets with the maximal tracklet length of 1 130 frames and 1 646 tracklets over twenty frames long.

The full implementation of our object tracker and dataset exporter is available at [https://github.com/petrpulc/MRCNN-ORB\\_to\\_MARS\\_dataset](https://github.com/petrpulc/MRCNN-ORB_to_MARS_dataset) and the exported dataset is available for future experiments on Kaggle: <https://www.kaggle.com/petrpulc/mask-rcnn-people-tracklets>.

To evaluate the added benefit of our task-specific dataset, we will stick to the model of Cosine Metric Learning with a cosine softmax classifier, as presented in [155], and follow closely the same evaluation protocol.

Because we provide a fourth channel with an object mask in our dataset, whereas the original model considered only RGB images, we modified the image loading to accept images in PNG format and changed the number of input channels. Otherwise, the architecture is the same to allow more direct comparison. We publish the model modifications, including the original training code in GitHub repository: [https://github.com/petrpulc/cosine\\_metric\\_learning](https://github.com/petrpulc/cosine_metric_learning).



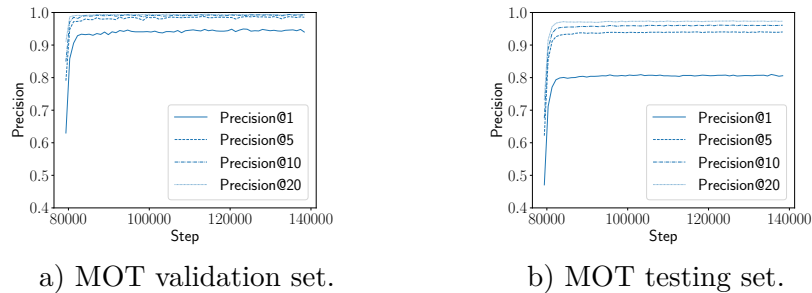


**Figure 8.2.** Precision@ $k$  of the altered cosine metric model trained on MARS with added alpha channel.

Due to the change in the input dimensionality, we sadly cannot reuse the pre-trained model provided by the author and need to train it from scratch. We show the progress of our modified model training on the MARS dataset with original splits in Figure 8.2. Following the original publication of [155], we measure the progress of the training with precision at  $k$ , where  $k$  refers to the number of images considered for assessment after sorting the target gallery by a cosine distance from the query image. In subfigures, we present the precision on the validation set taken from MARS training data, MARS’s testing set, and our testing set constructed from the MOT videos.

From Figure 8.2a) we conclude that our modification by adding a channel with no information (as there is no transparency in the MARS dataset) does not hurt the model at all, as [155] reports the same results. Not surprisingly, after approximately 20 thousand steps, the precision on the MARS validation and testing sets continues to increase, while the precision on our dataset constructed from MOT starts to decrease, which is probably caused by slight overfitting of the neural network.

With that, we take a model pre-trained on the MARS dataset (now with alpha channel) and continue its fine-tuning with our dataset constructed from the MOT video sequences. To this end, we reload a snapshot of the model and reset the mean vectors leading from the unit ball representation to object indices, as they are dependent on the similarity of the underlying objects and define the dimension of the optimization tensor. After that, we can execute a fine-tuning of the network, leading to the results shown in Figure 8.3.



**Figure 8.3.** Precision@ $k$  of the altered cosine metric model trained on MARS and fine-tuned on MOT.

As a result, we can fine-tune the cosine metric model in less than four thousand steps and are able to achieve 80 % precision@1 on the MOT training set, opposed to the maximum of 57 % precision@1 of the model trained only on MARS if stopped training prematurely, but more likely the precision@1 around 50 % if usual stopping condition would be used.

## Chapter 9

### Conclusions

After more than seven years dedicated to multimedia content processing and extraction of knowledge from it, we have to state that this area is highly dynamic and competitive. When we started with our research, deep convolutional neural networks for processing of single images with a minimum resolution was at their beginnings, acceleration of neural computations on graphic cards was cumbersome and required translating the programs into shaders that the GPU was able to process, dedicated hardware tensor processing units were introduced as late as in May 2016. The sufficient power required for using neural networks in video processing was available only a couple of years back.

Therefore, we originally designed this thesis to use as many processing methods from the traditional machine learning landscape as possible and soon discovered its shortcomings in their accuracy.

At the defense of my doctoral study report in 2017, our original motivation shifted slightly towards the issue of explainability and processing speed of the knowledge extraction components, as these were the two most glaring issues of the neural network approaches of the time.

Therefore, we came up with an approach utilizing meta-learning principles to efficiently extract knowledge “from the bottom-up” – by first extracting the most obvious and easy to grab pieces of information.

During my undergraduate studies at the university, I collaborated on a science popularization festival called “Week of Science” where we collected many hours of content, namely a video of the lecture hall (compressed to an unusable quality according to today’s standards), audio signal capturing the speech of the lecturer in a decent quality and pictures of individual lecture slides. We then extracted a dataset documented in Appendix A from this corpus that we used for many of our first experiments.

This dataset worked great for our new goal of presenting approaches that use only the relatively easy to export information, such as the text transcribed from the slides, opposed to harder modalities, for example, the speech transcription. However, as this dataset had a limited scope and many of the approaches were commercially available and hard to improve single-handedly, we had to focus on the areas that were at the time still dominated by bulky neural networks.

One of such areas is the scene recognition, which is commonly taken as a task of scene type classification and is again dominated by deep neural networks that are by design able to exploit any structure in the data if provided with an extensive amount of inputs and a relatively small number of output classes. However, the usability of such approaches is limited if we want to train the network on private datasets and classify the relatively broad spectrum of inputs into a relatively broad set of classes.

One of such tasks was proposed by the project NARRA, where we took part in a wider proposed collaboration between our faculty and the Film and TV School of Academy of Performing Arts in Prague. The proposed information generator (as the component was named in NARRA) should recognize known locations in a sparsely annotated data collection. To this end, we prepared a second dataset based on a popular TV series,

The Big Bang Theory, that contains a hierarchical identifier of the given set for each shot and some additional information that was deemed helpful for further research. The dataset is described in Appendix B.

On top of this dataset, we proposed in collaboration with Tomáš Šabata a rather simplistic method of scene recognition. To tackle the issue of a limited number of annotated data, we prepared a combination of semi-supervised and active learning to demonstrate a significant speed-up in the labeling of the remaining data that the active learning approach can achieve.

One of the significant issues that we had during the pre-processing of the data and training of the classifier was that significant parts of the scene were obstructed by the foreground objects that were changing clothes and thus modifying the color histogram that we internally used for our classification.

Therefore, one of our experiments was concerned with the removal of objects from the video sequences, such that we obtained the background information with as little noise as possible.

This way, we entered the problem of Visual Simultaneous Localization and Mapping that is not trivial to resolve when both the objects and the camera move around the scene. Although there are several heuristic approaches available, to the best of our knowledge, vSLAM is still an open research problem up to this day, and again, the current approaches utilize hard-to-train neural networks that we had and still do not have any ambition to beat.

However, a small subtask of the general problem of a 3D scene reconstruction with moving objects from a single-lens camera video was still accessible: multiple object tracking.

The area of tracking multiple objects in the video is a lot more accessible, as it needs only two ingredients – a method of object detection and a process of matching the detected objects from one frame to the other.

Most of the current state of the art is heavily focused on the first part, as there are great (and money-attracting) applications in the domains of self-driving cars, satellite imagery processing, security, and many more.

To our surprisal, not much attention was given to the second area. One of the possible causes is that object matching cannot be trained on extensive object classification datasets because we are not interested in the general type of the object but instead in the extraction of a unique numerical representation of the particular object. We also have different requirements for the invariance, as the representation should be invariant at least to changes in the camera’s position and the object, lighting, and to the changes introduced by the flexibility of the object itself, such as the motion of limbs of a person.

To resolve this issue, we came up with a somewhat different approach based on tracking visual features and connecting the detected objects not by their numerical representations but by the shared visual features mapped from one frame to the other.

Our approach of deducing the object tracking from visual feature tracking had just one significant flaw: if the object gets obstructed, the visual feature tracking is lost, and the object detection gets fragmented. As this is one of the primary metrics on the MOT Challenge, we could not obtain good-enough results, and several articles were, understandably, rejected.

Therefore, we revisited our idea and partially joined the mainstream approach to object tracking with a method that, in the end, utilizes the “tracking by detection” schema; however, it uses our approach of hierarchical visual feature tracking for the construction of the fine-tuning dataset.

Seven and a half years later from the initial idea, we present to you three datasets, the theoretical approach of meta-learning in multimedia data, several applications of multimodal classification with custom data pre-processing and verification of several classification methods, two scene classification algorithms, and a hierarchical visual feature tracker with its use in domains of image segmentation and object tracking.

As the possible follow-up from this point, an exciting challenge would be the backward propagation of our approaches and, hopefully, propagating the good results back to the original idea of meta-learning. For example, the first step might be to verify the fine-tuned model for object tracking, object removal from the foreground, background imagery stitching and classification by scene, and the selection of new processing algorithms to extract more semantic knowledge from the given clip. However, this journey back may entail just hard engineering work with possibly diminishing returns in the research area.

Another possible continuation of this work is hidden in the individual chapters of our research, as the scene recognition approach was several open ends concerning the use of active learning, the hierarchical visual feature tracking would benefit significantly from an overhaul and more thorough formal verification, and our approach to multiple object tracking provides several places with a possibility of improvement. However, for now, this is the end of our journey.

Thank you, dear reader, for letting me walk you through this journey as well. Feel free to contact me at petr@pulc.eu in case of any questions or remarks.

As a last-minute note, sometimes we have been questioned about the ethical and privacy aspect of such a comprehensive multimedia processing system. The question arises even more urgently with the current geopolitical situation (these words are written on the 26th of February 2022).

Our answer to this question is two-fold. On the one hand, the meta-learning framework does not provide much more power to any of the individual heavily specialized systems that provide, for example, face recognition, vehicle license plate transcription, or detection of suspicious behavior in public areas. Quite the contrary, the deployment of the meta-learning framework will usually be slower than executing a single specialized multimedia processing pipeline.

On the other hand, if we have no prior information about the multimedia file in question, the framework should allow us to extract only the relevant information as quickly as possible and enable matching of the extracted information against a collection of already annotated data. Such a system can help against the spread of disinformation or even help spot discrepancies in the video and its title if set up in a particular way.

On the adverse side, the improvement in the processing speed and precision of the methods connected on top of multiple object tracking (such as facial recognition that can be executed only on some of the frames and filled in by the tracking of the person) can, in theory, lead to higher availability of such multimedia processing methods. At the same time, our approach does not contribute to any weakening of any security system, nor does it violate privacy in any way partially because our methods are specifically designed to be executed as locally as possible and with a minimum of retained low-level information.



## References

- [1] Michal Hořejší, and Michal Mocňák. *Korpus textů o Šumavě*. <http://sumava-corpus.narra.eu/>.
- [2] Eric Rosenzweig. *Conflations*. Academy of Performing Arts in Prague, 2020. ISBN 978-80-7331-531-3.
- [3] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*. 1949, 37 (1), 10–21.
- [4] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. *Evaluating bag-of-visual-words representations in scene classification*. In: *Proceedings of the international workshop on Workshop on multimedia information retrieval*. 2007. 197–206.
- [5] Sheng Xu, Tao Fang, Deren Li, and Shiwei Wang. Object classification of aerial images with bag-of-visual words. *IEEE Geoscience and Remote Sensing Letters*. 2009, 7 (2), 366–370.
- [6] Yi Yang, and Shawn Newsam. *Bag-of-visual-words and spatial extensions for land-use classification*. In: *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. 2010. 270–279.
- [7] Juan Ramos, and others. *Using tf-idf to determine word relevance in document queries*. In: *Proceedings of the first instructional conference on machine learning*. 2003. 29–48.
- [8] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*. 1990, 41 (6), 391–407.
- [9] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*. 1998, 25 (2-3), 259–284.
- [10] George A Miller. WordNet: a lexical database for English. *Communications of the ACM*. 1995, 38 (11), 39–41.
- [11] Tomáš Skopal, and Pavel Moravec. *Modified LSI model for efficient search by metric access methods*. In: *European Conference on Information Retrieval*. 2005. 245–259.
- [12] Jeffrey Pennington, Richard Socher, and Christopher D Manning. *Glove: Global vectors for word representation*. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014. 1532–1543.
- [13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. *Distributed representations of words and phrases and their compositionality*. In: *Advances in neural information processing systems*. 2013. 3111–3119.
- [14] Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. Towards understanding linear word analogies. *arXiv preprint arXiv:1810.04882*. 2018,

- [15] Navneet Dalal, and Bill Triggs. *Histograms of oriented gradients for human detection*. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. 2005. 886–893.
- [16] Navneet Dalal, Bill Triggs, and Cordelia Schmid. *Human detection using oriented histograms of flow and appearance*. In: *European conference on computer vision*. 2006. 428–441.
- [17] Ming-yu Chen, and Alexander Hauptmann. Mosift: Recognizing human actions in surveillance videos. 2009,
- [18] Rizwan Chaudhry, Avinash Ravichandran, Gregory Hager, and René Vidal. *Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions*. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009. 1932–1939.
- [19] Ivan Laptev, and Patrick Pérez. *Retrieving actions in movies*. In: *2007 IEEE 11th International Conference on Computer Vision*. 2007. 1–8.
- [20] Hiroaki Sakoe, and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*. 1978, 26 (1), 43–49.
- [21] Michael Peter Norton, and Denis G Karczub. *Fundamentals of noise and vibration analysis for engineers*. Cambridge university press, 2003.
- [22] Jonathan Foote. *A similarity measure for automatic audio classification*. In: *Proc. AAAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora*. 1997.
- [23] Hyoung-Gook Kim, Nicolas Moreau, and Thomas Sikora. *MPEG-7 audio and beyond: Audio content indexing and retrieval*. John Wiley & Sons, 2006.
- [24] Markus Forsberg. Why is speech recognition difficult. *Chalmers University of Technology*. 2003,
- [25] Climent Nadeu, Dušan Macho, and Javier Hernando. Time and frequency filtering of filter-bank energies for robust HMM speech recognition. *Speech Communication*. 2001, 34 (1-2), 93–114.
- [26] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*. 2011, 20 (1), 30–42.
- [27] Biing Hwang Juang, and Laurence R Rabiner. Hidden Markov models for speech recognition. *Technometrics*. 1991, 33 (3), 251–272.
- [28] Alex Graves, and Navdeep Jaitly. *Towards end-to-end speech recognition with recurrent neural networks*. In: *International conference on machine learning*. 2014. 1764–1772.
- [29] Avery Wang. The Shazam music recognition service. *Communications of the ACM*. 2006, 49 (8), 44–48.
- [30] Eugene Weinstein, and Pedro Moreno. *Music identification with weighted finite-state transducers*. In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*. 2007. II–689.
- [31] Ken Irwin. Musipedia: The open music encyclopedia. *Reference Reviews*. 2008,
- [32] Justin Salamon, and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*. 2012, 20 (6), 1759–1770.

- [33] Tao Liu, Xianglin Huang, Lifang Yang, and Pengju Zhang. *Query by Humming: Comparing Voices to Voices*. In: *2009 International Conference on Management and Service Science*. 2009. 1–4.
- [34] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. Openunmix-a reference implementation for music source separation. *Journal of Open Source Software*. 2019, 4 (41), 1667.
- [35] Lawrence G Roberts. *Machine perception of three-dimensional solids*. 1963.
- [36] Judith MS Prewitt, and others. Object enhancement and extraction. *Picture processing and Psychopictorics*. 1970, 10 (1), 15–19.
- [37] Irwin Sobel, R Duda, and P Hart. Sobel-Feldman Operator.
- [38] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*. 1986, (6), 679–698.
- [39] Rachid Deriche. Using Canny’s criteria to derive a recursively implemented optimal edge detector. *International journal of computer vision*. 1987, 1 (2), 167–187.
- [40] C-H Teh, and Roland T. Chin. On the detection of dominant points on digital curves. *IEEE Transactions on pattern analysis and machine intelligence*. 1989, 11 (8), 859–872.
- [41] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*. 1962, 8 (2), 179–187.
- [42] Jan Flusser, and Tom Suk. Rotation moment invariants for recognition of symmetric objects. *IEEE Transactions on Image Processing*. 2006, 15 (12), 3784–3790.
- [43] David G Lowe. *Object recognition from local scale-invariant features*. In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. 1999. 1150–1157.
- [44] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. *Surf: Speeded up robust features*. In: *European conference on computer vision*. 2006. 404–417.
- [45] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. *ORB: An efficient alternative to SIFT or SURF*. In: *2011 International conference on computer vision*. 2011. 2564–2571.
- [46] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. *Brief: Binary robust independent elementary features*. In: *European conference on computer vision*. 2010. 778–792.
- [47] John Wang, and Edwin Olson. *AprilTag 2: Efficient and robust fiducial detection*. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016. 4193–4198.
- [48] Vahid Kazemi, and Josephine Sullivan. *One millisecond face alignment with an ensemble of regression trees*. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014. 1867–1874.
- [49] Vinícius Fernandes de Sousa, Herman Martins Gomes, and José Eustáquio Rangel de Queiroz. *Comparative evaluation of facial fiducial point detection approaches*. In: *2017 IEEE Visual Communications and Image Processing (VCIP)*. 2017. 1–4.
- [50] Martin A Fischler, and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*. 1981, 24 (6), 381–395.
- [51] Paul Viola, and Michael J Jones. Robust real-time face detection. *International journal of computer vision*. 2004, 57 (2), 137–154.

- [52] Joseph Redmon, and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv*. 2018,
- [53] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014. 580–587.
- [54] Ross Girshick. *Fast r-cnn*. In: *Proceedings of the IEEE international conference on computer vision*. 2015. 1440–1448.
- [55] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster r-cnn: Towards real-time object detection with region proposal networks*. In: *Advances in neural information processing systems*. 2015. 91–99.
- [56] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. *Mask r-cnn*. In: *Proceedings of the IEEE international conference on computer vision*. 2017. 2961–2969.
- [57] Karen Simonyan, and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. 2014,
- [58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep residual learning for image recognition*. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. 770–778.
- [59] Mingxing Tan, and Quoc Le. *Efficientnet: Rethinking model scaling for convolutional neural networks*. In: *International Conference on Machine Learning*. 2019. 6105–6114.
- [60] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and others. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 2020,
- [61] Li-Jia Li, Hao Su, Fei-Fei Li, and Eric P Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. 2010,
- [62] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Scene classification using a hybrid generative/discriminative approach. *IEEE transactions on pattern analysis and machine intelligence*. 2008, 30 (4), 712–727.
- [63] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*. 2015,
- [64] Limin Wang, Sheng Guo, Weilin Huang, Yuanjun Xiong, and Yu Qiao. Knowledge guided disambiguation for large-scale scene classification with multi-resolution CNNs. *IEEE Transactions on Image Processing*. 2017, 26 (4), 2055–2068.
- [65] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2017,
- [66] Ondřej Jamriška, Daniel Sýkora, and Alexander Hornung. *Cache-efficient graph cuts on structured grids*. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012. 3673–3680.
- [67] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-net: Convolutional networks for biomedical image segmentation*. In: *International Conference on Medical image computing and computer-assisted intervention*. 2015. 234–241.



- [68] Abhishek Chaurasia, and Eugenio Culurciello. *Linknet: Exploiting encoder representations for efficient semantic segmentation*. In: *2017 IEEE Visual Communications and Image Processing (VCIP)*. 2017. 1–4.
- [69] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. *Pyramid scene parsing network*. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017. 2881–2890.
- [70] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Zhiyong Gao, and Ming-Ting Sun. *Deep kalman filtering network for video compression artifact reduction*. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018. 568–584.
- [71] Ray Smith. *A simple and efficient skew detection algorithm via text row accumulation*. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. 1995. 1145–1148.
- [72] Ray Smith. *An overview of the Tesseract OCR engine*. In: *Ninth international conference on document analysis and recognition (ICDAR 2007)*. 2007. 629–633.
- [73] Bindu Reddy, and Anita Jadhav. *Comparison of scene change detection algorithms for videos*. In: *2015 Fifth International Conference on Advanced Computing & Communication Technologies*. 2015. 84–89.
- [74] Emmanuel Veneau, Rémi Ronfard, and Patrick Bouthemy. *From video shot clustering to sequence segmentation*. In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. 2000. 254–257.
- [75] Costas Cotsaces, Nikos Nikolaidis, and Ioannis Pitas. Video shot detection and condensed representation. a review. *IEEE signal processing magazine*. 2006, 23 (2), 28–37.
- [76] Brandon Castellano. *PySceneDetect*. <https://github.com/Breakthrough/PySceneDetect>. 2017.
- [77] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3D video stabilization. *ACM Transactions on Graphics (ToG)*. 2009, 28 (3), 1–9.
- [78] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *IEEE signal processing magazine*. 2003, 20 (3), 21–36.
- [79] Heng Wang, and Cordelia Schmid. *Action recognition with improved trajectories*. In: *Proceedings of the IEEE international conference on computer vision*. 2013. 3551–3558.
- [80] H. Lim, J. Lim, and H. J. Kim. *Real-time 6-DOF monocular visual SLAM in a large-scale environment*. In: *ICRA*. 2014. 1532–1539.
- [81] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. *KinectFusion: Real-time dense surface mapping and tracking*. In: *ISMAR*. 2011. 127–136.
- [82] Jakob Engel, Thomas Schöps, and Daniel Cremers. *LSD-SLAM: Large-scale direct monocular SLAM*. In: *ECCV*. 2014. 834–849.
- [83] Eagle S Jones, and Stefano Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *IJRR*. 2011, 30 (4), 407–430.

- [84] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual-inertial odometry using nonlinear optimization. *IJRR*. 2015, 34 (3), 314–334.
- [85] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*. 1981, 293 (5828), 133.
- [86] David Nistér. An efficient solution to the five-point relative pose problem. *TPAMI*. 2004, 26 (6), 756–770.
- [87] Tobias Weyand, and Bastian Leibe. Visual landmark recognition from internet photo collections: A large-scale evaluation. *Computer Vision and Image Understanding*. 2015, 135 1–15.
- [88] Meinard Müller. *Information retrieval for music and motion*. Springer, 2007.
- [89] Daniel Keysers, Thomas Deselaers, Henry A Rowley, Li-Lun Wang, and Victor Carbune. Multi-language online handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*. 2016, 39 (6), 1180–1194.
- [90] David Ha, and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*. 2017,
- [91] James Ze Wang, Gio Wiederhold, Oscar Firschein, and Sha Xin Wei. *Wavelet-based image indexing techniques with partial sketch retrieval capability*. In: *Proceedings of ADL'97 Forum on Research and Technology. Advances in Digital Libraries*. 1997. 13–24.
- [92] Xiaoou Tang, and Xiaogang Wang. *Face sketch synthesis and recognition*. In: *Proceedings Ninth IEEE International Conference on Computer Vision*. 2003. 687–694.
- [93] Andreas Leibetseder, Sabrina Kletz, and Klaus Schoeffmann. *Sketch-based similarity search for collaborative feature maps*. In: *International Conference on Multimedia Modeling*. 2018. 425–430.
- [94] Philip A Legg, David HS Chung, Matthew L Parry, Rhodri Bown, Mark W Jones, Iwan W Griffiths, and Min Chen. Transformation of an uncertain video search pipeline to a sketch-based visual analytics loop. *IEEE transactions on Visualization and Computer Graphics*. 2013, 19 (12), 2109–2118.
- [95] S-S Cheung, and Avideh Zakhor. Efficient video similarity measurement with video signature. *IEEE Transactions on Circuits and Systems for video Technology*. 2003, 13 (1), 59–74.
- [96] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. *GroupLens: An open architecture for collaborative filtering of netnews*. In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 1994. 175–186.
- [97] Benjamin Satzger, Markus Endres, and Werner Kießling. *A preference-based recommender system*. In: *International Conference on Electronic Commerce and Web Technologies*. 2006. 31–40.
- [98] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and others. *The YouTube video recommendation system*. In: *Proceedings of the fourth ACM conference on Recommender systems*. 2010. 293–296.
- [99] F Maxwell Harper, and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*. 2015, 5 (4), 1–19.

- [100] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*. 2009, 42 (8), 30–37.
- [101] Carl-Herman Hjortsjö. *Man's face and mimic language*. Studentlitteratur, 1969.
- [102] Soujanya Poria, Erik Cambria, Amir Hussain, and Guang-Bin Huang. Towards an intelligent framework for multimodal affective data analysis. *Neural Networks*. 2015, 63 104–116.
- [103] Jacob Perkins. *Python text processing with NLTK 2.0 cookbook*. Packt Publishing Ltd, 2010.
- [104] Clayton J Hutto, and Eric Gilbert. *Vader: A parsimonious rule-based model for sentiment analysis of social media text*. In: *Eighth International AAAI Conference on Weblogs and Social Media*. 2014.
- [105] Ludmila I Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014.
- [106] Zhen-zhong Lan, Lei Bao, Shoou-I Yu, Wei Liu, and Alexander G Hauptmann. *Double fusion for multimedia event detection*. In: *International Conference on Multimedia Modeling*. 2012. 173–185.
- [107] Gilles De Mey. *Google Speech API v2*. 2014.  
<https://github.com/gillesdemey/google-speech-v2>.
- [108] Kevin Atkinson. *Gnu aspell 0.60. 4*. 2006.
- [109] Martin F Porter. *Snowball: A language for stemming algorithms*. 2001.
- [110] Pavel Brazdil, Christophe Giraud-Carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. ISBN 978-3-540-73262-4.  
<http://link.springer.com/10.1007/978-3-540-73263-1>.
- [111] Navid Serrano, Andreas Savakis, and A Luo. *A computationally efficient approach to indoor/outdoor scene classification*. In: *Object recognition supported by user interaction for service robots*. 2002. 146–149.
- [112] Kamal M Othman, and Ahmad B Rad. An indoor room classification system for social robots via integration of CNN and ECOC. *Applied Sciences*. 2019, 9 (3), 470.
- [113] Sudeep Pillai, and John Leonard. Monocular slam supported object recognition. *arXiv preprint arXiv:1506.01732*. 2015,
- [114] Kalyan Sunkavalli, Fabiano Romeiro, Wojciech Matusik, Todd Zickler, and Hanspeter Pfister. *What do color changes reveal about an outdoor scene?*. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008. 1–8.
- [115] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*. 1982, 43 (1), 59–69.
- [116] Teuvo Kohonen. Fast evolutionary learning with batch-type self-organizing maps. *Neural Processing Letters*. 1999, 9 (2), 153–162.
- [117] Robert Neumayer, Michael Dittenbach, and Andreas Rauber. *Playsom and pocketsomplayer, alternative interfaces to large music collections*. In: *ISMIR*. 2005. 618–623.
- [118] Pitoyo Hartono, and Ryo Yoshitake. Automatic playlist generation from self-organizing music map. *Journal of Signal Processing*. 2013, 17 (1), 11–19.
- [119] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.

- [120] J Ross Quinlan. C4. 5: Programming for machine learning. *Morgan Kauffmann*. 1993, 38 (48), 49.
- [121] Roberta Siciliano, and Francesco Mola. Multivariate data analysis and modeling through classification and regression trees. *Computational Statistics & Data Analysis*. 2000, 32 (3-4), 285–301.
- [122] Robert Andrews, Joachim Diederich, and Alan B Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*. 1995, 8 (6), 373–389.
- [123] Joachim Diederich, Alan B Tickle, and Shlomo Geva. *Quo vadis? Reliable and practical rule extraction from neural networks*. 2010.
- [124] Sushmita Mitra, and Yoichi Hayashi. Neuro-fuzzy rule generation: survey in soft computing framework. *IEEE transactions on neural networks*. 2000, 11 (3), 748–768.
- [125] Alan B Tickle, Robert Andrews, Mostefa Golea, and Joachim Diederich. The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Transactions on Neural Networks*. 1998, 9 (6), 1057–1068.
- [126] Artur SD’Avila Garcez, Luis C Lamb, and Dov M Gabbay. *Neural-symbolic cognitive reasoning*. Springer Science & Business Media, 2008.
- [127] Gregor PJ Schmitz, Chris Aldrich, and Francois S Gouws. ANN-DT: an algorithm for extraction of decision trees from artificial neural networks. *IEEE Transactions on Neural Networks*. 1999, 10 (6), 1392–1401.
- [128] Emad W Saad, and Donald C Wunsch II. Neural network explanation using inversion. *Neural networks*. 2007, 20 (1), 78–93.
- [129] Jan Zilke. Extracting rules from deep neural networks. 2015,
- [130] Jan Brejcha, and Martin Čadík. State-of-the-art in visual geo-localization. *Pattern Analysis and Applications*. 2017, 20 (3), 613–637.
- [131] Liang-Hua Chen, Yu-Chun Lai, and Hong-Yuan Mark Liao. Movie scene segmentation using background information. *Pattern Recognition*. 2008, 41 (3), 1056–1065.
- [132] Jianping Fan, Ahmed K Elmagarmid, Xingquan Zhu, Walid G Aref, and Lide Wu. ClassView: hierarchical video shot classification, indexing, and accessing. *IEEE Transactions on Multimedia*. 2004, 6 (1), 70–86.
- [133] Diederik P Kingma, and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 2014,
- [134] Yichuan Tang. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*. 2013,
- [135] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*. 2012, 25
- [136] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. *Imagenet: A large-scale hierarchical image database*. In: *2009 IEEE conference on computer vision and pattern recognition*. 2009. 248–255.
- [137] John A Marchant, and Christine M Onyango. Shadow-invariant classification for scenes illuminated by daylight. *JOSA A*. 2000, 17 (11), 1952–1961.
- [138] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. *High-resolution image inpainting using multi-scale neural patch synthesis*. In: *Proceed-*

- ings of the *IEEE conference on computer vision and pattern recognition*. 2017. 6721–6729.
- [139] Bruce D Lucas, Takeo Kanade, and others. *An iterative image registration technique with an application to stereo vision*. In: 1981.
- [140] Jianbo Shi, and others. *Good features to track*. In: *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. 1994. 593–600.
- [141] Marius Muja, and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration.. *VISAPP (1)*. 2009, 2 (331-340), 2.
- [142] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*. 1975, 18 (9), 509–517.
- [143] J. L. Blanco-Claraco. *nanoflann*.  
<https://github.com/jlblancoc/nanoflann>.
- [144] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A Benchmark for Multi-Object Tracking. *arXiv:1603.00831 [cs]*. 2016,
- [145] Andrea Colombari, Andrea Fusiello, and Vittorio Murino. Segmentation and tracking of multiple video objects. *Pattern Recognition*. 2007, 40 (4), 1307–1317.
- [146] Shu Wang, Huchuan Lu, Fan Yang, and Ming-Hsuan Yang. *Superpixel tracking*. In: *2011 International Conference on Computer Vision*. 2011. 1323–1330.
- [147] Zongpu Zhang, Yang Hua, Tao Song, Zhengui Xue, Ruhui Ma, Neil Robertson, and Haibing Guan. *Tracking-assisted Weakly Supervised Online Visual Object Segmentation in Unconstrained Videos*. In: *Proceedings of the 26th ACM international conference on Multimedia*. 2018. 941–949.
- [148] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. *Learning spatiotemporal features with 3d convolutional networks*. In: *Proceedings of the IEEE international conference on computer vision*. 2015. 4489–4497.
- [149] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. *Microsoft COCO: Common objects in context*. In: *European conference on computer vision*. 2014. 740–755.
- [150] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. *CoRR*. 2019, abs/1905.04804
- [151] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*. 2018,
- [152] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking. *arXiv preprint arXiv:2004.01888*. 2020,
- [153] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*. 1955, 2 (1-2), 83–97.
- [154] Liang Zheng, Zhi Bie, Yifan Sun, Jingdong Wang, Chi Su, Shengjin Wang, and Qi Tian. *Mars: A video benchmark for large-scale person re-identification*. In: *European Conference on Computer Vision*. 2016. 868–884.
- [155] Nicolai Wojke, and Alex Bewley. *Deep cosine metric learning for person re-identification*. In: *2018 IEEE winter conference on applications of computer vision (WACV)*. 2018. 748–756.

**Reviewed Publications of the Author Relevant to the Thesis**

- [A1] Lukáš Korel, Petr Pulc, Jiří Tumpach, and Martin Holeňa. *Video Scene Location Recognition with Neural Networks*. In: *Proceedings of the 21th Conference Information Technologies - Applications and Theory (ITAT 2021)*, 2021. ISSN 1613-0073.
- [A2] Petr Pulc, and Martin Holeňa. *Unsupervised Construction of Task-Specific Datasets for Object Re-identification*. In: *ICCTA 2021 Conference Proceedings*. New York: Association for Computing Machinery, 2021. ISBN 978-1-4503-9052-1.
- [A3] Matěj Fanta, Petr Pulc, and Martin Holeňa. *Rules extraction from neural networks trained on multimedia data*. In: *Proceedings of the 19th Conference Information Technologies - Applications and Theory (ITAT 2019)*, 2019. ISSN 1613-0073.
- [A4] Petr Pulc, and Martin Holeňa. *Hierarchical Motion Tracking Using Matching of Sparse Features*. In: *Proceedings of the 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, 2018. ISBN 978-1-5386-9385-8.
- [A5] Petr Pulc, Oliver Kerul-Kmec, Tomáš Šabata, and Martin Holeňa. *Motion Segmentation by Semi-Supervised Classification in Dynamic Scenery*. In: *Proceedings of Poster Session of 3rd ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data (AALTD 2018)*, 2018.
- [A6] Tomáš Šabata, Petr Pulc, and Martin Holeňa. *Semi-supervised and Active Learning in Video Scene Classification from Statistical Features*. In: *Proceedings of the Workshop on Interactive Adaptive Learning (IAL 2018) co-located with European Conference on Machine Learning (ECML 2018) and Principles and Practice of Knowledge Discovery in Databases (PKDD 2018)*, 2018. ISSN 1613-0073.
- [A7] Oliver Kerul-Kmec, Petr Pulc, and Martin Holeňa. *Semisupervised segmentation of UHD video*. In: *Proceedings of the 18th Conference Information Technologies - Applications and Theory (ITAT 2018)*, 2018. ISSN 1613-0073. ISBN 978-1-7272-6719-8.
- [A8] Jiří Kožusznik, Petr Pulc, and Martin Holeňa. *Sentiment analysis from utterances*. In: *Proceedings of the 18th Conference Information Technologies - Applications and Theory (ITAT 2018)*, 2018. ISSN 1613-0073. ISBN 978-1-7272-6719-8.
- [A9] Petr Pulc, and Martin Holeňa. *Towards Real-time Motion Estimation in High-Definition Video Based on Points of Interest*. In: *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems*, 2017. ISSN 2300-5963. ISBN 978-83-946253-7-5.
- [A10] Petr Pulc, and Martin Holeňa. *Application of Meta-learning Principles in Multimedia Indexing*. In: *DATESO 2016: Databases, Texts, Specifications, and Objects*, 2016. ISBN 978-80-248-4031-4.
- [A11] Petr Pulc, Eric Rosenzweig, and Martin Holeňa. *Image Processing in Collaborative Open Narrative Systems*. In: *ITAT 2016: Information Technologies - Applications and Theory: Conference on Theory and Practice of Information Technologies*, 2016. ISSN 1613-0073.
- [A12] Michal Kopp, Petr Pulc, and Martin Holeňa. *Search for Structure in Audiovisual Recordings of Lectures and Conferences*. In: *ITAT 2015 conference proceedings*, 2015. ISSN 1613-0073. ISBN 978-1-5151-2065-0.

- 
- [A13] Petr Pulc, and Martin Holeňa. *Case Study in Approaches to the Classification of Audiovisual Recordings of Lectures and Conferences*. In: *Proceedings of the 14th conference ITAT 2014 – Workshops and Posters*, 2014. ISBN 978-80-87136-19-5.

#### ■ **Diploma Theses Supervised by the Author Relevant to the Thesis**

- [B1] Jan Kostecký. *Identification of dams and embankments*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.
- [B2] Ondřej Pírko. *Digital elevation model refinement from satellite imagery*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.
- [B3] Richard Burkoň. *Slide and animation detection in screen capture*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.
- [B4] Josef Rypáček. *Spherical panorama crawler and viewer*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.
- [B5] Tomáš Dejmek. *Dancing style recommender*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.
- [B6] Petr Kubín. *YouTube Connector for NARRA*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2015.
- [B7] Dmitry Vanyagin. *Adobe Premiere Pro Plugin for NARRA*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2015.

#### ■ **Remaining Publications of the Author Relevant to the Thesis**

- [C1] Martin Holeňa, Petr Pulc, and Martin Kopp. *Classification Methods for Internet Applications*. Springer, 2020. ISSN 2197-6503. ISBN 978-3-030-36961-3.





# Appendix A

## Week of Science and Technology Lecture Dataset

The data for our experiments on efficient approaches to knowledge extraction have been acquired on the Week of Science and Technology. This festival includes many lectures on various topics, and thus one of the targets was to extract topic labels on the level of individual slides.

The lectures were recorded by MediaSite device, which enables capture and streaming of audiovisual signal with separate capture of currently shown slides as still images. To this end, the copy of a signal to the projector is processed such that a major difference in the incoming image triggers the capture of a new slide. This way, we also gather precise information on presentation timing.

The file structure of a lecture recorded by MediaSite is as follows:

<code>MediasitePresentation_50.xml</code>	metadata, including slide timing.
<code>Player.html</code>	HTML page with player, not relevant.
<code>App_Themes</code>	the directory of helper files for player, not relevant.
<code>PlayerOptions</code>	the directory of player branding, not relevant.
<code>Players</code>	the directory of player itself, not relevant.
<code>Content/</code>	the directory of multimedia content.
<code>&lt;unique_hash&gt;.wmv</code>	audiovisual recording.
<code>slide_&lt;nnnn&gt;_full.jpg</code>	full slide image.
<code>thumbnail_&lt;unique_hash&gt;.jpg</code>	thumbnail image.

In our pre-processing stage, the `MediasitePresentation_50.xml` file is decoded and timing information is gathered. With the timing information, we cut the video recording in file `<unique_hash>.wmv` into sections that correspond to individual slides for later processing.

Sound is extracted from the multimedia file and cut to fragments shorter than 10 seconds, optimised such that the cut is in the places with the lowest loudness. This is due to the limitation of the Google Speech API that we use for the automated speech recognition. Transcribed audio is also assigned to the individual slides; overhanging pieces of audio are assigned to both slides.

Pictures of slides are thresholded for ease of letter boundary recognition and passed to system Tesseract for optical character recognition. Acquired words are filtered against Aspell dictionary, stemmed and removed of stop words.

Visual information from the camera is averaged into a color histogram, accompanied by visual features extracted by the SURF algorithm.

The resulting information are stored in a JSON file for later processing.

Structure of a JSON file describing all the extracted features.

```

{
  title,                Name of the lecture
  description,          Textual description
  duration,             Duration in ms
  slides: [             Set of slides
    number,             Number of the slide
    start,              Starting time
    end,                Ending time
    ocr,                Text recognized by OCR
    ocr_stemm,          Stemmed version of OCR text
    ocr_clean,          OCR stemms cleaned by Aspell
    audio: [           Set of audio snippets
      result: [        Transcription result
        alternative: [ Set of alternatives
          transcript,  Actual transcript
          confidence   Confidence of the result
        ],
        final          If the transcription is final
      ],
      result_index     Which alternative is preferred
    ],
    audio_stemm: [],    Set of stemmed transcriptions
    audio_stemm_alter: [], Set of stemmed alternatives
    histogram: [],     Histogram of the video, 256 * RGB
    slide_histogram: [], Histogram of the slide, 256 * RGB
    surf               SURFs detected in the video frame
  ]
}

```

## Appendix B

### The Big Bang Theory Scene Dataset

We created a custom dataset from the popular TV series The Big Bang Theory for our experiment on scene recognition. The primary motivation for using this particular series is that it is a long-running series, the sets have not changed significantly over many years, but at the same time, the visual appearance of the individual sections of the set is relatively diverse, and also the set decorations allow for many shooting angles.

Dataset is structured by the series and episode. Each episode contains the following metadata files:

- `splits.p` list of frame numbers with detected scene change.
- `frame_count.p` number of frames.
- `sequences.p` list of continuous content sequences as list of scene indexes.

Each scene is then referred to by a zero-prefixed index *nnn* and following files are stored:

- `nnn.jpg` a representative frame from the scene.
- `nnn.p` information on scene key, shot size and number of visible faces.
- `nnn_h.p` global hue histogram.
- `nnn_s.p` global saturation histogram.
- `nnn_v.p` global value histogram.
- `nnn_h_bg.p`
- `nnn_s_bg.p`
- `nnn_v_bg.p` global histograms with ignored section from the centre of the frame modeled as a disc with diameter equal to the half of the frame height.
- `nnn_rgb_8.p` flattened RGB histogram with 8 bins per channel.
- `nnn_rgb_16.p` flattened RGB histogram with 16 bins per channel.
- `nnn_hsv_30.p` flattened HSV histogram with 30 H bins and 8 S and V bins.
- `nnn_hsv_30_4_4.p` flattened HSV histogram with 30 H bins and 4 S and V bins.
- `nnn_hsv_30_4_4_2x2.p` the same as above, counted separately on four image quadrants.
- `nnn_hsv_20_4_4_2x2.p` the same as above, with reduced number of H bins to 20.