Extractie en representatie van semantische informatie in digitale media

Extraction and Representation of Semantic Information in Digital Media

Gaëtan Martens

Promotoren: prof. dr. ir. R. Van de Walle, prof. dr. M. Leman
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Computerwetenschappen

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. J. Van Campenhout
Faculteit Ingenieurswetenschappen
Academiejaar 2010 - 2011

UNIVERSITEIT
GENT

ISBN 978-90-8578-396-1
NUR 980, 983
Wettelijk depot: D/2011/10.500/1

There is no reality, there is only perception.

Dankwoord

Aan de totstandkoming van dit proefschrift gaat een lange tewerkstellingsperiode aan Universiteit Gent vooraf. Tijdens deze periode heb ik de kans gekregen om mij te verdiepen in veel en vaak uiteenlopende domeinen. Dit is dus voor mij een enorm leerrijke periode geweest, zowel op professioneel als op persoonlijk vlak. Daarom zijn er een hele hoop mensen die ik wil bedanken voor de bijdrage die zij geleverd hebben.

In de eerste plaats wil ik uitdrukkelijk mijn promotoren prof. dr. ir. Rik Van de Walle en prof. dr. Marc Leman loven. Marc, zonder jouw initiatief zou ik nooit aan de Universiteit Gent begonnen met werken zijn. Behoren tot het IPEM is op veel vlakken een leerrijke ervaring geweest die me voor de rest van mijn leven zal bijblijven. Rik, bedankt dat ik van jou de kans én het vertrouwen kreeg om uiteindelijk tot dit werk te komen. Het was een hele eer om deel uit te maken van Multimedia Lab.

Ook prof. dr. Hans De Meyer wil ik in het bijzonder bedanken, voor de samenwerking op MAMI, het luisterend oor, de wijze raad en natuurlijk niet te vergeten de vele gezellige gesprekken. “Maar de volgende keer trakteer ik wel hoor!” Ook dr. Chris Poppe verdient een speciaal dankwoordje, niet alleen voor de vele feedback die ik van hem gekregen heb en voor het werk dat we samen verricht hebben, maar ook voor de vele aangename conversaties.

Een bijzonder woord van dank gaat ook naar Erik Mannens, voor de opvolging in de projecten, het vertrouwen en de onvoorwaardelijke steun. Ook dr. Peter Lambert, Steven Verstockt en Sam Coppens wil ik nog eens danken voor hun feedback en de interessante gesprekken, alsook Ellen Lammens voor de administratieve ondersteuning. Maar ook mijn andere ex-collega’s van Multimedia Lab wil ik hier nog eens extra danken voor de vier leerrijke en leuke jaren samen met jullie... wat zijn ze voorbijgevlogen!

Ook de ex-collega’s van het IPEM vergeet ik uiteraard niet. Koen Tanghe wil ik langs deze weg nog eens speciaal bedanken voor de leerrijke samenwerk-

ing en de vele C++ tips. Ook dr. Jelle Dierickx, Olmo Cornelis, dr. Micheline Lesaffre, dr. Dirk Moelants en Ivan Schepers wil ik nog eens bedanken voor de nuttige commentaren, inspirerende discussies en de vele onvergetelijke momenten die we samen beleefd hebben. Voor de vele feedback tijdens het MAMI-project wens ik ook nog prof. dr. Bernard De Baets en prof. dr. ir. Jean-Pierre Martens te bedanken.

Mijn vrienden en familieleden wil ik ook nog loven voor de vele steun over de voorbije jaren. Nu dit doctoraatswerk afgerond is, hoop ik wat meer tijd met hen te kunnen doorbrengen. Rosa, ik was zo overtuigd dat jij het resultaat van mijn onderzoek nog zou meemaken. Je hebt tot op het laatste moment in me geloofd. Daarom draag ik dit werk dan ook aan jou op.

Ten slotte, maar zeker niet het minst, wil ik Wendy bedanken. Wendy, je hebt me al die jaren onvoorwaardelijk gesteund en ik kon altijd op jou rekenen. Nu dit proefschrift afgewerkt is, kijk ik samen met jou en onze flinke zoon Yben uit naar de toekomst die voor ons ligt, en waar we samen verder iets moois zullen van maken.

Gaëtan Martens
3 september 2010

Summary

Over the last decades, the development of digitization techniques and the increasing capacity of storage media together with decreasing storage costs has led to an explosion of digital content. Huge amounts of audio, images, and videos are generated daily and are mostly stored in an unstructured repository of multimedia information, much of which can be accessed through the Internet. There will be approximately 1.8 zettabytes of data in 2011. The biggest growth in data is visual in nature, from devices such as digital cameras, digital surveillance cameras, and digital televisions. Further, the music sector is generating far greater value from the online and mobile market than any other sector in the creative industries, with the exception of electronic games. For example, Apple's iTunes Store offers its customers over ten billion songs to choose from and the file sharing networks have billions of tracks available for download.

To deal with these huge amounts of data, one of the main imperatives IT organizations already face, are new tools and standards for data search and analysis. In particular, there is a need for efficient techniques that are able to extract information directly from the content. Indeed, with the growth of digital content, the need for *content-based* retrieval techniques has drastically increased. Further, approximately 70% of all data are created by individuals. As a result, the need for representative data models for the management and disclosure of personal data becomes higher. Today, one's personal content is stored on many devices, such as a smart phone, an MP3-player, a laptop, or somewhere on the World Wide Web. As a result, it becomes more difficult to efficiently manage these personal data and the related metadata. Therefore, a system to manage these data is needed that, transparently, keeps track of the data sources, allows (both manually or automatically obtained) annotations, preserves the semantics, and is interoperable with other metadata standards so that exchanging metadata is facilitated.

However, within the domain of content-based information retrieval, there is a major discrepancy: the semantic gap. The semantic gap states that a user wants to retrieve data on a semantic level, but the characterizations can only provide a low-level similarity. Over the last years, a plethora of approaches have been developed to bridge the semantic gap. These attempts mainly apply two kinds of strategies: a bottom-up or a top-down strategy. The bottom-up approach starts from extracting features at the signal level (low level) and then applies machine learning techniques for concept detection (high level). Contrary, the top-down approach starts from applying domain knowledge to gain insight into the compositional elements which are then refined in greater detail.

In any content-based retrieval application, a *good* content descriptor (or feature) is essential. In this context, good also means that the descriptor should be invariant to the accidental variance introduced by the content creation process. However, there is a trade-off between this invariance and the discriminative power of features since a very wide class of invariance loses the ability to discriminate between essential differences.

The work presented in this dissertation proposes methods to bridge the semantic gap in the computer vision and the musical audio mining domain. We propose features that are related with human perception and methods relying on machine learning techniques to relate these features with concepts. Next, we present a system to model, disclose, and manage different types of metadata. Furthermore, the interoperability with metadata standards is also tackled.

In the computer vision domain, we focus on the modeling of textures. Texture is an important property in computer vision and it plays an important role in many image analysis applications since it is more robust to lighting conditions than, e.g., color. Furthermore, it can play an important role in image understanding since texture information can often be directly related with the depicted concepts. However, there is no formal definition of texture, and consequently, it can be modeled in a variety of ways. Since texture is related with human perception, we propose a novel texture feature based on outputs of cells that are found in the visual cortex of humans. These features can be successfully used for different applications. At first, we show that for unsupervised segmentation of textured images, the proposed features obtain the highest accuracy. Secondly, for supervised texture classification, we show that our features can compete with state-of-the-art texture analysis methods. However, for most practical applications, the presence of noise in images is a problem in texture characterization and causes difficulties for interpretation. On the third part, we conduct classification experiments with various amounts of noise. We have shown that the presented approach is highly robust to various levels of uniform, speckle, and Gaussian noise. Also by applying JPEG image compres-

sion, we have shown that the presented texture features are highly robust and can deal better with the impact on the high spatial-frequencies of the textures than other texture analysis methods. Fourth, we have also examined the use of texture features for material classification. The presented texture features also achieve the highest classification rates in our material classification tests. Finally, the presented texture features are used for the interpretation of textured outdoor scenery images by applying segmentation, followed by material detection on the obtained image regions, and the application of domain knowledge. The experiments point out that the use of perceptually based texture information is an added value for scene interpretation since these low-level features can be related to semantic concepts. To model cognition, we have ingested domain knowledge by applying a top-down approach. An ontology describing the conditions and restrictions of the depicted concepts, can tackle many errors which are related to a bottom-up approach.

In the musical audio mining domain, we present a bottom-up approach to extract the melody and tonality from musical audio signals. Melody and tonality are main characteristics of Western music, and are related to pitch perception. It is a fact that people often remember the melody of a song rather than the performer and the title. Even after the lyrics have been forgotten, it is the melody that humans are likely to recall or are able to reproduce by humming or whistling. Despite the fact that after hearing a song, listeners are able to reproduce the melody, automatic melody transcription is not an easy task. Moreover, as music is nowadays typically produced by different, simultaneously playing instruments, melody transcription becomes more complex. The latter is because frequencies of concurrent sounds may coincide. In this dissertation, we propose a transcription system to automatically extract the melody from – both monophonic and polyphonic – musical audio signals. We aim to distinguish individual musical notes characterized by specific temporal boundaries in order to obtain a symbolic representation. Our proposed melody transcription system consists of four stages. In the first stage, we used an auditory model to obtain the necessary pitch information for the later processing stages. In the second stage, we estimated the fundamental frequency in individual time frames, and we created pitch trajectories over adjacent time frames. In the third stage, the obtained trajectories are transformed into tones so that a note representation can be easily deduced. Finally, in the fourth and last stage, post-processing which removes low-salience tones or too short tones, is applied to obtain the melody.

Next to the extraction of melodic information, we also considered the extraction and classification of tonality information. Like melody, tonality also en-

tails pitch information, but it manifests at a relatively larger time interval than melody. Tonality is a harmonic system of 24 keys (i.e., 12 major and 12 minor keys). Both major and minor keys can be placed on a circle of fifths which models their relationships. The extraction of tonality from musical audio signals is based on pitch induction which transforms a musical audio signal into high-dimensional pitch patterns. However, if the low-dimensionality of tonality is exploited, we obtain different assignments relying on the metric-based approach. Therefore, we have presented a tree-based approach which overcomes this shortcoming. Furthermore, the tree-based method offers a way of regrouping keys into larger key clusters by ascending one (or more) level(s) in the tree.

Finally, our third research area deals with the representation of semantic metadata. In order to efficiently manage personal data and the related metadata, a content management system (CMS) is needed. Besides a set of procedures to manage the work flow, a CMS typically involves a metadata model. Within the context of a single CMS, the semantics of different fields of the metadata model are well-defined. However, there is a major discrepancy when it comes to the interoperability between different metadata models. Therefore, it is indispensable that the semantics can be formally defined by the metadata model. In this dissertation, we have presented a solution for this problem by introducing a CMS to manage one's personal content that is stored on differed devices. The metadata model of the proposed CMS relies on an ontology that is created using Semantic Web technologies. This ontology is an OWL DL ontology as reasoning needs to be supported. We have outlined how some common difficulties, with respect to ontology modeling, can be tackled and what the deficiencies of the OWL DL language are. The presented ontology for the CMS consists of three major parts that describe system-, security-, and user-centric metadata. The system-centric metadata concern the storage location and how these data can be accessed. The security-centric metadata describe the rights that end-users have concerning a document and its metadata. Besides intellectual property rights, the user-centric metadata include content-descriptive metadata which concern the management of both extracted features and manually added descriptions. These metadata are content-aware since different types of media (such as images, video, music, or text) require specific properties. We have elaborated on image metadata and we have presented an image ontology that is enclosed by the metadata model of the CMS. The image ontology is a rich ontology that supports many concepts for detailed image annotation, and is extended with a taxonomy that refines additional concepts and properties for annotation. An advantage of the application of Semantic Web technologies is to tackle interoperability issues between different meta-

data standards. This issue can be solved by creating semantic representations of metadata standards and a mapping between the resulting ontologies. This way, the proposed image ontology is used as an upper-ontology to relate concepts from other image metadata standards.

Samenvatting

De laatste decennia is er een enorme toename geweest in de hoeveelheid digitale media. De ontwikkeling van algoritmes en technologieën voor het digitaliseren van media alsook de alsmaar stijgende capaciteit en dalende kostprijs van opslagmedia kunnen deze toename verklaren. Grote hoeveelheden audio, beeld en video worden dagelijks geproduceerd. Zo zal er tegen 2011 maar liefst 1.8 zettabytes aan digitale data zijn. De grootste toename in de laatste jaren betreft visuele data, o.a. afkomstig van digitale fotocamera's, bewakingsbeelden en digitale televisie, en zal ook de komende jaren blijven toenemen. Ook de onlinemarkt zal blijven groeien. Zo is bijvoorbeeld de online muziekverkoop, op de game-industrie na, nu al de grootste onlinemarkt en bovendien zal vanaf 2011 het overgrote deel van de muziekverkoop via internet plaatsvinden. Om een beeld van de grootteorde te geven, Apple iTunes biedt nu reeds meer dan 10 miljard nummers aan en file-sharing netwerken hebben ook nog eens miljarden nummers voor het downloaden ter beschikking.

Veel organisaties en bedrijven hebben bijgevolg nood aan applicaties en standaarden om met deze enorme hoeveelheden data om te kunnen gaan. Er is in het bijzonder een grote noodzaak aan technieken om informatie rechtstreeks uit de inhoud van de data te extraheren zodoende deze te kunnen ontsluiten. Bovendien is ongeveer 70% van alle digitale data geproduceerd door individuen. Bijgevolg is er ook nood aan representatieve datamodellen voor het beheren en ontsluiten van persoonlijke data. We kunnen bovendien bemerken dat persoonlijke data tegenwoordig bewaard wordt op verschillende toestellen, zoals bijvoorbeeld op een mobiele telefoon, een MP3-speler, een laptop of gewoon op het World Wide Web. Doordat digitale eigendom dus op verschillende toestellen bewaard wordt, wordt het moeilijker om deze data en haar metadata te beheren. Daarom is er nood aan contentmanagementsystemen die, transparant, met zulke opslagvormen overweg kunnen en ondersteuning bieden voor (zowel manuele als automatisch verkregen) annotaties ter onsluiting. Bovendien dient semantische interoperabiteit ten opzichte van

andere metadatastandaarden gewaarborgd te worden zodoende de uitwisseling van metadata te vergemakkelijken.

Binnen het domein van inhoudsgebaseerde ontsluitingstechnieken is echter een belangrijk obstakel: de semantische kloof. De semantische kloof duidt aan dat eindgebruikers data willen bevragen op een semantisch niveau terwijl de beschrijvende kenmerken die uit de inhoud kunnen geëxtraheerd worden enkel een similariteit op een laag niveau kunnen waarborgen. Bijgevolg is er de voorbije jaren veel onderzoek gebeurd naar methodes om deze semantische kloof te overbruggen. Deze methodes passen hoofdzakelijk twee strategieën toe: een *bottom-up* of een *top-down* strategie. In de bottom-up strategie worden beschrijvende kenmerken uit het signaal (laagste niveau) geëxtraheerd en daarna worden machine learning technieken gebruikt om zodoende bepaalde concepten (hoogste niveau) te detecteren. De top-down methodologie gaat net andersom te werk: er wordt domeinkennis gebruikt om inzicht te vergaren in de componenten van een concept en, in een volgende iteratie, worden deze componenten dan weer verfijnd.

Een *goed* beschrijvend kenmerk is essentieel voor elk inhoudsgebaseerd ontsluitingssysteem. *Goed* betekent in deze context dat het kenmerk voldoende robuust moet zijn voor bepaalde variaties die geïntroduceerd kunnen worden door het creatieproces. Maar een grote invariantie betekent vaak ook dat een groot deel van het onderscheidend vermogen van een bepaalde kenmerk verloren gaat.

Dit proefschrift introduceert een aantal methodes om de semantische kloof in het computervisiedomein en muziekverwerkingsdomein te overbruggen. We stellen kenmerken voor die gerelateerd zijn aan perceptie en we gebruiken machine learning technieken om deze te relateren met bepaalde concepten. Daarnaast stellen we een systeem voor om verschillende types metadata te modelleren, te ontsluiten en te beheren. Daarnaast wordt ook de interoperabiliteit van het systeem met andere metadatastandaarden aangepakt.

In het computervisiedomein hebben we ons gefocust op het modelleren van textuur. Textuur is een belangrijke eigenschap in computervisie aangezien het robuuster is dan bijvoorbeeld kleur ten opzichte van veranderingen in de belichting. Textuur kan bovendien een belangrijke rol spelen in beeldherkenning aangezien het gerelateerd is aan perceptie en bovendien vaak rechtstreeks gerelateerd kan worden aan bepaalde afgebeelde concepten. Maar er bestaat geen formele definitie van textuur en bijgevolg zijn er heel veel methodes om textuur te modelleren. Aangezien textuur gerelateerd is aan perceptie, stellen wij in dit proefschrift een nieuw beschrijvend textuurkenmerk voor.

Dit kenmerk is gebaseerd op de respons van cellen in de humane visuele cortex. Deze textuurkenmerken worden dan gebruikt in verscheidene toepassingen. In een eerste toepassing tonen we aan dat door gebruik te maken van de voorgestelde textuurkenmerken, de beste resultaten verkregen worden voor ongesuperviseerde classificatie van textuurbeelden. In een tweede toepassing tonen we aan dat voor supergeviseerde classificatie, de voorgestelde textuurkenmerken aanleiding geven tot even goede resultaten als door gebruik te maken van state-of-the-art textuurkenmerken. Aangezien ruis in veel praktische applicaties een veel voorkomend probleem is, hebben we de impact van verschillende types en hoeveelheden ruis voor textuurclassificatie onderzocht. Hierbij hebben we aangetoond dat de voorgestelde textuurkenmerken, in tegenstelling tot de state-of-the-art technieken, robuust zijn tegen grote hoeveelheden ruis. Ook compressieartefacten (zoals geïntroduceerd door JPEG-compressie) hebben een negatieve invloed op textuurclassificatie. We hebben aangetoond dat de voorgestelde textuurkenmerken voldoende robuust hiertegen zijn. In een vierde toepassing hebben we de voorgestelde kenmerken gebruikt voor materiaalclassificatie. Ook in deze toepassing worden de beste resultaten verkregen door de voorgestelde kenmerken. Tenslotte hebben we de automatische interpretatie van openluchtfoto's welke natuurlijke textuurinformatie bevatten, onderzocht. Bij deze automatische interpretatie van foto's wordt er gebruik gemaakt van segmentatie, materiaaldetectie en het toepassen van domeinkennis. Deze domeinkennis beschrijft de relaties tussen afgebeelde concepten en wordt dan via de top-down strategie toegepast om mogelijke fouten te elimineren. De experimenten duiden aan dat door gebruik te maken van de voorgestelde kenmerken, de beste resultaten worden verkregen.

In het muziekverwerkingsdomein hebben we een bottom-up strategie gebruikt om melodie- en tonaliteitinformatie te verwerven uit muziekopnames. Zowel melodie als tonaliteit zijn twee essentiële kenmerken van Westerse muziek en zijn gerelateerd aan toonhoogteperceptie. Luisteraars onthouden vaak makkelijker de melodie van een nummer dan de titel en de uitvoerder, of de gezongen tekst. Maar het automatisch extraheren van melodische informatie uit meerstemmige muziek is geen triviale taak. Dit komt omdat de frequenties van verschillende instrumenten elkaar soms overlappen. In dit proefschrift stellen we een methode voor om de melodie uit meerstemmige muziekopnames te extraheren. Onze doelstelling is om een symbolische notennotatie te verkrijgen. Ons voorgestelde systeem bestaat uit vier delen. In het eerste deel wordt toonhoogte-informatie uit de muziekopname geëxtraheerd door gebruik te maken van een gehoorsmodel. In het tweede deel wordt in elke tijdsvenster de fundamentele frequentie geschat (deze is gerelateerd aan de perceptuele toonhoogte) en toonhoogtetrajecten worden gecreëerd over opeenvolgende tijdsvensters. In het derde deel worden deze trajecten omgevormd tot een symbolische voorstelling. Tenslotte wordt in het laatste deel een filtering toegepast om minder belangrijke symbolen te verwijderen en zodoende een symbolische notenvoorstelling te bekomen.

Naast het bekomen van melodische informatie hebben we ook de extractie en classificatie van tonaliteitinformatie behandeld. Zoals melodie is ook tonaliteit gerelateerd aan toonhoogteperceptie, maar tonaliteit manifesteert zich dan over een relatief groter tijdsinterval. Tonaliteit is een harmonisch systeem van 24 toonaarden (12 majeur en 12 mineur) dat aangeeft welke soort intervallen gebruikt worden. De extractie van tonaliteitinformatie uit muziekopnames wordt verkregen door toonhoogte-inductie welke het signaal transformeert in een hoogdimensionale kenmerkenvector. Ondanks dat aangetoond is dat tonaliteit laagdimensioneel is, worden verschillende toonaardsequenties bekomen wanneer we deze laagdimensionaliteit benutten door middel van de afstandsgebaseerde methode. In dit proefschrift stellen we een nieuwe methode voor die gebaseerd is op het gebruik van classificatiebomen welke niet onderhevig aan deze beperking. Bovendien biedt de voorgestelde methode een mogelijkheid aan om toonaarden te groeperen in klassen door een of meerder niveaus te stijgen in de boom en zodoende een compactere representatie te bekomen.

In ons derde en laatste onderzoeksdomein hebben we de representatie van semantische metadata onderzocht. Om efficiënt persoonlijke data te beheren en te onsluiten, zijn er specifieke vereisten voor een contentmanagementsysteem. Naast een verzameling van procedures die de work-flow regelen, bevat een contentmanagementsysteem ook een metadatamodel. Binnen de context van een contentmanagementsysteem is de semantische betekenis van de verschillende velden van het metadatamodel duidelijk, maar de interoperabiliteit tussen andere metadatastandaarden is vaak niet gegarandeerd. Om de uitwisseling van metadata ten goede te komen, is deze interoperabiliteit echter een noodzaak. Daarom vinden we het noodzakelijk dat het metadatamodel de semantiek formeel kan definiëren. In dit proefschrift hebben we hiervoor een oplossing voorgesteld en deze uitgewerkt in een contentmanagementsysteem voor persoonlijke data welke op verschillende toestellen bewaard wordt. Het metadatamodel van het voorgestelde contentmanagementsysteem is een ontologie die gemodelleerd is door middel van Semantische Webtechnologieën. De ontworpen ontologie is een OWL DL ontologie aangezien automatische redenering op de metadata ondersteund moet worden. We hebben enkele modelleringtechnieken uiteengezet en zwakke punten van OWL DL aangeduid teneinde een OWL DL ontologie te kunnen bekomen. De gemodelleerde ontologie bestaat uit drie grote delen: het eerste deel omvat de metadata die het systeem beschrijft, het tweede deel beschrijft de beveiliging van de data en metadata en het derde deel beschrijft de metadata die door gebruikers kunnen worden toegevoegd. Deze laatste categorie omvat, naast metadata om de intellectuele eigendom te modelleren, ook metadata om de inhoud te beschrijven. Inhoudsbeschrijvende metadata is afhankelijk van het type inhoud dat beschreven wordt aangezien verschillende mediatypes (zoals bijvoorbeeld beeld, geluid, video of tekst) verschillende eigenschappen hebben. We hebben ons verdiept in metadata voor digitale beelden en een ontologie uitgewerkt die deel uitmaakt van het metadatamodel van het contentmanagementsysteem. Deze ontologie is een rijke ontologie die veel concepten ondersteunt om beelden te annoteren en bovendien een taxonomie omvat. Door gebruik te maken van Semantische Webtechnologieën kunnen interoperabiliteitsproblemen tussen verschillende metadatastandaarden en -modellen gemakkelijk verholpen worden. De voorgestelde oplossing maakt dan gebruik van semantische representaties van metadatastandaarden en afbeeldingen tussen de respectievelijke ontologiën. Zodoende fungeert de voorgestelde ontologie als een upper-ontologie om concepten uit verschillende metadatastandaarden te relateren.

List of Abbreviations

BMU	Best Matching Unit
BRICKS	Building Resources for Integrated Cultural Knowledge Services
CBIR	Content-Based Information Retrieval
CVIR	Content-Based Visual Information Retrieval
CMS	Content Management System
COV	Color Opponent Values
DFT	Discrete Fourier Transform
EXIF	Exchangeable Image File Format
FFT	Fast Fourier Transform
FOAF	Friend of a Friend
FT	Fourier Transform
GLCM	Gray Level Co-occurence Matrix
GMRF	Gaussian Markov Random Field
HSV	Hue Saturation Value
HVS	Human Visual System
I3A	not-for-profit International Imaging Industry Association
IPR	Intellectual Property Rights
JPEG	Joint Picture Experts Group
LBP	Local Binary Pattern
LGN	Lateral Geniculate Nucleus
MMSem	W3C Multimedia Semantics Incubator Group
MPEG	Moving Picture Experts Group
OWL	Web Ontology Language
PCA	Principal Component Analysis
PDF	Probability Density Function
PECMAN	Personal Content Management Platform
RDF	Resource Description Framework
RDFS	RDF Schema
RGB	Red Green Blue
SOM	Self-Organizing Map
SPARQL	SPARQL Protocol And RDF Query Language
STFT	Short-Time Fourier Transform
SWRL	Semantic Web Rule Language

URIref	Uniform Resource Identifier Reference
VIR	Visual Information Retrieval
W3C	World Wide Web Consortium
WWW	Word Wide Web
XML	Extensible Markup Language

Contents

Chapter 1

Introduction

The only limit to our realization of tomorrow,
will be our doubts of today.
– Franklin D. Roosevelt (1901 – 1945)

In this dissertation, we present our research on the extraction, application, and representation of semantic information. The presented research deals with typical difficulties of three domains: (i) computer vision, (ii) music audio mining, and (iii) metadata modeling. In the computer vision domain, we focus on the extraction of perceptual-based texture information and its application for segmentation, classification, and interpretation of images. The modeling of a novel texture feature is inspired by the human visual system, and we show that these texture features have high discriminating capabilities and are robust to noise and image compression artefacts. In the musical audio mining domain, we tackle the extraction of melodic lines and tonality information from polyphonic music. Unlike most other melody extraction approaches, we aim to explicitly distinguish individual musical notes, characterized by specific temporal boundaries. Finally, the third domain handles the representation of metadata using Semantic Web technologies. This domain is extended to the integration of metadata standards in distributed (personal) content management systems and the annotation of digital images. In the next section, we describe the general context and explain some important concepts.

1.1 Context

Over the last decades, the development of digitization techniques and the increasing capacity together with the decreasing costs of storage media, has led to an explosion of digital content. Huge amounts of audio, images, and videos are generated dailey and are mostly stored in an unstructured repository of multimedia information, much of which can be accessed through the Internet. According to a study of the International Data Corporation (IDC), there will be approximately 1.8 zettabytes of data in 2011 [1]. The biggest growth in data is visual in nature, from devices such as digital cameras, digital surveillance cameras, and digital televisions. Further, the music sector is generating far greater value from the online and mobile market than any other sector in the creative industries, with the exception of electronic games. For example, Apple's iTunes Store[1] offers its customers over ten billion songs[2] to choose from and the file sharing networks have billions of tracks available for download.

To deal with these huge amounts of data, one of the main imperatives IT organizations already face, are new tools and standards for data search and analysis. In particular, there is a need for efficient techniques that are able to extract information directly from the content. Indeed, with the growth of digital content, the need for *content-based* retrieval techniques has drastically increased. Content-based means that, e.g., a search relies on the content rather than relying on human-inputted metadata, such as captions or keywords. Obviously, manual annotation is a cumbersome and expensive task for large collections, and is often subjective, context-sensitive, and incomplete. As a result, it is difficult for the traditional text-based methods to support a variety of task-dependent queries, such as finding a song by singing the melody or finding similar images. However, despite the vast amount of digitized media files and the existence of various types of portable devices, the disclosure of these multimedia data is still mainly text-based. For example, queries to retrieve musical audio are still based on categories, such as artist, title, or genre. This leads to a number of limitations for both the end-users and service providers in what concerns database search and the manual assignment of metadata, respectively. Thus, to disclose large databases in an efficient manner, it is important that the property of interest, such as the melody and vocal parts for music databases or the depicted concepts for photo collections, can be automatically extracted from the content.

Within the domain of content-based information retrieval, there is a major dis-

[1]http://www.itunes.com

[2]http://www.apple.com/pr/library/2010/02/25itunes.html

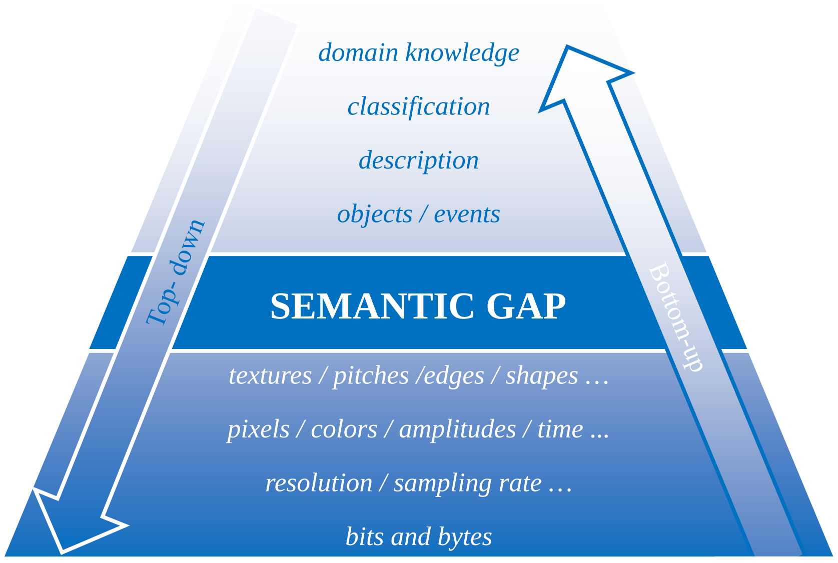

domain knowledge
classification
description
objects / events
SEMANTIC GAP
textures / pitches /edges / shapes …
pixels / colors / amplitudes / time ...
resolution / sampling rate …
bits and bytes
Top- down
Bottom-up

Figure 1.1: The semantic gap in content-based information retrieval.

crepancy, i.e., the *semantic gap*. Smeulders et al. [2] describe the semantic gap as "the lack of coincidence between the information that one can extract from the data and the interpretation that the same data have for a user in a given situation", as is shown in Figure 1.1. This means that a user wants to retrieve data on a semantic level, but the characterizations can only provide a low-level similarity. For example, one wants to find images containing flowers, but an image file only contains information about the color of the pixels.

Over the last years, a plethora of approaches have been developed to bridge the semantic gap. These attempts mainly apply two kinds of strategies: (i) a bottom-up or (ii) a top-down strategy. The bottom-up approach starts from extracting features at the signal level (low level) and then applies machine learning techniques for concept detection (high level). Contrary, the top-down approach starts from applying domain knowledge to gain insight into the compositional elements which are then refined in greater detail. However, content-based information retrieval systems often combine both strategies to generate perceptually and semantically more meaningful retrieval results, e.g., by incorporating relevance feedback obtained from the end-user. Figure 1.2 depicts the architecture of a typical content-based information retrieval system. In such a system, the contents of the item in the database are extracted and described by (multi-dimensional) feature vectors. The feature vectors of the items in the

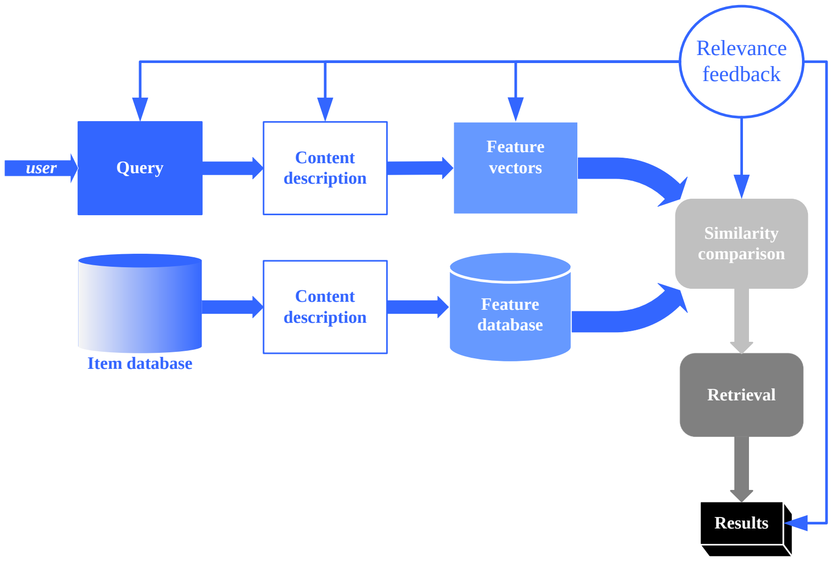


Figure 1.2: The architecture of a content-based information retrieval system.

database form a feature database. To retrieve items, a query can be formulated by an example (i.e., the query-by-example paradigm where users input content they consider similar to the desired retrieval results). Queries can also be formulated by keywords or descriptions if the latter can be correlated with the extracted features by the system. Such systems are usually interactive with a domain specific definition of similarity. In a content-based retrieval system, it is important to select the appropriate features that have optimal discriminatory power. The process of automatically or interactively choosing the best set of features for a particular application, is denoted as feature selection. In the case of query-by-example, the system then transforms the example into its internal representation of feature vectors. When the features have been selected, a metric is chosen, and the similarities between the feature vectors of the query and those of the items in the database are calculated. An ideal measure of feature similarity should be correlated to the user's intuitive sense of similarity. In the last stage, retrieval is performed, possibly by incorporating users' relevance feedback. The query results can be outputted as a list ranked by the similarity measure.

Next to the semantic gap, a second discrepancy for content-based retrieval can be identified: the *sensory gap*. The sensory gap is the gap between the object in the world and the information in a (computational) description derived from a recording [2]. In other words, the sensory gap is the discrepancy between the properties in a (digital) representation, e.g., an image or audio-recording and the properties of the concept. The sensory gap makes the description of objects an ill-posed problem: it yields uncertainty in what is known about the state of the object. The sensory gap is particularly poignant when a precise knowledge of the recording conditions is missing.

A *good* content descriptor is essential in any content-based retrieval application. In this context, good also means that the descriptor should be insensitive to the accidental variance introduced by the content creation process, e.g., the variation of the illuminant in visual data or the presence of noise in an audio recording. However, there is a tradeoff between this invariance and the discriminative power of features since a very wide class of invariance loses the ability to discriminate between essential differences.

Further, the study of the IDC [1] also states that approximately 70% of all data are created by individuals. As a result, the need for representative data models for the management and disclosure of personal data becomes higher. Today, one's personal content is stored on many devices, such as a smart phone, an MP3-player, a laptop, or somewhere on the World Wide Web. As a result, it becomes more difficult to efficiently manage these personal data and the related metadata. Therefore, a system to manage these data is needed that, transparantly, keeps track of the data sources and allows annotations.

The work we present in this dissertation proposes methods to bridge the semantic gap in two domains, i.e., in computer vision and in musical audio mining. We propose methods to relate these features with concepts that are related with human perception. In the computer vision domain, we focus on the modeling of textures. As will be explained, texture is an important property in computer vision. Next, important properties of Western music are (besides rhythm) melody and tonality. In this thesis, we present a bottom-up approach to extract the melody and tonality from musical audio signals. Finally, our third research area handles the representation of semantic metadata. This domain is extended to the modeling and the integration of standardized metadata in personal content management systems. In the next three sections, we elucidate these three research topics.

1.2 Texture

Texture analysis plays an important role in many image analysis applications. Even though color is important for interpreting images, there are situations

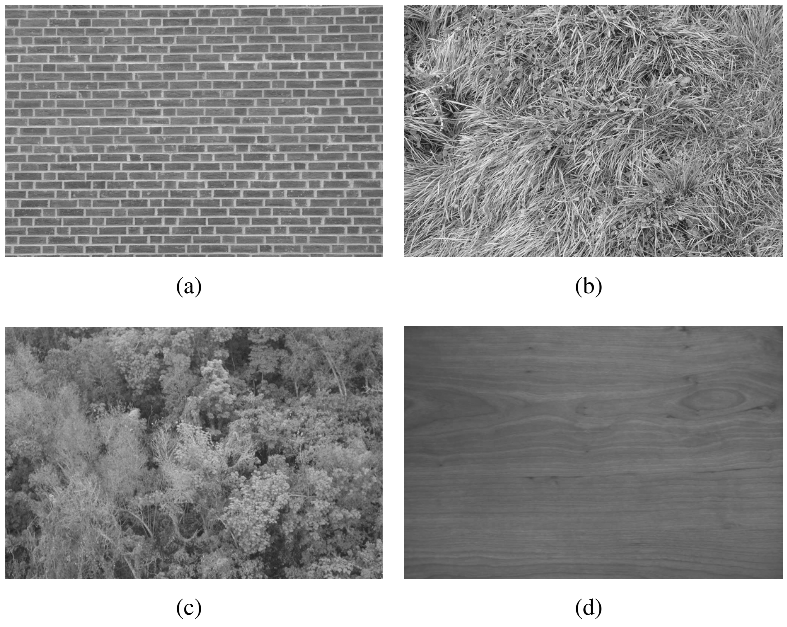


Figure 1.3: Examples of some everyday textures: (a) bricks, (b) grass, (c) trees, and (d) wood.

where color information is not enough, nor even applicable. For example, in industrial visual inspection color and texture information can be used for error detection. Or, in some applications, such as in the quality control of paper or parquet slabs, there is no color at all. Texture measures can also cope better with varying illumination conditions, for instance in outdoor conditions. Therefore, they can be useful tools for high-level interpretation of natural scene image content. The latter property is exemplified in Figure 1.3 which depicts some everyday textures. Despite the lack of color information, one is able to recognize the depicted concepts. Texture methods can also be used in medical image analysis, biometric identification, remote sensing, content-based image retrieval, document analysis, and model-based image coding.

Many methods have been proposed in the literature to analyze textures, such as statistical, structural, and stochastic approaches. In this dissertation, we present a texture feature that is inspired by the human visual system. We rely on responses of cells that are located in the primary visual cortex of humans. These features are then used for unsupervised segmentation of textured images, semi-supervised texture classification, and the automatic interpretation of outdoor scenery images. Furthermore, we show that these texture features are robust to noise and image compression artifacts.

1.3 Musical Audio Mining

In order to overcome the limitations that are related to traditional text-based retrieval in music databases, there is a need for automatic music transcription techniques. Music transcription refers to the analysis of a musical signal in order to produce a representation of, e.g., the sounding notes in the musical signal. Melody is one of the main characteristics of Western music, and is related to pitch perception. Furthermore, it is a fact that people often remember the melody of a song rather than the performer and the title. Even after the lyrics have been forgotten, it is the melody that humans are likely to recall or are able to reproduce by humming or whistling. Despite the fact that after hearing a song – even untrained – listeners are able to reproduce the melody, automatic melody transcription is not an easy task. Moreover, as music is nowadays typically produced by different, simultaneously playing instruments, melody transcription becomes more complex. The latter is because frequencies of concurrent sounds may coincide. As a result, it is difficult to designate the frequency components that have been produced by the instrument which plays the melody. If the melody can be automatically extracted from a musical signal, it is possible to query a music database by singing, humming, or whistling the melody, i.e., query by melody. Besides query by melody, other applications, such as plagiarism detection, music analysis, and performance analysis, could gain from melody transcription as well. In this dissertation, we present a transcription system to automatically extract the melody from both monophonic and polyphonic musical audio recordings. We aim to distinguish individual musical notes characterized by specific temporal boundaries in order to obtain a symbolic representation.

Besides melody, another important musical property that is related to pitch perperception, is tonality. The pitch that attains the greatest stability in a musical passage is called the musical key or tonal center. Almost all Western music is built in a way such that certain pitches functionally operate as attractor of other pitches. In music theory, this is expressed by the concept of tonality. Tonality is a harmonic system of 24 keys, i.e., 12 major and 12 minor keys. Recognition of musical key is an important step in the exploration of methods that allow the development of multi-level music representation schemes for musical content, such as chord recognition and also emotion detection since the musical key is deemed to provide a specific emotional connotation.

1.4 Metadata

In order to efficiently manage personal data and the related metadata, a content management system (CMS) is needed. Besides a set of procedures to manage the work flow, a CMS typically involves a metadata model. Within the context of a single CMS, the semantics of different fields of the metadata model are well-defined. However, there is a major discrepancy when it comes to the interoperability between different metadata models. Therefore, it is indispensable that the semantics can be formally defined by the metadata model. Different metadata standards have been proposed using the Extensible Markup Language (XML) [3] as underlying language. XML allows to structure data according to a schema which defines the constructs that represent the metadata. However, as XML is a structuring language, it infers some problems to define the semantics of the described concepts. In this dissertation, we have tackled this issue in the context of a personal CMS and the annotation of photos. We introduce the application of Semantic Web technologies for personal CMSs and as a way to integrate metadata standards. The Semantic Web provides a framework that allows to formally describe the semantics of information resources. As such, these metadata can be shared and reused across different applications, communities, and enterprises without losing the semantic meaning of the concepts and relationships. Within this dissertation, we deploy the Web Ontology Language (OWL) [4] which permits to define concepts and relationships on which can be reasoned. Finally, relying on OWL, we present a layered architecture for a semantic personal CMS.

1.5 Outline

This dissertation is organized as follows. In Chapter 2, we present the modeling of a novel texture feature which is inspired by the human visual system. First, we explain the typical characteristics of texture, and second, we present related work in this domain. Next, we present the texture model by relying on responses of cells that are located in the primary visual cortex of humans. Then, after the parameter selection, we use this novel feature for unsupervised texture segmentation, semi-supervised texture classification, semi-supervised material classification, and outdoor scene labeling. The accuracy of our approaches is shown by comparing our results with state-of-the-art texture features.

In Chapter 3, we present our research in the musical audio mining domain.

This encloses melody and tonality extraction from polyphonic music. First, we propose an automatic melody transcription system which exists of four phases: (i) auditory-model-based pitch detection, (ii) fundamental frequency estimation and pitch tracing, (iii) tone creation, and (iv) filtering. We evaluate our approach on different labeled melody collections and discuss the performance. Second, we present a method for extracting tonality information from polyphonic music recordings. We discuss two key-recognition methods for musical audio: one (classical) metric-based method and a novel tree-based method.

In Chapter 4, we present a distributed personal content management system that relies on Semantic Web technologies. At first, we present the system requirements and the related work. Next, we present a model for a personal CMS by relying on Semantic Web technologies, and we introduce a taxonomy for (photo) annotation. Finally, we solve the addressed interoperability issues between metadata standards using Semantic Web technologies.

Finally, the conclusions and future work of this dissertation are presented in Chapter 5.

1.6 Overview of Publications

The research activities that have lead to this dissertation resulted in 7 journal papers that are listed in the Science Citation Index from which 3 as first author and 4 as co-author. Our work als contribtuted a book chapter in Self-Organizing Maps (In-Tech), and in Data Management in the Semantic Web (Nova). Further, our work contributed to 14 papers.

1.6.1 SCI-listed Publications

1. M. Leman, J. Dierickx, and G. Martens. The IPEM-archive conservation and digitalization project. *Journal Of New Music Research*, 30(4):389–393, 2001

2. G. Martens, H. De Meyer, B. De Baets, M. Leman, M. Lesaffre, and J.P. Martens. Tree-based versus distance-based key recognition in musical audio. *Soft Computing*, 9(8):565–574, 2005

3. C. Poppe, G. Martens, S. De Bruyne, P. Lambert, and R. Van de Walle. Robust spatio-temporal multimodal background subtraction for video surveillance. *Optical Engineering*, 47:110101, October 2008

4. C. Poppe, G. Martens, E. Mannens, and R. Van de Walle. Personal Content Management System a Semantic Approach. *Visual Communication and Image Representation*, 47:131 – 144, February 2009

5. G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Semi-supervised classification of robust texture features inspired by the human visual system. *Visual Communication and Image Representation*, 2010. Under review

6. G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Noise and compression robust biological features for texture classification. *The Visual Computer*, 26(6-8):915–922, June 2010

7. C. Poppe, G. Martens, P. De Potter, and R. Van de Walle. Semantic web technologies for surveillance metadata. *Multimedia Tools and Applications*

1.6.2 Book Chapters

1. G. Martens, P. Lambert, and R. Van de Walle. *Self-Organizing Maps*, chapter Bridging the semantic gap using human vision system inspired features, pages 261–276. In-Tech, 2010

2. C. Poppe, G. Martens, E. Mannens, and R. Van de Walle. *Data Management in the Semantic Web*, chapter Creating Personal Content Management Systems Using Semantic Web Technologies. Distributed, Cluster and Grid Computing. Nova, 2011

1.6.3 Other Publications

1. G. Martens, H. De Meyer, B. De Baets, M. Leman, J.P. Martens, L. Clarisse, and M. Lesaffre. A tonality-oriented symbolic representation of musical audio generated by classiffication trees. In *The EUROFUSE Workshop on Information Systems*, 2002

2. M. Leman, L. Clarisse, B. De Baets, H. De Meyer, M. Lesaffre, G. Martens, J.P. Martens, and D. Van Steelant. Tendencies, perspectives, and opportunities of musical audio-mining. In A Calvo-Manzano, A Perez-Lopez, and J Salvador Santiago, editors, *Revista de Acustica*, volume 33, 2002

3. M. Lesaffre, K. Tanghe, G. Martens, D. Moelants, M. Leman, B. De Baets, H. De Meyer, and J.P. Martens. The MAMI Query-By-Voice Experiment: Collecting and annotating vocal queries for music information retrieval. In *Proceedings of the International Conference on Music Information Retrieval*, pages 65–71, 2003

4. M. Lesaffre, D. Moelants, M. Leman, B. De Baets, H. De Meyer, G. Martens, and J.P. Martens. User behavior in the spontaneous reproduction of musical pieces by vocal query. In *Proceedings 5th Triennial ESCOM Conference*, pages 208–211, 2003

5. G. Martens, C. Poppe, and R. Van de Walle. Enhanced grating cell features for unsupervised texture segmentation. In W. Ponweiser, editor, *Performance Evaluation for Computer Vision: 31ste AAPR/OAGM Workshop 2007*, pages 9–16, Wien, 5 2007. Osterreichische Computer Gesellschaft

6. C. Poppe, G. Martens, P. Lambert, and R. Van de Walle. Mixture Models Based Background Subtraction for Video Surveillance Applications. In *Lecture Notes in Computer Science: Proceedings 12th International Conference on Computer Analysis of Images and Patterns*, volume 4673, pages 28–35, August 2007

7. C. Poppe, G. Martens, P. Lambert, and R. Van de Walle. Improved Background Mixture Models for Video Surveillance Applications. *Lecture Notes in Computer Science, 8th Asian Conference on Computer Vision(ACCV 2007)*, 4843:251–260, 2007

8. C. Poppe, G. Martens, P. Lambert, and R. Van de Walle. Dealing with Quick Illumination Changes when using Background Mixture Models. In *Proceedings 8th Asian Conference on Computer Vision*, volume 4843, Tokyo, November 2007

9. C. Poppe, G. Martens, S. De Bruyne, P. Lambert, and R. Van de Walle. Dealing with Gradual Lighting Changes in Video Surveillance for Indoor Environments. In *Proceedings of 15th World Congress on Intelligent Transport Systems*, November 2008

10. C. Poppe, S. De Bruyne, G. Martens, P. Lambert, and R. Van de Walle. Intelligent Preprocessing for Fast Moving Object Detection. In *Proceedings of SPIE Security and Defense*, volume 6978, March 2008

11. D. Van Deursen, C. Poppe, G. Martens, E. Mannens, and R. Van de Walle. XML to RDF conversion : a generic approach. In P. Nesi,

K. Ng, and J. Delgado, editors, *Fourth International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, pages 138–144. IEEE Computer Society, 2008

12. G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Unsupervised texture segmentation and labeling using biologically inspired features. In D. Feng, D. Sikora, W.C. Siu, J. Zhang, J. Guan, L. Dugelay, Q. Wu, and Li W., editors, *Proceedings of the 2008 IEEE 10th workshop on Multimedia Signal Processing*, pages 159–164, Cairns, 10 2008. IEEE Signal Processing Society

13. G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Perceptual-based textures for scene labeling: a bottom-up and a top-down approach, 5 2010

14. G. Martens, C. Poppe, and R. Van de Walle. Lifting a metadata model to the semantic multimedia world. In *The 2010 International Workshop on Advanced Future Multimedia Services*, 2010. Accepted

Chapter 2

Texture Analysis

You can't predict the future, but you can invent it.
– Dennis Gabor (1900 – 1979)

2.1 Introduction

Due to the large volume and variety of digital images that are used in different application domains, intelligent retrieval techniques have become compulsory. In particular for large collections, there is an increasing need for automatic visual content analysis in order to efficiently retrieve information. Large collections of visual information can be found in many application domains, such as medicine, journalism, identification and surveillance, geographical information systems, remote sensing, entertainment, digital catalogues, etc.

As in conventional information retrieval, the purpose of a visual information retrieval (VIR) system is to retrieve all the images (or image sequences) that are relevant to a user query while retrieving as few non-relevant images as possible. A content-based VIR system must be able to interpret the content of the documents (i.e., images and/or video) in a collection and rank them according to a degree of relevance to the user query. The interpretation process involves extracting (semantic) information from the documents and using this information to match the user needs [28]. A number of related problems and constrains to content-based VIR are:

- *Lighting*. Besides the physical content of the scene and the characteristics of the camera, the color of a recorded image also depends on the illumination incident on the scene. Human perception normalizes perceived spectra with respect to global scene illumination, a phenomenon known as "color constancy" [29]. However, in the case of image acquisition the scene illumination is often not known or not homogeneous. Given certain assumptions about lighting conditions, the reconstruction of the actual color of an object is not trivial since the light reflected from the object also depends on the material where the object is made of and the object's surface, two parameters which are often unknown.

- *Context.* The context of a content-based VIR system and the type of visual information have a strong effect on how the content is described and compared. For example, searching for images with a similar color composition or searching for images with similar objects yield two different approaches concerning the type of visual information. Domain knowledge and contextual information are essential in deriving an appropriate image representation. Furthermore, the context may play an important role in determining which methods are to be used.

- *Object recognition.* The ability to detect objects would support the retrieval of semanticly similar visual content. However, object recognition is a computationally hard problem. Recognition typically involves the computation of a set of components at one step and their combination in the next step [30, 31]. If a reasonably good match is found, successful object recognition will occur.

- *Image segmentation.* Partitioning an image into regions that are meaningful with respect to a particular application, is essential in image understanding. Segmentation could be used for, e.g., object recognition, occlusion boundary, estimation within motion or stereo systems, image compression, image editing, or image database look-up, etc. However, these are difficult to achieve since one has to use low-level image features (e.g., color) which are obtained from a 2-dimensional representation that depicts a 3-dimensional scene. Furthermore, humans probably use object recognition in conjunction with segmentation.

- *Motion.* In computer vision, there are several tasks related to motion estimation of image sequences:

  - Ego motion: knowledge of the camera rotation and translation between frames is crucial input to many vision applications.
  - Tracking: following or predicting the movements of objects or a set of interest points in the image sequence.

– Optical flow: the apparent motion in a visual scene caused by the relative motion between the camera and the scene.

Content-based VIR retrieval resides on features that are able to represent the content for the desired application. The choice of a feature is thus essential for efficient retrieval. These features are computed from the pixels' values. Content-based VIR systems do not use all known features as this would involve large amounts of data and increase the necessary computing time. Instead, a set of features appropriate to the given task is usually selected. However, it is difficult to judge beforehand which features are appropriate for which tasks. Three major types of low-level features can be discriminated for content-based VIR:

1. *Color*. The choice of a color space is of great importance for a VIR application. Color induces equivalent classes in the actual retrieval algorithm. Important criteria in the choice of a color system are, e.g., independence of the underlying imaging device (when images are recorded by different devices) and numerical distances within the color space that can be related to perceptual differences (visual similarity). Color histograms are widely used in image retrieval and give an estimation of the distribution of the colors in the image.

2. *Shape*. An interesting shape description should be invariant to translation, rotation, scaling and starting point transformations. Typical problems relate to shape matching which deals with transforming shape patterns and measuring the resemblance. Generally, shapes can be analyzed at the boundary level or interior level.

3. *Texture*. In computer vision, there is no exact definition of texture. Texture is an intuitive concept. The lack of a formal definition of texture has led to a variety of analysis methods. Both statistical, stochastical and structural measures have been proposed for texture analysis.

Depending on the context of the application, the importance of a certain content descriptor can differ. For example, color is an essential feature for classifying different ground regions from aerial or satellite photographs, while in echocardiographic images texture is very important.

Since texture is much more robust than color with respect to lighting conditions, it can play an important role in building a system that implements a bottom-up approach (see Section 1.1) to extract information from visual data. Indeed, Renninger and Malik already have concluded that the human visual system (HVS) uses texture analysis for rapid scene identification [32]. Furthermore, humans outperform most machine vision systems in many aspects and carry out those tasks seemingly effortlessly. Therefore, building a system for texture classification that emulates the processing principles of the human brain is an attractive and challenging idea.

In this chapter, we focus on texture analysis methods. More specifically, we propose a new texture feature which is based on the HVS. We investigate the segmentation of textured images, the classification of textures, material classification, and outdoor scene analysis using the proposed texture analysis method. The outline of this chapter is as follows. Section 2.2 describes the general context of the notion texture. In Section 2.3, we give an overview of related work in the domain of texture analysis. In Section 2.4, we describe our texture model which is based on the HVS and in Section 2.5 we clarify how the parameters of the model are selected. In the context of this dissertation, a texture model transforms a window of an image into a feature vector which can be thought of as a point in an multi-dimensional feature space. Windows taken from the same texture sample form a cluster in feature space, and windows taken from different texture samples are far away from each other in feature space. Section 2.6 describes the unsupervised segmentation of textured images. Both the segmentation of mosaic images and outdoor scenery photos are considered. Next, in Section 2.7 we investigate the classification of textures. Furthermore, we examine the robustness against various kinds and levels of noise in textures for classification. Also the impact of lossy image compression algorithms on texture analysis is investigated. Section 2.8 then investigates the use of texture information obtained from a single image for material classification. The segmentation and labeling of the obtained regions of outdoor scenery images by linking texture information with semantic concepts is considered in Section 2.9. We end this chapter with conclusions and future work in Section 2.10.

2.2 Texture, a Concept

We all know that a wall of brick has a right-angled pattern while the branches of a tree, if viewed from an appropriate distance, have a tangled appearance, and that the surface of our woolen jersey looks different than fleece. Despite the intuitive character of texture, a precise definition is hard to formulate. Textures have been described in terms of the HVS, i.e., textures don't have a uniform intensity but are perceived as homogeneous regions, but also in terms of the application in which textures are used. Some examples of texture definitions are:

- "An image texture may be defined as a local arrangement of image irradiances projected from a surface patch of perceptually homogeneous irradiances." [33]
- "Texture regions give different interpretations at different distances and at different degrees of visual attention. At a standard distance with normal attention, it gives the notion of macroregularity that is characteristic of the particular texture." [34]
- "An image texture is described by the number and types of its (tonal) primitives... A fundamental characteristic of texture: it cannot be analyzed with a frame or reference of tonal primitive being stated or implied. For any smooth gray tone surface, there exists a scale such that when the surface is examined, it has no texture. Then as resolution increases, it takes on a fine texture and then a coarse texture." [35]
- "Texture is defined for our purposes as an attribute of a field having no components that appear enumerable. The phase relations between the components are thus not apparent. Nor should the field contain an obvious gradient. The intent of this definition is to direct attention of the observer to the global properties of the display, i.e., its overall 'coarseness', 'bumpiness', or 'fineness'. Physically, nonenumerable (aperiodic) patterns are generated by stochastic as opposed to deterministic processes. Perceptually, however, the set of all patterns without obvious enumerable components will include many deterministic (and even periodic) textures." [36]
- "The notion of texture appears to depend upon three ingredients: (i) some local order is repeated over a region which is large in comparison to the order's size, (ii) the order consists in the nonrandom arrangement of elementary parts, and (iii) the parts are roughly uniform entities having approximately the same dimensions everywhere within the textured region." [37]

However, despite these differences, there are two widely accepted properties of texture:

1. Within a texture, there is significant variation in intensity levels between nearby pixels.

2. Texture is a uniform property at some spatial scale larger than the pixel size of an image.

In spite of the lack of a definition, texture plays an important role in many image analysis applications, such as surface inspection (e.g., aerial image segmentation, automated industrial inspection), content-based image retrieval, biomedical image analysis, computer graphics (e.g., texture mapping in surface or scene rendering applications, texture synthesis to create large textures from usually small texture samples).

2.2.1 Texture Analysis Methods

The lack of a formal definition and the great variety of applications in which texture analysis plays an important role, has led to a plethora of texture analysis methods in the literature. Generally, there exist three approaches for texture analysis: (i) statistical texture measures, (ii) stochastic texture modeling, and (iii) structural texture measures.

Statistical Texture Measures

Since textures may be random within certain consistent properties, one obvious way to describe textures is to treat a texture as a statistical phenomenon. Statistical methods analyze the spatial distribution of gray values and derive a set of statistics from the distribution. The reason behind this is the fact that the spatial distribution of gray values is one of the defining properties of texture.

First-order texture measures are statistics calculated from the original image values and do not consider pixel neighborhood relationships. The measures can be computed from the histogram of intensities (i.e., gray-levels) from an image or image region. Suppose that we construct the histogram of the intensities (h_i) from an image or image region and h_i denotes the number of pixels with gray-level i. Further, let N be the total number of pixels and G the number of gray-levels, then:

$$\sum_{i=0}^{G-1} h_i = N. \tag{2.1}$$

The normalized histogram (H_i) with $H_i = h_i/N$ then represents the empirical probability density function of the gray-values. Statistics computed from H_i are, e.g.:

- the gray-level mean: $\mu = \sum_{i=0}^{G-1} iH_i$,
- the gray-level variance: $\sigma^2 = \sum_{i=0}^{G-1} (i-\mu)^2 H_i$,
- the gray-level skewness: $\gamma = \frac{1}{\sigma} \sum_{i=0}^{G-1} (i-\mu)^3 H_i$,
- the gray-level kurtosis: $\kappa = \frac{1}{\sigma} \sum_{i=0}^{G-1} (i-\mu)^4 H_i - 3$,
- the gray-level energy: $e = \sum_{i=0}^{G-1} H_i^2$ where $G^{-1} \leq e \leq 1$,
- the gray-level entropy: $s = \sum_{i=0}^{G-1} H_i \log H_i$ where $0 < s \leq \log G$.

The mean μ measures the average gray-level, the variance σ^2 measures the global contrast, the skewness γ measures the extent to which outliers favor one side of the distribution, the kurtosis κ measures the "peakedness" of the gray-level distribution, the energy e measures the non-uniformity of the histogram or how much intensity variation there is, and the entropy s measures the uniformity. However, texture analysis based solely on first-order statistics suffers from the limitation that it provides no information about the relative position of pixels to each other. For example, 2 completely different images each with a 50% black and 50% white pixels (such as a checkerboard and a salt & pepper noise pattern) produce the same gray-level histogram and have the same first-order statistics.

As a means to overcome these issues, gray-level co-occurrence matrices (GLCM) were introduced by Haralick [38]. The gray-level co-occurrence matrix estimates image properties related to second-order statistics. The GLCM for a displacement vector d is defined as follows: the entry (i, j) of the matrix is the number of occurrences of the pair of gray-levels i and j which are a distance d apart. The co-occurrence matrix reveals properties about the spatial distribution of the gray-levels in the texture image. For example, if most of the entries in the co-occurrence matrix are concentrated along the diagonals, then the texture is coarse with respect to the displacement vector d. Haralick has proposed a number of useful texture features that can be computed from the co-occurrence matrix [35]. However, the co-occurrence matrix features suffer from a number of difficulties. There is no well established method of selecting the displacement vector and computing co-occurrence matrices for different values of d is not feasible. For a given displacement vector d, a large number of features can be computed from the co-occurrence matrix. This means that some sort of feature selection method must be used to select the most relevant features.

Identical second-order statistics do not guarantee identical textures, higher than second-order statistics contain little information that can be used for texture discrimination [39].

An important property of statistical methods is that the image cannot be recreated from the measured set. A survey of statistical approaches for texture is given by [35, 39].

Stochastic Texture Modeling

This approach assumes that a texture is the realization of a parametrical stochastic process. A model is defined and the parameters are estimated so that the stochastic process can be reproduced from the model and its associated parameters. The estimated parameters can serve as features for texture classification and segmentation problems. Stochastic texture models can be divided into three major groups: probability density function (PDF) models, gross shape models, and partial models [40].

1. *PDF models* model a texture as a random field and a statistical model is fitted to the spatial distribution of intensities in the texture. Typically, the interaction of a small number of pixels is measured. Examples of parametric PDF methods are Gaussian-Markov Random Field (GMRF) [41] and Uniform Clique Random Fields [42]. Examples of non-parametric PDF methods include texture spectrum methods which use PDF models that are sensitive to high-order interactions, such as the local binary pattern [43].

2. *Gross shape models* measure features which a human would consciously perceive. The features can either occur at regular intervals (periodicity) or can be spatially compact (e.g., edges, lines). The latter relate to *primitive* methods, such as Gabor filter and wavelet outputs [44–47] and the former relates to *harmonic* methods, such as autocorrelation features or Fourier domain features, e.g., [48, 49]. Harmonic methods model a texture as a summation of waveforms so they assume that intensity is a strongly periodic function of the spatial coordinates. The difference between primitive and harmonic methods is that primitive methods measure spatially local features and harmonic methods model spatially dispersed features.

3. *Partial models* focus on some specific aspect of texture to the detriment of other aspects. *Fractal* methods measure how a texture varies with

the scale, while *line* methods measure properties along one-dimensional contours in a texture [50].

Structural Methods

Structural methods define texture as a composition of texture elements or primitives. These primitives may be of varying or deterministic shape, such as circles, hexagons, or even dot patterns. The analysis usually depends upon the geometrical properties of these texture elements. In contrast to primitive methods which model a texture as many simple primitives, structural methods tend to have one complex primitive for which placement rules apply. In general, this class of algorithms is limited in power unless one is dealing with very regular textures. Structural texture analysis consists of two major steps: the extraction of the texture elements and the inference of the placement rule.

There are a number of ways to extract texture elements in images. Usually texture elements consist of regions in the image with uniform gray levels. Voorhees and Poggio [51] argued that blobs are important in texture perception. They have proposed a method based on filtering the image with Laplacian of Gaussian (LoG) masks at different scales and combining this information to extract the blobs in the image. Blostein and Ahuja [52] perform similar processing in order to extract texture elements in images by examining the response of the LoG filter at multiple scales. They integrate their multi-scale blob detection with surface shape computation in order to improve the results of both processes. Tomita and Tsuji [53] also suggest a method of computing texture elements by doing a medial axis transform on the connected components of a segmented image. They then compute a number of properties such as intensity and shapes of these detected elements. Zucker [54] has proposed a method in which he regards the observable textures (real textures) as distorted versions of ideal textures. The placement rule is defined for the ideal texture by a graph that is isomorphic to a regular or semi-regular tessellation. These graphs are then transformed to generate the observable texture. Which of the regular tessellations is used as the placement rule is inferred from the observable texture. This is done by computing a 2-dimensional histogram of the relative positions of the detected texture elements. Another approach to modeling texture by structural means is described by Fu [55]. In this approach, the texture image is regarded as texture primitives arranged according to a placement rule. The primitive can be as simple as a single pixel that can take a gray value, but it is usually a collection of pixels. The placement rule is defined by a tree grammar. A texture is then viewed as a string in the language defined by the grammar whose terminal symbols are the texture primitives. An advantage of this method is that it can be used for texture analysis as well as texture generation. The patterns generated by the tree grammars could also be regarded as ideal textures in Zucker's model.

2.2.2 Texture Classification

Given a set of known texture categories, the goal of texture classification is to assign a category to a texture sample. In order to achieve this, one needs to have a priori knowledge of the classes to be recognized. Once this knowledge is obtained, machine learning techniques can be applied for classification. Supervised classification algorithms need some knowledge of the data, be it either training samples or parameters of the assumed feature distributions. Classification algorithms can either be parametric or non-parametric. Parametric classifiers, like Bayesian [56] and Mahalanobis [57] classifiers, make certain assumptions about the distribution of features. Non-parametric classifiers, like the k-NN classifier [58], can be used with arbitrary feature distributions and without assumptions about the forms of the underlying densities. With non-supervised techniques, classes are to be found with no prior knowledge. This process is often called clustering. Examples of such methods include vector quantization, utilized in texture classification, and Self-Organizing Maps [59]. Semi-supervised classification is a hybrid approach that uses both labeled and unlabeled data in the decision process, e.g., by using labeled data to label a large amount of unlabeled data.

2.2.3 Texture Segmentation

Texture segmentation can be seen as a subclass of *image segmentation*. The goal of image segmentation is to cluster pixels into salient image regions, i.e., regions corresponding to individual surfaces, objects, or natural parts of objects so that a more compact image representation is obtained. This representation should be easier to analyze and bridge the gap between the low-level and the higher-level structures. However, general-purpose image segmentation is still an unsolved problem, i.e., there is no universally applicable segmentation technique that will work for all images. Therefore, image segmentation techniques using low-level features aim to identify homogeneous regions in an image. The homogeneity can be based on one or more of several properties, such as texture, color, distribution of the densities of the image elements. Image segmentation may use statistical classification, thresholding, edge detection, region detection, or any combination of these techniques. In contrast to the notion of texture, image segmentation is a well-defined problem. An image consists of an array of elements (pixels) and the aim is to give each element a label, i.e., to assign a pixel to one region.

Analogously, the goal of *texture segmentation* is to segment an image into regions according to the texture of the regions.

In general, segmentation methods are based on two basic properties of the pixels in relation to their local neighborhood: discontinuity and similarity. Methods that are based on some discontinuity property of the pixels are called boundary-based methods, whereas methods based on some similarity property are called region-based methods.

- *Region-based.* The region-based approach tries to isolate areas in images that are homogeneous according to a given set of features. These image features can consist of: (i) intensity values from original images or computed values based on an image operator, (ii) patterns that are unique to each type of region or, (iii) spectral profiles that provide multi-dimensional image data. Drawbacks of region-based techniques are that region boundaries are generally distorted. In addition, commonly used region-based techniques such as region-growing and clustering, have several critical design issues to be dealt with. The performance of most region-growing approaches crucially depends on the selection of the initial regions.

- *Boundary-based.* Boundary-based techniques use the assumption that abrupt changes occur with regard to the features of the pixels. A distinction can be made between ridge detection and edge detection. Ridge detection follows the local maxima in the original image whereas edge detection tracks maxima in the gradient space instead of the original image space. Major drawbacks of both boundary-based techniques is that they can produce spurious, missing, or discontinuous edges. Extensive post-processing may be needed to provide an accurate segmentation.

2.3 Related Work

In this section, we take a closer look at some widely used texture descriptors. The lack of a formal definition of texture makes the problem of analyzing textures ill-posed as analysis methods can be proven neither correct nor wrong. As a result, evaluation must be carried out in an empirical manner.

Texture description approaches that quantify the distribution of pixel intensities by single values, such as first order statistics, have a fundamental weakness: since the image is analyzed at a single scale, much important information about the distributions might be lost. Since Haralick et al. proposed GLCM for the classification of regions in satellite images, GLCM have become one of the most well-known and widely used texture features. But GLCM suffer from a number of difficulties as described in Section 2.2.1. Also Gaussian-Markov Random Fields (GMRF) have been extensively used [41]. GMRF methods use a Gaussian PDF to model the intensity of a pixel as a stochastic function of its neighboring pixels' intensity. The mean of the PDF is a linear function of the pixels' intensities. An important aspect of GMRF is the neighborhood considered round a pixel. Typically, a small neighborhood is used, and consequently GMRF features have problems in modeling macrotextures (i.e., textures that emerge over a relatively large area). Therefore, it is recommended to combine multiple GMRF features of different orders in one vector. Further, GMRF models also assume that second order PDFs are sufficient to characterize a texture. However, Hall and Gainnakis [60] created a test to determine whether a sample is likely to have been drawn from a 2-dimensional Gaussian distribution and they found that textures from the Brodatz album [61] failed the test. More recently, Ojala et al. [43] proposed the local binary pattern (LBP). A LBP code is calculated by thresholding a circular neighborhood with the value of the center pixel. The distribution of these codes is then represented by histograms. In general, the radius of the neighborhood and the number of samples can take arbitrary values. One of the major advantages of the LBP is the low computational complexity. As a result, LBP have been successfully used in real-time surface inspection applications, e.g, [62]. An important limitation of the LBP operator is its small spatial support area. For example, patterns calculated in a 3×3 pixels area cannot capture large-scale structures that may be prominent features of some textures. The latter can be solved by combining LBP with different parameters such that a multi-scale description of textures can be obtained [62, 63]. However, the enormous number of histogram bins results in huge memory consumption and slower classification.

Scenes in the world contain objects of many sizes and these objects contain features of many sizes. Moreover, objects can be at various distances from the viewer. As a result, any analysis procedure that is applied only at a single scale may miss information at other scales. The solution is to carry out analysis at all scales or resolutions simultaneously. Multi-resolution analysis provides a framework for identifying the properties of a signal, i.e., for taking note of its existence, measuring it, representing it, and even for reproducing it. At different resolutions, the details of a signal generally characterize different physical aspects. At coarse resolution, these details correspond to the larger overall aspects, i.e., the context. At fine resolution these details correspond to its distinguishing features. For multi-resolution signal analysis, wavelets and filter banks have been used independently in the fields of applied mathematics, computer vision, and signal processing.

The use of multi-resolution approaches is also very popular for texture analysis due to their relation with the HVS (see Appendix A for a general overview of the HVS). In fact, it is well-known that the human perception of texture is primarily a function of spatial-frequency analysis [64]. By studying the brain of the macaque monkey, De Valois et al. found that the simple cells in the primary visual cortex area have receptive fields which are restricted to small regions of space and respond only to appropriately oriented stimuli [65]. Complex cells are also orientation selective, but unlike simple cells, they have some degree of spatial invariance. They will respond to patterns in a certain orientation within a large receptive field, regardless of the exact location. Daugman [66] proposed the use of Gabor filters on the modeling of the receptive fields of the simple cells [66]. Gabor filters are bandpass filters tuned to a specific orientation and frequency that produce frequency compositions that achieve the theoretical lower bound of Heisenberg's uncertainty principle [67]. Consequently, filter banks consisting of multiple Gabor filters tuned to a unique orientation and periodicity, have been extensively used for texture analysis applications [68–70]. For a comprehensive comparison of different Gabor-based features for texture segmentation and classification, we refer to Clausi and Jernigan [71].

Wavelet analysis is performed using contracted and expanded versions of a single prototype function called a wavelet. Fine spatial resolution is achieved by using a contracted version of the wavelet, while fine frequency resolution can be achieved using an expanded version. However, in wavelet analysis, there is a tradeoff between spatial and frequency resolution. The latter is caused by the Heisenberg's uncertainty principle, i.e., one can gain good frequency response at the expense of a poor place resolution. From a signal processing point of view, wavelets can be thought of as a bandpass filter. So, filter banks can be used to implement a wavelet transform.

In computer vision, a successive approximation for a multi-resolution technique called *image pyramid* is used. Image pyramids derive a low resolution version of the original signal and then predict the original based on the coarse version by taking the difference between the original and the prediction. At the reconstruction, the difference is added back to the prediction. A shortcoming of this scheme is oversampling as we end up with a low-resolution version and a full-resolution difference. Without going deeper into detail, examples of pyramids are the Laplacian pyramid [72–74] and steerable pyramids [75, 76].

Despite the many approaches to analyze texture, state-of-the-art methods deal with a couple of shortcomings. When many textures have to be recognized (such as for segmentation and classification), their discriminative power rapidly drops as the number of unknown textures increases. Also, for material classification which is similar to texture classification, state-of-the-art methods obtain a poor result. Furthermore, one of the main shortcomings is their sensitivity to noise. Noise is a frequent problem in texture characterization and often causes difficulties for interpretation. However, the HVS can relatively well cope with noisy data. Since humans still outperform any vision system, it is attractive to build a system that relies on the processing principles of the HVS to deal with texture data. In the next sections, we will propose a texture model that relies on the HVS and we will examine how well our method behaves for different texture applications.

2.4 Model

In computer vision the use of visual neuroscience has often been limited to a tuning of Gabor filter banks. Little or no attention has been given to biological features of higher complexity. In 1992, von der Heydt et al. reported on the existence of a new type of complex, orientation-selective neuron in the visual cortex of higher primates which they called grating cells [77]. Grating cells respond vigorously to a grating of bars of appropriate orientation and periodicity, but in contrast to the simple cells, they hardly or don't respond to single bars (i.e., bars that do not make part of a grating). This non-linear behaviour is quite different from the spatial frequency filtering behaviour of other orientation selective cells. It is assumed, but not proven at the time of writing, that grating cells receive their input from simple cells. The main purpose of grating cells in the HVS is a fast and reliable detection of periodic patterns of a certain orientation which are common in images of human-made structures, like fabrics and tiles, and to regular natural periodic patterns, however, which are relatively rare in nature [78]. Although they represent a minority of the cells in the visual cortex (about 4% of the cells in V1 and 1.6% of the cells in cortical area V2 are identified as grating cells), their behaviour could be a vital property of a robust texture operator since they don't respond to non-texture features such as edges that do not make part of a grating or isolated pixels.

The texture features proposed in this dissertation, are based on cell responses of the human primary and secondary visual cortex. We propose to combine responses of simple cells and grating cells, and to apply particular post- and preprocessing, respectively. At first in Section 2.4.1, we explain the model of a simple cell by using a Gabor filter. Then, in Section 2.4.2 the spatial smoothing of the simple cell responses with regard to texture analysis is considered. Section 2.4.3 describes the grating cell model. In Section 2.4.4, we describe how an optimal output of the grating cell operator can be obtained.

2.4.1 Simple Cell

Daugman proposed the use of 2-dimensional isotropic real Gabor filters to model the receptive field response (i.e., the firing rate) of simple cells to a stimulus (a flashing light or a dark spot) in function of the (x, y) location of the stimulus:

$$g_{\lambda,\theta,\varphi}(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\varphi\right), \qquad (2.2)$$

where

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta. \end{cases} \qquad (2.3)$$

The parameters of this Gabor function are: σ, γ, λ, θ, and φ. The standard deviation σ of the Gaussian factor determines the effective size of the surrounding of a pixel which will be considered for further processing. The harmonic factor (i.e., the cosine) of the Gabor function has a spatial frequency of $1/\lambda$ (or a wavelength of λ pixels). The orientation of the Gabor function (counter-clockwise with respect to the x-axis) is denoted by $\theta \in [0, \pi[$. This is the angle between the dashed line (i.e., the normal to the parallel lobes of the filter in the spatial domain) and the x-axis as depicted in Figure 2.1.

The frequency bandwidth B (in octaves) can be set to a constant value to match psychovisual data. According to [71], the unknown parameter σ is determined by setting the frequency cut-off to -6 dB:

$$\sigma = \sqrt{\frac{\ln 2}{2}}\frac{\lambda}{\pi}\frac{(2^B+1)}{(2^B-1)}.$$

So, the bandwidth B in octaves is determined by the ratio σ/λ and is given by:

$$B = \log_2\left(\frac{\frac{\sigma}{\lambda}\pi + \sqrt{\frac{\ln 2}{2}}}{\frac{\sigma}{\lambda}\pi - \sqrt{\frac{\ln 2}{2}}}\right).$$

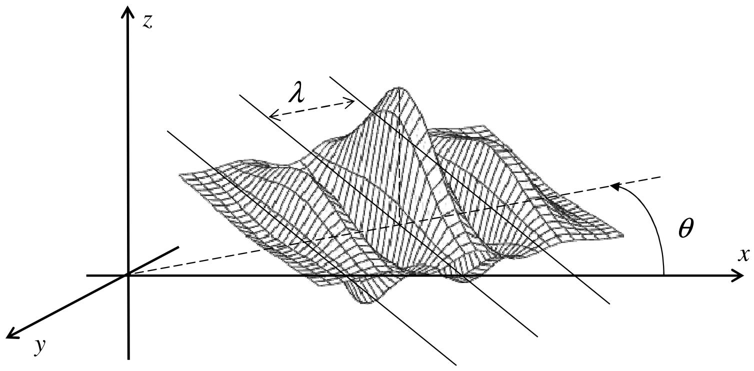


Figure 2.1: A 2-dimensional Gabor function $z = g_{\lambda,\theta,\varphi}(x, y)$ with orientation θ, wavelength λ, and phase offset $\varphi = 0$.

Experiments indicate that the frequency bandwidth B of simple cells is about one octave [79]. Thus for $B = 1$, we obtain:

$$\frac{\sigma}{\lambda} = \frac{1}{\pi}\sqrt{\frac{\ln 2}{2}}\left(\frac{2^B + 1}{2^B - 1}\right) \approx 0.56. \tag{2.4}$$

The spatial aspect ratio γ determines the eccentricity and herewith the eccentricity of the receptive field ellipse. According to Jones and Palmer, it has been found that γ varies in a limited range of $0.23 < \gamma < 0.92$ [80]. For γ-values closer to 1, the shape of the receptive field is more circular and smaller γ-values yield a narrower elliptical field. Because of this elliptical shape, elongated stimuli, such as bar and edges, are better detected. In our model, γ is set to a constant value of 0.5[1] as also suggested by Petkov and Kruizinga [81]. According to [71], the angular bandwidth Q is obtained by setting the cut-off frequency in the angular direction to -6 dB:

$$\frac{\sigma}{\gamma} = \frac{\sqrt{\ln 2}\lambda}{\sqrt{2}\pi \tan{(Q/2)}}.$$

Thus, for $\gamma = 0.5$ and $\frac{\sigma}{\lambda} = 0.56$, we have:

$$Q = 2 \arctan\left(\sqrt{\frac{\ln 2}{2}}\frac{\gamma}{\pi}\frac{\lambda}{\sigma}\right) \approx 19^\circ.$$

[1]We empirically found that values in the range of $[0.4, 0.6]$ yield a good response of the filter to textures and that small variations in this range yield a hardly noticable effect, while for lower and higher values there is a noticable effect on the filter's response. Consequently, $\gamma = 0.5$ is a good choice for texture analysis.

Finally, the phase offset φ affects the symmetry of the Gabor function (2.2). For $\varphi = 0$ and $\varphi = \pi$ the function is symmetric, while for $\varphi = \pi/2$ or $\varphi = -\pi/2$, the function is anti-symmetric, and for other values the function is asymmetric.

The response of a linear and spatially invariant filter to an arbitrary signal can be written as a *convolution product*, or simply *convolution*. In general, the convolution product of two signals f_1 and f_2 is denoted as $f_1 * f_2$ and is defined by:

$$f_1 * f_2(x) = \int_{-\infty}^{+\infty} f_1(t) f_2(x-t)dt.$$

Further, a spatially invariant linear filter L is entirely determined by its impulse response function (also called the *kernel*) h. For every signal f, we have $L(f) = f * h$.

Thus, the receptive field function of a simple cell tuned to an orientation θ and a frequency $1/\lambda$ to the luminance channel of an input image $I(x,y)$, $(x,y) \in \Omega$ (Ω denotes the visual field domain[2], but in the context of this dissertation, it is sufficient to use all pixels of the input image), is then given by the following convolution:

$$r_{\lambda,\theta,\varphi}(x,y) = \iint_{\Omega} I(s,t) g_{\lambda,\theta,\varphi}(x-s, y-t)dsdt. \tag{2.5}$$

To obtain invariance to the local average luminance of the input image, $r_{\lambda,\theta,\varphi}$ is divided by the weighted average gray value within the receptive field. Taking the Gaussian window, which defines the receptive field – see equation (2.2) – into account, the weighted average gray value $a_\lambda(x,y)$ of the surrounding pixel values is obtained by:

$$a_\lambda(x,y) = \iint_{\Omega} I(s,t) \exp\left(-\frac{(x-s)^2 + \gamma^2(y-t)^2}{2\sigma^2}\right) dsdt. \tag{2.6}$$

In order to obtain a contrast response similar to the ones measured on real cells, the response of a simple cell of the visual cortex to $I(x,y)$ is, according to Petkov and Kruizinga [82], obtained by normalizing the receptive field function $r_{\lambda,\theta,\varphi}(x,y)$ with the average gray value in the following way:

$$s_{\lambda,\theta,\varphi}(x,y) = \begin{cases} 0 & \text{if } a_\lambda(x,y) = 0 \\ \chi\left(\dfrac{\frac{r_{\lambda,\theta,\varphi}(x,y)}{a_\lambda(x,y)} R}{\frac{r_{\lambda,\theta,\varphi}(x,y)}{a_\lambda(x,y)} + C}\right) & \text{otherwise,} \end{cases} \tag{2.7}$$

[2]The visual field actually denotes the total area in which objects can be seen while focussing on a central point.

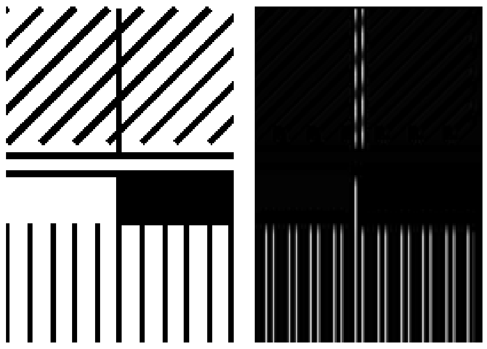


Figure 2.2: (a) An input image (height 189 pixels, width 128 pixels) (b) response to the input image of a simple cell tuned to orientation 0 and a periodicity of 2 pixels.

where R denotes the maximum response level, C is the semi-saturation constant (i.e., the stimulus required to induce a half maximum response), and $\chi(t) = t$ for $t \geq 0$ and $\chi(t) = 0$ for $t < 0$. The semi-saturation constant in the denominator prevents very low-contrast signals from being normalized to a large value.

Figure 2.2 depicts the response of the simple cell defined by (2.7) with orientation $\theta = 0$ and periodicity $\lambda = 2$ pixels. As can be seen in the response image, the simple cell detects every edge (black to white and vice versa) to the normal of the orientation.

The above figure illustrates that the output of the simple cell, as given by equation (2.7), detects the presence of edges. However, since a texture typically contains many edges, the output of a simple cell is not a suitable representation for texture discrimination. Ideally, for texture analysis and classification purposes, the output of a description method should be constant for a given texture. It can easily be shown that the simple cell response to a sinusoidal input signal is not a constant. Functionally, the output of a simple cell is a real Gabor function. For simplicity, consider the real part g_r of a 1-dimensional Gabor function:

$$g_r(x) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma^2}\right) \cos\left(2\pi F x\right).$$

A cosine with frequency $F = 1/\lambda$ in the frequency domain is given by its

Fourier transform [83]:

$$F_s(u) = \frac{1}{2}\left\{\delta(u-F) + \delta(u+F)\right\},$$

where $\delta(x)$ denotes the *Dirac delta function* [83].

The Fourier transform F_g of a real Gabor function g_r is given by the convolution of the Fourier transform of the Gaussian function and the Fourier transform of the sinusoidal function:

$$\begin{aligned} F_g(u) &= \sigma\sqrt{2\pi}\exp\left(-2\pi^2u^2\sigma^2\right) * \frac{1}{2}(\delta(u-F) + \delta(u+F)) \\ &= \frac{\sigma\sqrt{2\pi}}{2}\left\{\exp\left(-2\pi^2(u-F)^2\sigma^2\right) + \exp\left(-2\pi^2(u+F)^2\sigma^2\right)\right\}. \end{aligned}$$

In the frequency domain, the real Gabor filter's response to the sinusoidal signal is given by the multiplication of $F_g(u)$ and $F_s(u)$. Given the fact that $\delta(u) = 0$ for $u \neq 0$, the convolution of a harmonic signal with a Gabor function can be written in the frequency domain as:

$$\begin{aligned} F_s(u)F_g(u) =& \frac{1}{4}[(\delta(u-F) + \delta(u+F)) \\ & \sigma\sqrt{2\pi}(\exp(-2\pi^2(u-F)^2\sigma^2) + \exp(-2\pi^2(u+F)^2\sigma^2))] \\ =& \frac{1}{4}[(\exp(-2\pi^2 4F^2\sigma^2) + 1)(\delta(u-F) + \delta(u+F)) \\ & \sigma\sqrt{2\pi}((\exp(2\pi^2(u-F)^2\sigma^2) + \exp(2\pi^2(u+F)^2\sigma^2))] \\ =& \frac{c}{4}[\delta(u-F) + \delta(u+F)], \end{aligned}$$

where $c = \sigma\sqrt{2\pi}\exp(-2\pi^2 4F^2\sigma^2) + 1$. Thus, $F_s(u)F_g(u)$ produces again a cosine $\frac{c}{2}\cos\left(2\pi Fx\right)$ in the spatial domain. The response of a simple cell, which is given by equation (2.7), to a sinusoidal signal produces a sinusoidal in the spatial domain. The variations of this sinusoidal are not ideal, e.g., for classification since a constant value is preferred. Simple cell responses are edge detectors, but for texture analysis some post-processing is required in order to obtain a constant response. To achieve the latter, we propose a spatial smoothing as described in the next section.

2.4.2 Spatially Smoothed Responses

Hall and Hall describe the existence of sustained channels in the visual system indicating that the HVS not only considers pixels in the field of view, but also pixels in the vicinity [84]. This effect can be obtained by applying a smoothing on the obtained simple cell responses. Furthermore, textures which do not have sufficiently narrow bandwidths may suffer from leakage. The effects of leakage can also be reduced by post-filtering the simple cell responses (2.7) with a Gaussian filter that has the same shape as the corresponding filter, but with a greater spatial extent [33,71]. Therefore, smoothed Gabor responses are known to improve the performance for texture analysis because it suppresses large variations of the filter responses in areas which belong to the same texture. However, too much smoothing can have a negative effect (especially on the localization of texture region edges).

Therefore, we propose to spatially smooth the simple cell responses which are obtained from a symmetric Gabor function, i.e., $s_{\lambda,\theta,0}$, by using a Gaussian filter with the same shape but a greater variance $\sigma' > \sigma$:

$$\tilde{s}_{\lambda,\theta}(x,y) = [s_{\lambda,\theta,0} * gauss](x,y), \tag{2.8}$$

where

$$gauss(x,y) = \frac{1}{2\pi\sigma'^2} \exp\left(-\frac{x^2+\gamma y^2}{2\sigma'^2}\right).$$

Different values for σ' have been investigated. Empirically, we observed that the best results where obtained for $\sigma' = 2\sigma$, i.e., by applying a Gaussian smoothing with twice the standard deviation of the Gabor filter.

2.4.3 Grating Cell

There exist different computational models of grating cells for oriented texture analysis. Kruizinga and Petkov use the above model of simple cells, see equation (2.7), to model center-on ($\varphi = 0$) and center-off ($\varphi = \pi$) receptive fields (in analogy to the retinal ganglion cells) [82]. Center-on and center-off receptive fields are alternately activated for grating patterns of corresponding orientation and frequency. Zhang et al. [85] introduce multi-valued logic in the model of Kruizinga and Petkov. However, this results into 2 new thresholds for which no optimal values are suggested, nor how they can be obtained. du Buf uses the notion of a grouping operator [86, 87]. The grouping is done in 2 steps: (i) detection of events along a line and (ii) amplitude completion by a filling-in process. Instead of simple cell responses, Lourens et al. use complex cell outputs as input for their grating cell operator [88, 89]. The authors model a simple cell by a complex Gabor filter and a complex cell operator

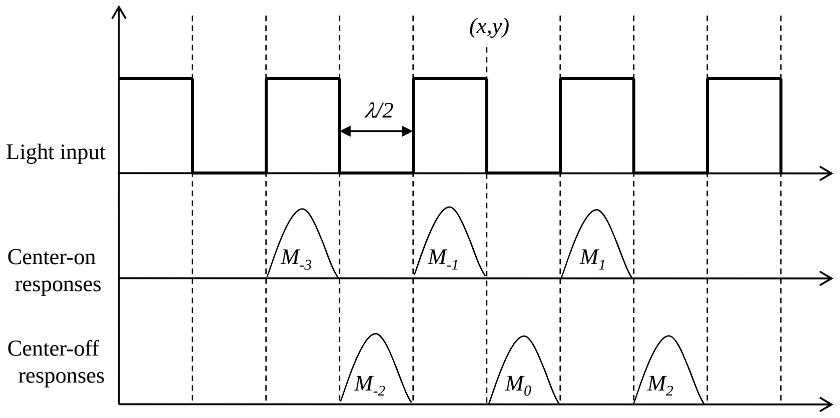


Figure 2.3: Luminance distribution along a normal to a set of three square bars and the distribution of the computed responses of center-on and center-off cells along this line.

then calculates the amplitude of the complex values. According to the authors, their grating cell model has similar response profiles as monkey grating cells as measured by von der Heydt et al.

In this dissertation, we applied the model of Kruizinga and Petkov since we experimently observed that it responds well to gratings. This model detects the presence of a grating if there are at least 3 parallel bars with the preferred orientation and periodicity. This non-linear behavior is modeled in two stages:

(i) the calculation of the activity of a grating subunit $q_{\lambda,\theta}(x, y)$ with a preferred orientation θ and frequency $1/\lambda$, see equation (2.9),

(ii) the summation of the responses for a given θ and λ, see equation (2.10).

A grating subunit $q_{\lambda,\theta}$ will be activated if the corresponding receptive field functions $r_{\lambda,\theta,\varphi_n}$ for $\varphi_n = 0$ $(n = -3, -1, 1)$ and $\varphi_n = \pi$ $(n \in \{-2, 0, 2\})$, are alternately activated in intervals of length $\lambda/2$ along a line segment of length 3λ centered on point (x, y) as illustrated in Figure 2.3.

A grating subunit takes as input the simple cell outputs defined in equation (2.7). At first, for a given λ, θ, and n, Kruizinga and Petkov have defined the quantities $M_{\lambda,\theta,n}$ and $N_{\lambda,\theta}$, which are computed as follows:

$$\begin{cases} M_{\lambda,\theta,n}(x,y) = \max_{(u,v)}\{s_{\lambda,\theta,\varphi_n}(u,v)\} \\ N_{\lambda,\theta}(x,y) = \max\{M_{\lambda,\theta,n}(x,y)|n = -3, -2, ..., 2\}, \end{cases}$$

where u and v satisfy the condition:

$$\begin{cases} n\frac{\lambda}{2}\cos\theta \le u - x < (n+1)\frac{\lambda}{2}\cos\theta \\ n\frac{\lambda}{2}\sin\theta \le v - y < (n+1)\frac{\lambda}{2}\sin\theta. \end{cases}$$

$M_{\lambda,\theta,n}$ is related to the maximum activity of one sort of simple cells (center-on or center-off) along the line segment of length 3λ passing through the point (x, y) in orientation θ. For instance, $M_{\lambda,\theta,-3}$ is the maximum activity of center-on simple cells in the corresponding interval (i.e., the leftmost interval) of length $\lambda/2$, $M_{\lambda,\theta,-2}$ is the maximum activity of center-off simple cells in the adjacent interval, etc. Center-on and center-off simple cell activities are thus alternately used in consecutive intervals. The quantity $N_{\lambda,\theta}$ is the maximum among the above interval maxima.

In case of a grating, the quantities $M_{\lambda,\theta,n}$ are approximately the same in each interval. Therefore, the activity of a grating subunit $q_{\lambda,\theta}$ is computed as:

$$q_{\lambda,\theta}(x,y) = \begin{cases} 1 & \text{if } \forall n, M_{\lambda,\theta,n}(x,y) \ge \rho N_{\lambda,\theta}(x,y) \\ 0 & \text{if } \exists n, M_{\lambda,\theta,n}(x,y) < \rho N_{\lambda,\theta}(x,y), \end{cases} \tag{2.9}$$

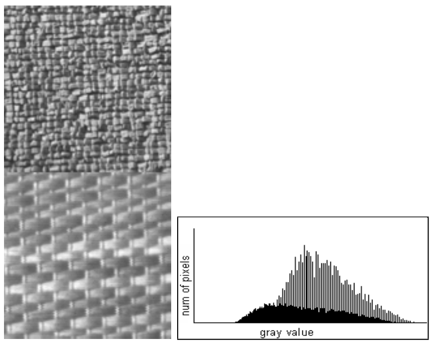
where $0 < \rho < 1$ is a threshold value in the proximity of 1. We have found that $\rho = 0.9$, as also suggested by Petkov and Kruizinga, yields good results. Values that are too close to 1 yield less activations of $q_{\lambda,\theta}$, while lower values yield more false activations.

In the final stage, the response $w_{\lambda,\theta}$ of a grating cell whose receptive field is centered on point (x, y) and tuned to frequency $1/\lambda$ and orientation θ to the normal of the grating, is computed by a weighted summation of the responses of the subunits $q_{\lambda,\theta}$. The operator is made symmetric by also considering the opposite direction, i.e., $\theta + \pi$:

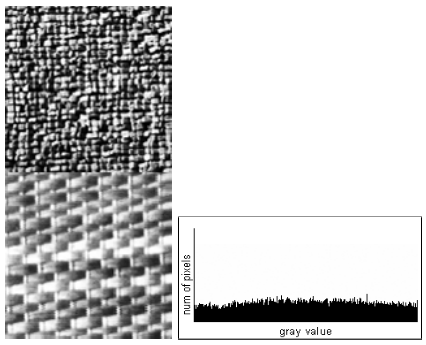
$$w_{\lambda,\theta}(x,y) = \iint_{\Omega} W(x,y)\left(q_{\lambda,\theta}(s,t) + q_{\lambda,\theta+\pi}(s,t)\right) dsdt, \tag{2.10}$$

where $W(x,y) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-s)^2+(y-t)^2}{2(\beta\sigma)^2}\right)$. The parameter β determines the size of the area over which effective summation takes place. According to Petkov and Kruizinga, a value of $\beta = 5$ results in a good approximation of the spatial summation properties of grating cells.

The output of the grating cell operator (2.10) depends on the output of the simple cell (2.7) which is actually an edge detector. A straightforward method to enhance this edge detection and thus also the output of the grating cell operator, is explained in the next section.



(a)



(b)

Figure 2.4: (a) Original and (b) enhanced image with their respective luminance histograms

2.4.4 Enhanced Grating Cell Operator

The HVS is more sensitive to contrast than absolute luminance. In order to better distinguish possible salient texture-specific periodicities and to obtain an improved texture discrimination, we increase the contrast in an image. As such, an enhanced image $\overline{I}(x, y)$ is created by applying a histogram equalization to the original input image $I(x, y)$. Histogram equalization is a well-known technique that rescales the range of the pixel values to produce an image whose pixel values are more uniformly distributed which results in an image with a higher contrast as shown in Figure 2.4. Furthermore, histogram equalization also seems to be used in biological neural networks. This has been proved in particular in the fly retina [90].

For each gray-level j in the original image $I(x, y)$, the new value $\overline{j}$ is calculated as follows:

$$\overline{j} = l \sum_{i=0}^{j} \frac{n_i}{n}, \tag{2.11}$$

where l is the maximum gray-level, n the total number of pixels, n_j the number of occurrences of gray-level j in $I(x, y)$, and $j, \overline{j}, l, n, n_i \in \mathbb{N}$.

The enhanced grating cell features $\overline{w}_{\lambda,\theta}(x, y)$ are obtained by substituting the enhanced image $\overline{I}(x, y)$ in equations (2.5) and (2.6). Remark that the effect of this enhancement on images having a uniform luminance histogram, will be nihil. The effect of the histogram equalization on the grating cell operator is exemplified in Figure 2.5 which depicts the grating cell and enhanced

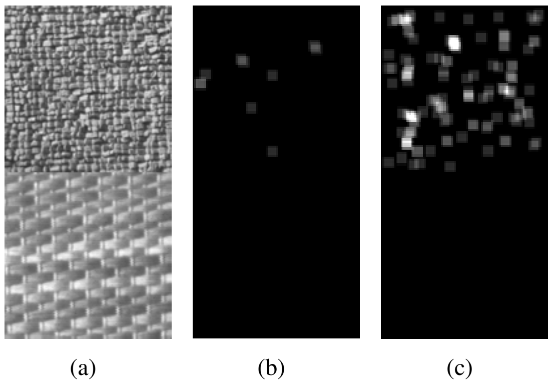


Figure 2.5: (a) Input image with 2 textures, (b) the grating cell responses, and (c) the enhanced grating cell responses for orientation $\theta = \pi/4$ and wavelength $\lambda = 2\sqrt{2}$. As can be seen, the enhanced grating cell operator gives a better distinction between the upper and lower textures.

grating cell response for orientation $\pi/4$ and a wavelength of $2\sqrt{2}$ pixels. As can be seen in Figure 2.5(b), there is almost no grating cell response to the upper texture and no response to the lower texture. Relying on this grating cell operator, it is thus difficult to distinguish these two textures. Obviously, the enhanced grating cell response (for the given orientation and frequency) in Figure 2.5(c) shows now a more clear distinction between the two textures since the response to the upper texture is now much stronger while the response to the lower texture remains unchanged.

2.5 Parameter Selection

Now that we have outlined the computational models, we explain how the texture features are computed. Since the output of one filter is not sufficient to characterize a texture, features are obtained from arrays of filters (i.e., filter banks) so that the input signal is decomposed into multiple elements. These elements are then arranged in a vector which is called the *feature vector*. Each element in the feature vector then accounts for the response of a unique filter tuned to a specific orientation and frequency.

Keeping in mind that the bandwidth of the simple cells is 1 octave, see Section (2.4.1), the center frequencies should be chosen accordingly to minimize the overlap of the filters and to maximize coverage. Related work [91–93] that applies Gabor filter banks for texture analysis, suggests the use of frequencies $\sqrt{2}(1, 2, 4, 8, ..N/4)$ cycles per image for texture images of resolution $N \times N$ (where N is a power of 2) pixels. However, for images of 512×512 pixels we omit the lower frequencies (patterns of $\lambda \geq 32\sqrt{2}$ are too large to be perceived as 'texture') so that we use the remaining wavelengths of $\lambda = \sqrt{2}, 2\sqrt{2}, 4\sqrt{2}, 8\sqrt{2}$ and $16\sqrt{2}$ pixels.

Further, we choose 8 orientations, i.e., (θ=0, $\frac{\pi}{8}$,..,$\frac{7\pi}{8}$), which results in an orientation selectivity of $22.5°$. Remark that if we would take a smaller number of orientations, e.g., 6 instead of 8, there will be orientations to which none of the channels of the filter bank will respond sufficiently (as the orientation bandwidth of the Gabor filter is approximately $19°$) and this will have a negative effect on the discrimination performance for textures that are dominated by the concerned orientations. This means that the discrimination performance will depend on the choice of the texture. The use of 8 orientations and 5 frequencies results in a 40-dimensional feature vector that is obtained from one filter bank.

An important aspect for implementing a Gabor filter, is the size of the neighborhood window to compute the filter's response. This aspect is examined by Garcia and Puig [94] who evaluated various texture analysis methods with different window sizes. They obtain significantly better results by using optimal window sizes. In our model, the window size k is related to the standard deviation of the Gaussian (and thus to the bandwidth of the filter), i.e., $k = 2\sigma + 1$, as also suggested by Khan et al. [95]. Taking equation (2.4) into account, the size of the window is obtained by: $k = nint(1.12\lambda + 1)$, where $nint$ denotes the nearest integer function. Consequently, the size of the window changes according to the periodicity of the texture: a smaller window for small periodicities (high frequencies) and a larger window for the larger periodicities (low frequencies).

2.6 Unsupervised Texture Segmentation

As mentioned in Section 2.2.3, a segmentation process may be the first step in processing a user query in a CVIR application. However, texture segmentation is a difficult problem since one usually does not know a priori how many and what types of textures exist in an image. Ideally, we should know how many and which types of textures there are present in an image. Suppose we have the priori information about the textures in Figure 2.4(a), i.e., that there are 2 different textures and we have the features of each texture class, but we do

nr. training samples	1024	768	512	256	128	64	32
nr. test samples	1024	256	512	768	896	960	992
% misclassifications	0.0	1.0	2.0	2.6	4.7	17.50	78

Table 2.1: Supervised classification of textures using a linear SVM. The accuracy of the classifier decreases as the number of training samples decreases.

not know where they appear in the image. To obtain the spatial arrangement of the textures in the image, we have trained a supervised classifier in order to assign a label to the textures in the image. Therefore, we have computed 1024 texture features (enhanced grating cell responses and spatially smoothed Gabor responses) as explained in the previous section, and we have used them to train a linear support vector machine[3] (SVM) [97]. Once trained, this SVM can then be used to classify the texture features accordingly. As can be seen in Table 2.1, the accuracy of the classifier is highly dependent on the number of training samples. All the 1024 training samples are correctly classified (i.e., the training set is equal to the test set) as can be seen in the first column. However, as the number of training features decreases, the number of misclassifications of the remaining test samples increases because the classifier is overfitted to the training data.

The above exemplifies why the use of unsupervised machine learning techniques (i.e., clustering) is favored over the use of supervised machine learning. Consequently, k-means clustering [98] has been widely applied. However, k-means is not fully unsupervised since a value for k to specify the number of distinct textures in the image has to be given. Other methods that try to find an optimal method for k, make assumptions about the underlying distribution of the data [99, 100]. The Bayesian information criterion [101] or Akaike's information criterion [102] is then calculated on data sets which are constructed by a set of Gaussian distributions. However, it has been shown that many textures, such as the Brodatz textures, have not been drawn from have been drawn from a Gaussian distribution [60]. Therefore, we recommend the use of Self-Organizing Maps (SOM) for the unsupervised segmentation of textured images [59, 103]. This algorithm operates fully unsupervised since no value or estimate for the number of textures has to be provided. Furthermore, SOM neural networks are particularly well-suited for the combined task of mapping the high-dimensional and non-linear data distribution to a low-dimensional plane while conserving the local neighborhood relations. An important prop-

[3]The linear SVM is trained using the LIBSVM package [96] and the cost of the constrain violation is set to 1.

erty of the SOM is that it clusters similar data vectors and projects dissimilar ones far from each other on the map (with respect to some predefined metric and the topology of the map). Generally, segmenting a textured image using the SOM algorithm consists of the following steps:

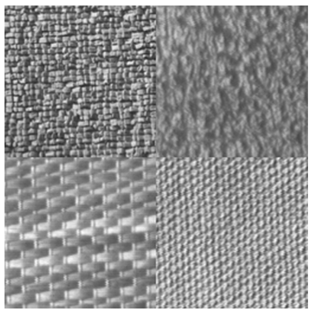
(i) *Training*. The SOM is trained using texture features obtained from an input image.

(ii) *Node clustering*. Identification of SOM node clusters based on the distances between their model vectors.

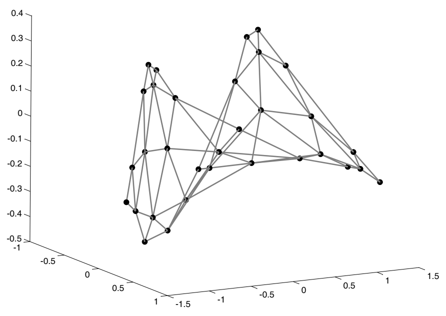
(iii) *Best matching unit (BMU) calculation*. The BMU of each feature vector obtained from the input image is calculated. This way, the pixels that correspond to the feature vectors are assigned to a cluster of one or more SOM nodes. These clusters then represent the regions in the image.

A result of the training phase is that pixels belonging to the same texture are assigned to the same or adjacent nodes. This means that nodes that are close to each other (with respect to the topology of the map), have been assigned similar feature vectors, while distant nodes are related to dissimilar textures. This effect is illustrated in Figure 2.6(c) which depicts the unified distance matrix of a 8×4 SOM trained with texture feature vectors obtained from an image containing 4 different textures. For each n-dimensional row or column of SOM nodes N_i, $i \in \{0..n-1\}$, the unified distance matrix contains a vector u of dimension $2n - 1$ so that $u_j,\ j = 1, 3, ...$ is the distance between N_i and N_{i-1}, and $u_j,\ j = 0, 2, ...$ is the mean of the surrounding distances. This way, texture features from different textures are assigned to different clusters and a segmentation is obtained. Four clusters can be distinguished in Figure 2.6 (i.e., the blue regions near the corners of the map) and each cluster corresponds to one of the four textures.

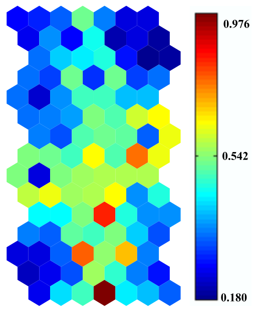
It is obvious that the obtained image segmentation depends on the quality of the trained SOM. However, measuring the quality of the SOM is not trivial. The SOM quality is usually measured with the following criteria: *quantization error* and *topographic error*. The quantization error is the mean distance between each data point and its BMU and measures the resolution preservation. The latter describes how accurately the SOM neurons respond to the given data set. For example, if the model vector of the BMU calculated for a given training vector is exactly the same, the error is 0. Normally, the number of data vectors exceeds the number of neurons and the quantization error is thus always different from 0. The topographic error represents the proportion of all data for which the first and second BMU are not adjacent. These measures are



(a)



(b)



(c)

Figure 2.6: (a) An input image containing 4 different textures. (b) Visualization of a 8×4 SOM trained with the proposed texture features obtained from the input image. The SOM is plotted along the biggest three principal components of the training data. (c) The unified distance matrix of the SOM. As can be seen, 4 clusters representing the 4 textures can be distinguished (blue regions).

thus data-dependent since they measure the map in terms of the given data, and the best measures are obtained in case of overfitting to the data. Overfitting might lead to oversegmentation and should thus be avoided. Consequently, these quality measures should be interpreted as an indication rather than an exact measure. Furthermore, different parameters for a SOM (e.g., dimension, lattice, grid, neighborhood function) will also affect the quantization and topographic error. Therefore, we have chosen to use a fixed dimension, lattice, and neighborhood for the SOM because optimal values for these parameters are not generic due to the data dependency of the quality measures.

The accuracy, i.e., the quality, of the obtained image segmentation will be defined by the percentage of correctly assigned pixels. This measure can be computed by the fraction of the number of true positives (t_p) and the sum of the true positives with the number of false negatives (f_n). In this context, a false negative is a pixel that is not assigned to the correct image region according to the ground truth. The accuracy corresponds to the average *recall*, i.e., the proportion of relevant pixels that are assigned to the correct image region (see eq. 2.12). Remark that the accuracy is not related to the *precision*, which is defined by the fraction of the number of true positives (t_p) and the sum of the number of true positives with the number of false positives (f_p) (see eq. 2.13).

$$\text{Recall} = \frac{t_p}{t_p + f_n}. \tag{2.12}$$

$$\text{Precision} = \frac{t_p}{t_p + f_p}. \tag{2.13}$$

The SOMs are trained using the well-known batch-training algorithm which is, in contrast to the sequential training algorithm, much faster to calculate and the results are as just as good [59]. The SOMs are created using the SOM Toolbox [104] which is a function package for Matlab that implements the SOM algorithm using the Euclidean distance. Further settings of the SOM are: (i) the lattice structure (grid) is hexagonal (each SOM node has 6 direct neighbors), and (ii) the topological structure of the map is a 2-dimensional sheet (experiments showed that the choice of neither the topological structure(e.g., a sheet, sphere, or cylinder) nor the lattice structure (e.g., rectangular or hexagonal) has significant impact on the obtained segmentation).

In the remainder of this section, we perform texture segmentation experiments and we show that the combination of enhanced grating cell features with Gaussian smoothed Gabor responses obtains the best segmentation results.

2.6.1 Experiment 1

In this first experiment, we test different Gabor-based texture features for segmentation. Further, two types of unsupervised classifiers have been used: a SOM- and a k-means-based classifier. Feature extraction is applied on composite images containing textures of the Brodatz album [61]: D6, D8, D24, D55, D68, D77, D84, and D95. For the k-means algorithm, the value of k is set to the number of distinct textures in the image. These textures are then used to create mosaic images that contain different textures. Five, five, three and three mosaic images of size 512×512 pixels containing two, four, five, and nine texture regions are created, respectively.

From these textured mosaics, the following feature types are calculated: (i) Gaussian smoothed Gabor responses G_s, (ii) grating cell responses (Grat), (iii) enhanced grating cell responses (Grat$_{enh}$), (iv) Gaussian smoothed real Gabor with grating cell responses (G$_s$ + Grat), and (v) Gaussian smoothed real Gabor with enhanced grating cell responses (G$_s$ + Grat$_{enh}$). The texture features are calculated from individual pixels and not from patches. This way, no pixel adjacency information has been used in the clustering process.

Scaling of the feature vectors that consist of different feature types (e.g., G$_s$ + Grat) is of special importance since our classifiers (both the k-means and SOM) use the Euclidean metric to measure the distances between feature vectors. Otherwise, bigger variables tend to dominate the others. Therefore, the features are normalized before they are combined in one large feature vector. For a feature vector x, each element x_i is normalized by: $x_i' = x_i / \|x\|$.

Using the SOM-algorithm, maps of different sizes have been created. We train 4×2, 4×4, 8×7, and 8×9 maps for the segmentation of images containing two, four, five, and nine textures, respectively. Remark that larger maps can also be used for a lower number of textures. A general rule is that a higher number of nodes results in better classification results, but a side effect is that overclassification may occur. On the other hand, small-sized maps are more attractive because of their lower computational cost during the training phase.

Figure 2.7 illustrates the pixelwise unsupervised segmentation of some mosaic images (containing 2, 4, 5 and 9 textures) using the SOM-based clustering algorithm – remark that there are actually 8 different textures in Figure 2.7(m) because 2 regions consist of the same texture. The recall of the segmentation obtained by the k-means and SOM-based clustering algorithm are listed in Table 2.2 and Table 2.3, respectively. For both clustering methods, we notice that the enhanced grating cell features Grat$_{enh}$ give a slight rise in recall compared to grating cell responses (Grat). However, Grat and Grat$_{enh}$ are not discriminative enough for segmenting images containing Brodatz textures. The combination of enhanced grating cells with Gaussian smoothed Gabor responses (G$_s$ + Grat$_{enh}$) achieves the best segmentation. However, we notice that the difference between G$_s$ and G$_s$ + Grat or G$_s$ + Grat$_{enh}$ is relatively small for 2 textures. But, as the number of textures in an image increases, the difference becomes larger. Furthermore, we observe that the SOM-based classifier generally yields better segmentation results than the k-means classifier. The fact that by the SOM the distance of each input from all of the model vectors (weighted by the neighborhood) is taken into account instead of just the closest one for k-means, can account for this phenomenon. The latter property of the SOM causes that the SOM follows, to a certain extent, more closely the distribution of the data set in the input space. This is because centroids used in a SOM have a predetermined topographic ordering relationship (the centroids that are

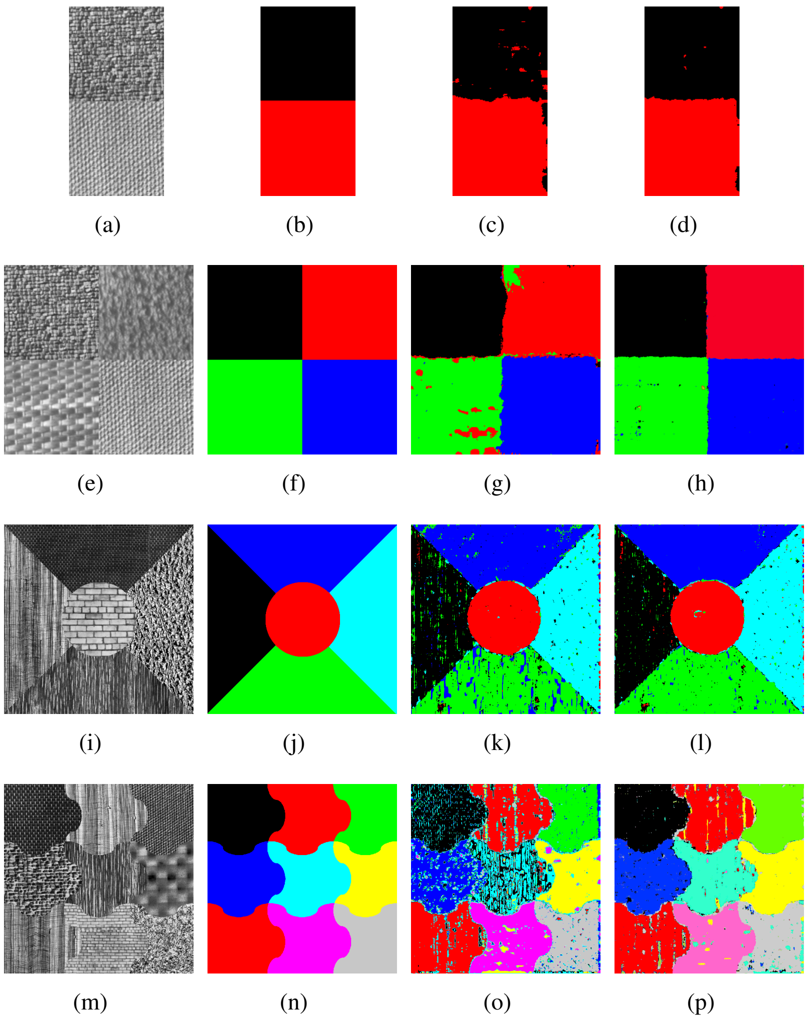


Figure 2.7: (a, e, i, m) Original image, (b, f, j, n) ground truth, pixelwise SOM-based clustering with: (c, g, k, o) smoothed Gabor responses, (d, h, l, p) enhanced grating cell features + smoothed Gabor responses.

close to each other in the SOM grid are more closely related to each other than to the centroids that are further away).

regions	G_s	Grat	G_s + Grat	Grat$_{enh}$	G_s + Grat$_{enh}$
2	94.1	73.5	94.8	78.5	95.1
4	83.4	54.3	85.2	66.0	88.7
5	57.5	50.3	73.1	52.4	86.1
9	51.7	18.3	71.8	27.8	85.7

Table 2.2: Recall in % of the pixelwise k-means clustering of Brodatz textures.

regions	G_s	Grat	G_s + Grat	Grat$_{enh}$	G_s + Grat$_{enh}$
2	96.9	85.6	97.0	88.7	97.2
4	85.0	64.6	87.5	71.7	90.3
5	84.8	65.2	87.5	66.6	89.3
9	81.4	55.1	81.5	59.2	85.9

Table 2.3: Recall in % of the pixelwise SOM-based clustering of Brodatz textures.

2.6.2 Experiment 2

In this experiment, we calculate the proposed texture features, i.e., enhanced grating cells with Gaussian smoothed Gabor responses (G_s + Grat$_{enh}$), from patches of various sizes. Image patches of size 2×2, 4×4, 8×8 and 16×16 pixels have been used for calculating the texture features. Since square patches are used, only mosaic images without straight region borders are considered – i.e., the images of the previous experiment that containing 5 and 9 textures. As can be seen in Table 2.4, we notice that for mosaic images containing 5 textures, the gain of using 2×2 and 16×16 patches is almost negligible. For images consisting of 9 texture regions, the gain is even negative for patches of 16×16 pixels. The latter is caused by the loss of detail near the borders between adjacent texture regions. The fact that for both mosaic image sets patches of 4×4 pixels obtain the highest gain, is rather coincidental because the decisive factor is the shape and positioning of the region borders. Therefore, we stress out that the numbers in Table 2.4 are purely indicative. The main conclusion is that bigger patch sizes might involve detail loss near the borders of texture regions.

In the second part of this experiment, we compute G_s + Grat$_{enh}$ features from patches of size 4×4 pixels. Patches of size 4×4 are chosen as a compromise between computation time and detail[4]. We compare these results with the seg-

[4]The computation of the texture features of a 512×512 pixels image take about $2'16''$.

regions	2×2	4×4	8×8	16×16
5	0.3	4.4	3.0	1.6
9	0.2	4.9	1.1	-1.1

Table 2.4: Recall gain (in %) of the SOM-based clustering of Brodatz textures using patches of different sizes.

regions	$\mathrm{G}_s + \mathrm{Grat}_{enh}$	GMRF_{1-7}
2	98.9	89.0
4	95.1	83.6
5	93.7	72.4
9	90.8	70.5

Table 2.5: Recall in % of the SOM-based clustering of 4×4 patches of Brodatz textures using GMRF and the proposed texture features.

mentations obtained using GMRF features. The GMRF features are obtained from GMRF models of order 1 to 7 concatenated into one 73-dimensional feature vector (GMRF_{1-7}) – the combined use of these models performs clearly better than the individual models, see Ojala et al. [105]. The implementation of the GMRF features is obtained from the MeasTex site [106] and the features are computed using the standard symmetric masks. Clustering is performed using a SOM of the same dimensions as in the previous experiment. This is exemplified by a mosaic image of nine texture regions in Figure 2.8. Introducing pixel adjacency also boosts the segmentation accuracy for images containing 2 or 4 texture regions (Table 2.5). The low performance of the GMRF features is mainly due to the relatively high size of the mask which causes detail loss near the borders of the texture regions. More, as the number of textures increases, the SOM-based clustering algorithm has difficulties to discriminate the GMRF features.

2.6.3 Experiment 3

To investigate the application of the texture features for segmenting scenery images, we collected real-life textures from the World Wide Web (WWW) (see

Remark that besides the computation time of extra texture features, also the classification time will increase.

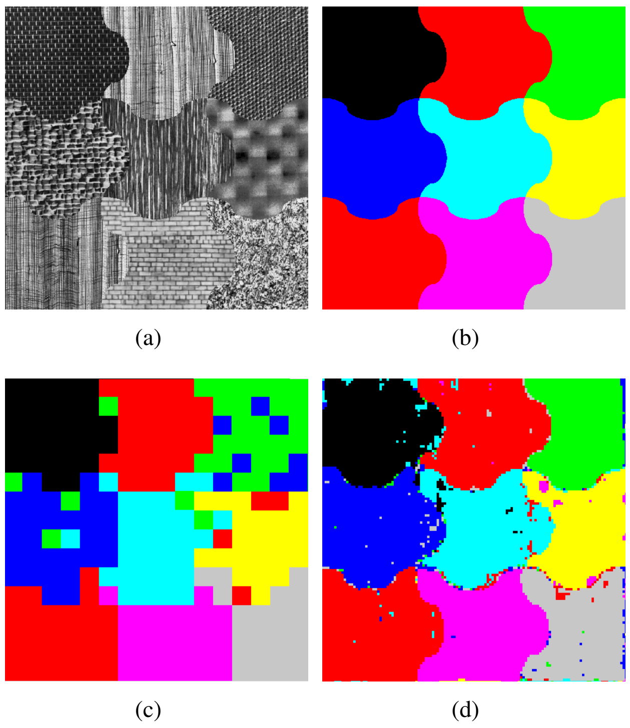


Figure 2.8: (a) Original image, (b) ground truth, SOM-based clustering with: (c) $GMRF_{1-7}$ features, (d) $G_s + Grat_{enh}$ extracted from patches of 4×4 pixels.

also Section 2.7) for testing instead of using Brodatz textures. It is important to remark that in contrast to the gray-scale textures from the Brodatz album, the intra-variation in terms of orientation and scale of a natural texture in scenery images is much higher. The latter is exemplified in Figure 2.9 which consists of four different grass textures. As can be seen, it is even for the human eye hard to distinguish the upper two textures, but the difference with the grass textures at the bottom of the image is much larger. Nevertheless, our segmentation algorithm is, to a certain extent, still capable to distinguish them, even using a small-sized (4×4) SOM.

We conduct segmentation experiments on mosaic texture images of size 512×512 pixels containing 4, 5 and 9 different texture regions. Next to the $\mathrm{G}_s + \mathrm{Grat}_{enh}$ texture information, we also introduce color information by adding the color features (which consist of 3 dimensions) to a feature vector. Both normalized HSI and normalized color opponent values (COV) have been used (see Appendix B). As in the previous experiment, the features are extracted

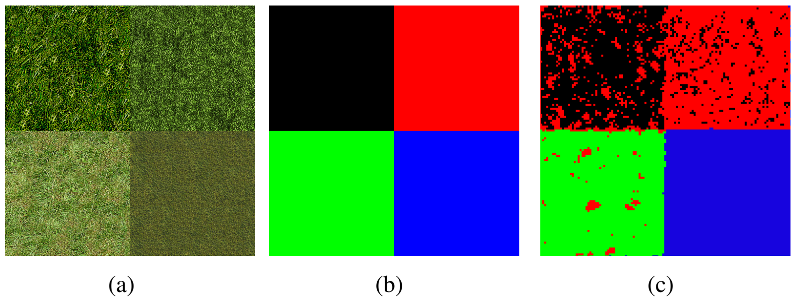


Figure 2.9: (a) Original image of 4 similar grass textures, (b) ground truth, (c) SOM-based segmentation using $\mathrm{G}_s + \mathrm{Grat}_{enh}$ texture features extracted from patches of size 4×4 pixels.

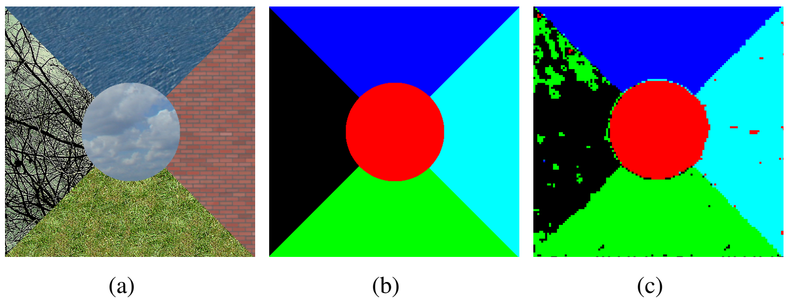


Figure 2.10: (a) Original image with 5 real-life textures, (b) ground truth, (c) SOM-based clustering of G_s+ Grat_{enh} texture features + COV color information.

from patches of 4×4 pixels.

The segmentation results listed in Table 2.6 indicate that the performance of the $\mathrm{G}_s + \mathrm{Grat}_{enh}$ texture features drops about 4 to 8% compared to the segmentation results using the Brodatz textures in Section 2.6.2. The latter is caused by the higher variations (of orientation and/or periodicity) inside the natural texture, as exemplified in the branches texture in Figure 2.10. Further, we remark that introducing color information gives a small boost in the segmentation results (up to 5% for segmentating 5 textures). In our experiments, the combination of $\mathrm{G}_s + \mathrm{Grat}_{enh}$ with COV is slightly better than $\mathrm{G}_s + \mathrm{Grat}_{enh}$ with HSI information, but using color information alone is not sufficient to discriminate the image regions.

regions	COV	HSI	$\mathrm{G}_s + \mathrm{Grat}_{enh}$	COV + $\mathrm{G}_s + \mathrm{Grat}_{enh}$	HSI + $\mathrm{G}_s + \mathrm{Grat}_{enh}$
4	0.75	0.76	0.89	0.92	0.89
5	0.74	0.69	0.89	0.94	0.93
9	0.53	0.49	0.83	0.91	0.91

Table 2.6: Recall in % of the SOM-based clustering of real-life textures.

2.7 Texture Classification

Given a set of known textures, texture classification involves deciding what texture class an observed sample belongs to. In contrast to the clustering algorithm for segmentation, the training data for classification are labeled. The approach presented in this dissertation is semi-supervised in the sense that no other a priori knowledge than the labels of the training data are used in the decision process. Semi-supervised learning using SOMs is similar to the SOM clustering algorithm for segmentation, but instead of using unlabeled training data, labeled training data are used. Generally, texture classification using SOM neural networks consists of the following steps:

(i) *Data labeling*. The training data are extracted from a set of known texture classes and are labeled. This means that for each training vector, a label that identifies the texture class, is assigned.

(ii) *Training*. The SOM is trained with the labeled training data. The training algorithm is identical to unsupervised learning, this means that the labels are not used in this stage.

(iii) *BMU calculation*. After the training phase, for each feature vector of the training set, the BMU is calculated. As a result, the feature vectors of a texture class are assigned to the same or neighboring SOM nodes.

(iv) *Node labeling*. For each node on the grid of the trained SOM, a label is assigned. This process is based on the label(s) of the related training data. Different strategies can be applied as will explained later.

(v) *Mapping*. For each feature vector obtained from the observed (i.e., the unknown) data samples, the BMU is calculated. The label of this BMU is then assigned to the unknown data sample.

The same remarks about the quality of the SOM also apply now. Even more, overfitting to the training data implies that the SOM will have a poor predictive performance.

SOM nodes that were never winning nodes in the BMU calculation step, obtain the same label as their closest neighbor (due to the topology preserving property of the SOM). When there are no misclassifications, the node labeling step is straightforward: the label of a node's related training vectors unambiguously identifies the node label. In the other case (i.e., there are training vectors from different texture classes assigned to the same node), a decision has to be made. Consider, e.g., that 1000 training vectors are related to a specific SOM node and 3 training vectors have a different label than the other ones. Then, it is obvious that this node should get its label from the remaining 997 vectors. However, this method is not so evident if the number of misclassifications is much higher. Furthermore, we have experienced that the number of misclassifications increases with the number of texture classes in the training data. By increasing the dimensions of the SOM, the number of misclassifications can decrease. Though, a larger map will also increase the computation time. However, a larger SOM won't prevent that some nodes are 'contaminated', i.e., that a node is associated with training data from different textures. The latter will be the case if, e.g., two textures are relatively similar to each other compared to several other textures in the training data. It is possible that these two textures can not be distinguished by the SOM. To tackle this issue, we employ a hierarchical approach utilizing the labels of the training data as some means of supervision. The training vectors associated with a contaminated node are used to train a new, smaller SOM as illustrated in Figure 2.11. A node's corresponding training vectors are used to train a new SOM if the proportion of the most occurring class of training vectors is below a certain threshold (τ). This process is iteratively repeated until a certain stopping criterion is reached or no progress in the classification can be obtained.

Suppose that a node N is related to a set V of training vectors of k texture classes c_i, $i = 0..k-1$:

$$N \leftarrow V = \left\{ v_{c_0,1} \,..\, v_{c_0,m_0},\, ..\, v_{c_j,1} \,..\, v_{c_{k-1},m_{k-1}} \right\},$$

where m_i denotes the number of training vectors of texture c_i related to node N. The stopping criterion is defined by the threshold $0 < \tau \leq 1$, so that:

$$\frac{\max_{i=0..k-1}\{m_i\}}{\sum_{i=0}^{k-1} m_i} \geq \tau. \tag{2.14}$$

If (2.14) is not satisfied, V is used to train a new SOM. This process is then repeated for the nodes of the resulting SOM. In other words, the iteration is

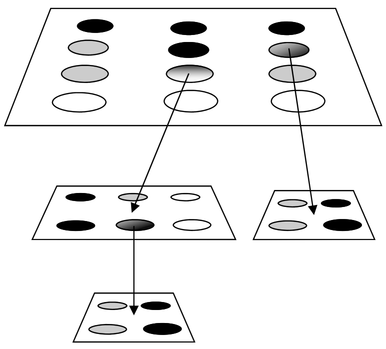


Figure 2.11: An illustration of a hierarchical SOM of 3 layers – On the highest layer, the 4×3 SOM has two contaminated nodes (gradient filled). The training vectors of these nodes are then used to train a smaller SOM. On the second layer, the 3×2 SOM also contains a contaminated node which training vectors are retrained by a 2×2 SOM on layer three.

stopped if the proportion of vectors related to a certain class is higher than a threshold, or no improvement in the classification can be obtained. If no improvement can be obtained and (2.14) is not satisfied, then the label that most frequently occurs in the node and the closest node, is chosen.

The parameter τ is set to 0.95 and the dimensions of the SOM for K texture classes are chosen as follows: 4×4 for $K = 2$ or 3; 8×8 for $K = 4$; 10×10 for $K = 5$; 15×15 for $K = 6$ and 20×20 for $K \geq 7$.

The performance and robustness of the SOM-based classification with the proposed texture features is analyzed using two different data sets: textures from the Brodatz and the Vistex album [107]. In contrast to the Brodatz textures, the Vistex textures are in 24-bit RGB colors but only the luminance is used to compute the texture features. Therefore, the Vistex textures are converted to 8-bit gray-scale images using the standard formula $L = 0.299R+0.587G+0.114B$. From the 512×512 pixel images, we crop an area of 512×452 pixels and use it as training data. The remaining 512×60 pixels are then used as test data for classification. The texture features are extracted from patches of 4×4 pixels as a compromise between the computation time and detail. This results in a total of 1920 test features per texture. The classification rate here is expressed as the total number of pixels that are correctly classified divided by total number of pixels.

We again use the SOM Toolbox [104] to create the SOMs, and the Euclidean distance is employed as distance metric. Next to the proposed texture features, we also tackle these classification problems using GMRF, multi-scale LBP, fuzzy LBP and Gaussian smoothed Gabor features (see equation 2.8). The implementation of the GMRF features is obtained from the MeasTex site [106] and the features are computed using the standard symmetric masks. The GMRF features from order 1 to 7 (GMRF$_{1-7}$, a 73-dimensional feature vector) and also GMRF features obtained from 13th and 14th order models (GMRF$_{13+14}$, an 80-dimensional feature vector), are considered. Further, the multi-scale LBP, i.e., LBP^{riu2} with parameters ($R_1 = 1$ pixel and $P_1 = 8$ pixels, $R_2 = 2.4$ pixels and $P_2 = 16$ pixels), are rotation invariant and uniform and the 2-dimensional co-occurrence LBP histograms are classified using a non-parametric L-statistic [63, 105]. The fuzzy LBP (FLBP) are calculated in a 3×3 neighborhood ($R = 1$ pixel, $P = 8$ pixels) with fuzzyfication parameter $T = 75$ as proposed in [108]. The resulting histograms are also classified using the L-statistic.

In the following section, different texture classification experiments will be discussed.

2.7.1 Classification Experiment

In this experiment, we test the classification performance of the different texture features separately on the two data sets. At first, 10 Brodatz images are used for classification: D6, D9, D12, D15, D19, D38, D68, D84, D94, and D104. As can be seen in Figure 2.12, the proposed features give rise to excellent classification results, e.g., a classification rate of 97.8% for $K = 10$ textures. The multi-scale LBP show similar results (97.7% for $K = 10$), they are well-suited for classification as well. In contrast, the FLBP obtain the lowest classification rate due to their small spatial support area (39.6% for $K = 10$). The spatially smoothed Gabor responses also obtain good classification results (91.9% for $K = 10$) and they are clearly better than the GMRF features for classifying the Brodatz textures. The GMRF$_{13+14}$ heavily underperform (55.9% for $K = 10$) whereas the GMRF$_{1-7}$ show more interesting classification results. However, as the number of textures increases, the classification performance of GMRF$_{1-7}$ drops to 81.5% for $K = 10$.

The same test is repeated for 10 textures of the Vistex album: Grass.0001, Bark.0009, Brick.0000, Metal.0002, Fabric.0005, Fabric.0015, Food.0005, Leaves.0008, Sand.0001, and Water.0005. As can be seen in Figure 2.13, the same behavior can be observed by classifying the Vistex textures. The proposed features and multi-scale co-occurrence LBP histograms have clearly the best performance (both 96.9% for $K = 10$). For the 10 texture classification problem, the smoothed Gabor filter responses also achieve high results (91.5%), whereas the GMRF_{1-7} (83.6%), the GMRF_{13+14} (51.6%), and the FLBP (39.1%) obviously underperform.

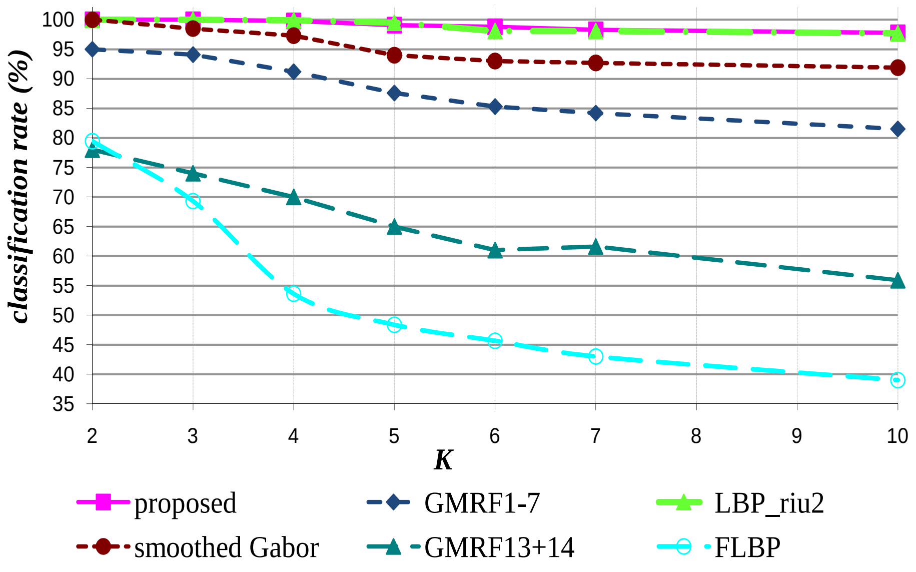


Figure 2.12: Classification rate of Brodatz textures in function of the number of textures *K*.

2.7.2 Noise Robustness

Although a plethora of texture analysis methods have been proposed in the literature, there are still some open issues where many applications struggle with. Noise is a frequent problem in texture characterization and causes difficulties for interpretation. Furthermore, different methods used to eliminate the noise, e.g., adaptive and non-adaptive filtering, eliminate some actual image data as well which results in loss of texture information.

In the literature, a few approaches exist to deal with noise. For example, Iakovidis et al. describe a fuzzy extension to the local binary pattern (LBP) operator to deal with noise in images [108]. An important drawback of their approach is that a parameter to control the degree of fuzziness has to be specified and the optimal value for this parameter is highly content-dependent. They also assume that the training and test images contain an equal amount of noise (in terms of the signal-to-noise ratio) . Furthermore, the small spatial support area

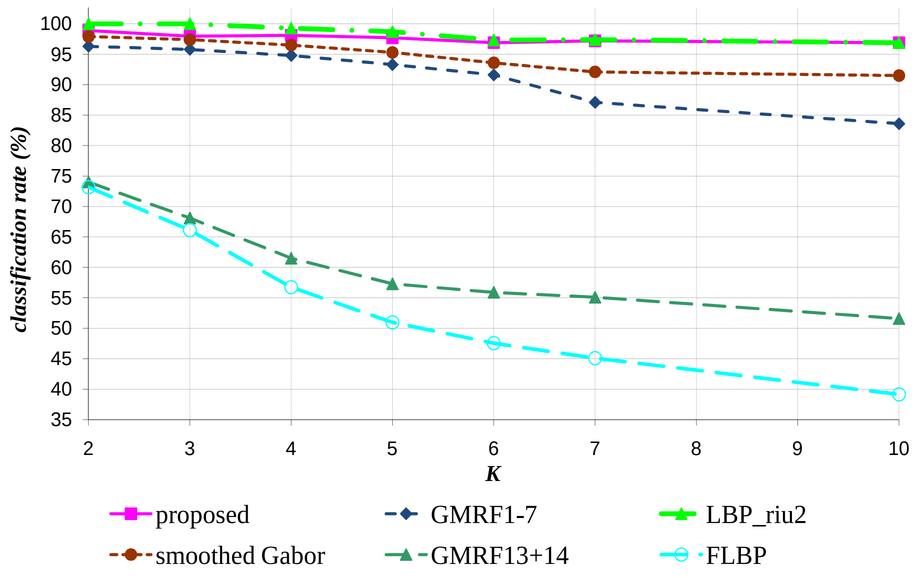


Figure 2.13: Classification rate of Vistex textures in function of the number of textures K.

(3×3 pixels) of the proposed operator cannot capture large-scale structures that may be prominent features of some textures. The use of larger areas results in huge histograms, and consequently, in high memory consumption and longer classification times. Murino et al. apply high-order statistics (which are insensitive to noise) to create a feature vector [109]. However, a selection algorithm is needed to reduce the high-dimensional data for classification. Fountain and Tan use a multi-channel Gabor filter bank for classifying textures from the Brodatz album [110]. By processing an image using multiple resolution techniques, filter banks have the ability to decompose an image into relevant texture features that can be used to classify the textures accordingly. They concluded that their method was relatively robust to Gaussian noise by using a sufficient number of features.

In order to test the robustness against noise for texture classification, we perform tests with uniform, speckle and Gaussian noise.

Uniform Noise

In this experiment, we add uniform noise to the same test images as in the previous experiments so that the image quality degraded with a PSNR of -20

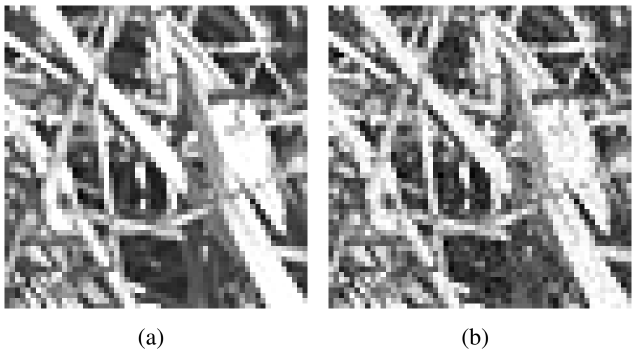


Figure 2.14: A detail of 60×60 pixels of a test image from the Vistex collection (Grass.0001) (a) original, (b) with uniform noise.

dB, as shown in Figure 2.14. Next, these noisy test images are classified using the network that is trained with the noise free training data. As can be seen in the graphs plotted in Figure 2.15 and Figure 2.16, we notice that the accuracy of the LBP and the GMRF features is drastically affected. The classification rate of the multi-scale LBP features drops to almost 'ad random' values (9.0% for $K = 10$). The FLBP obtain clearly better results (22.3% for $K = 10$) than the multi-scale LBP but they still underperform when taking into account more textures. In contrast to the classification results published by Iakovidis et al., the FLBP obtain remarkably lower classification rates in our tests. The reason of this might be that the training data in our test doesn't contain any noise while the training data in [108] also contain noise. The FLBP are clearly not suited for this classification problem. Both GMRF feature vectors rapidly become unreliable as the number of texture classes increases (GMRF$_{1-7}$: 25.3% and GMRF$_{13+14}$: 11.0% for $K = 10$). On the other hand, the filter bank approaches are more robust. The proposed texture features only lose about 5% accuracy compared to the noise free texture classification experiment of Section 2.7.1. The latter observation can be explained by the fact that the added noise did not greatly affect the periodic components of the textures, while for LBP and GMRF the random changes of the pixel values in the test data have a great impact on their distinguishing capabilities.

Speckle Noise

In the previous experiment, we have kept the amount of (uniform) noise constant and varied the number of textures. To know the classification rate be-

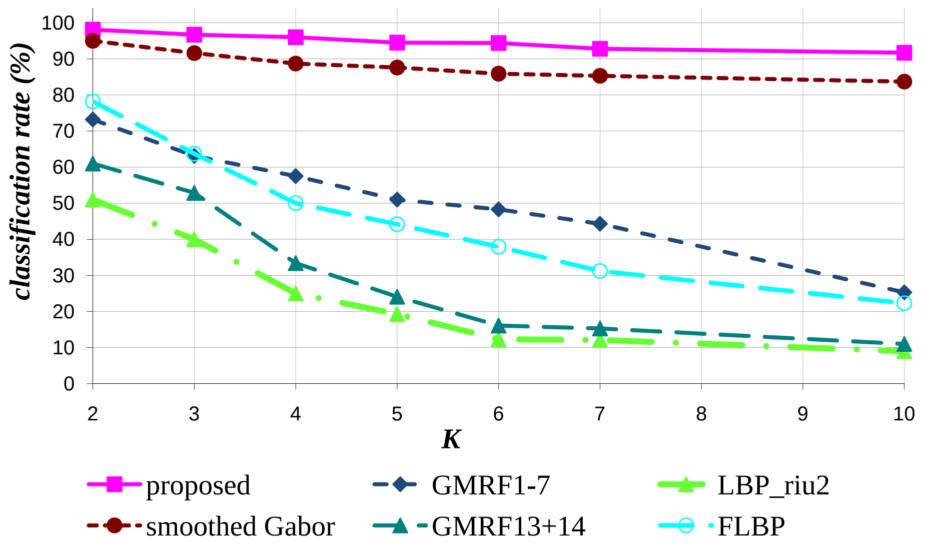


Figure 2.15: Classification rate of Brodatz test images with uniform noise as a function of the number of textures K.

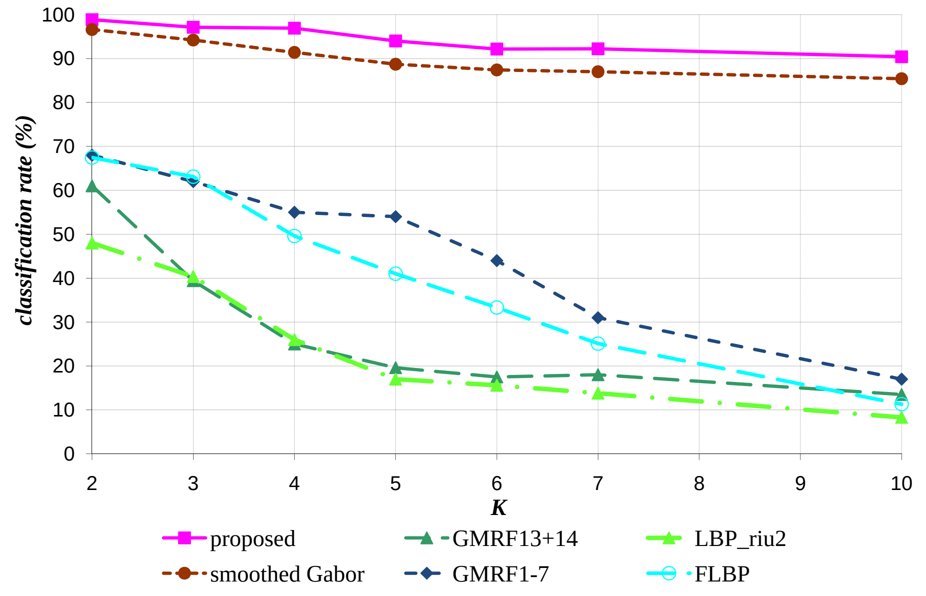


Figure 2.16: Classification rate of Vistex test images with uniform noise as a function of the number of textures K.

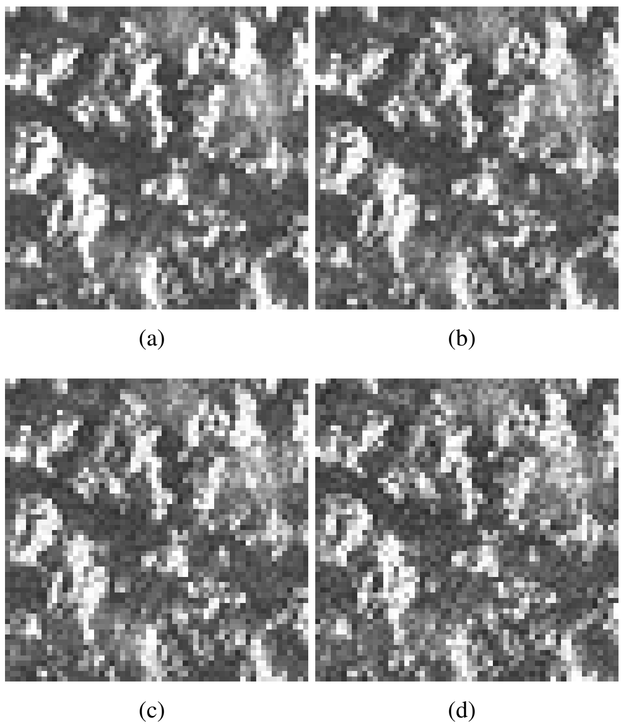


Figure 2.17: (a) Detail of the Vistex Metal.0002 texture, with zero mean speckle noise and variance (b) 0.01, (c) 0.02, and (d) 0.04.

haviour as a function of the amount of noise, we keep the number of textures constant at $K = 10$. In this experiment, we add zero mean speckle noise with different variances (0.0025, 0.005, 0.01, 0.02, 0.03, and 0.04) to the training data. Speckle noise is a random, granular noise and is an inherent characteristic of, e.g., ultrasound imaging. Furthermore, the observed speckle pattern does not correspond to the underlying structure of the texture. The noisy textures are then classified using the neural network that is trained with the noise free texture data.

Although hardly visible at first sight for the human eye (see Figure 2.17), the amount of speckle noise has actually a strong influence on the classification rates. As can be seen in Figure 2.18, the speckle noise affects the classification rates of all texture features, but the proposed texture features clearly obtain the best classification results and perform at least 25% better than the multi-scale LBP for the highest variance of the noise. For both the Brodatz and Vistex textures, the performance of the multi-scale LBP and the GMRF features drops as the variance of the noise increases. The classification rate of the FLBP only slightly decreases, and generally obtains a low performance. This not only due to the small spatial support of FLBP, but also due to the fact that the classifier is trained with noise free data, while in the experiments of Iakovidis et al. [108] the training data also contain noise.

Gaussian Noise

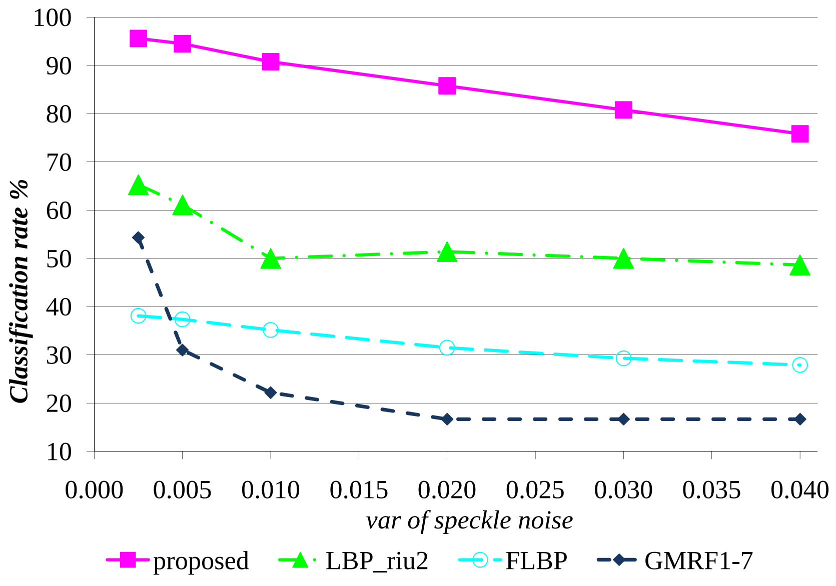
Similarly to the previous experiment, zero mean Gaussian noise of different variances (0.0025, 0.005, 0.01, 0.02, 0.03, and 0.04) is added to the test textures. As exemplified in Figure 2.19, the visual quality of the textures is highly affected.

Figure 2.20(a) and Figure 2.20(b) plot the classification rate of the Brodatz and Vistex texture samples with the Gaussian noise, respectively. As can be seen, the enhanced grating cell features with the Gaussian smoothed Gabor responses obtain the best classification rate while the other methods struggle with the noise. The performance of the FLBP only slightly decreases as the variance of the Gaussian noise increases, but obtains generally a low classification rate. Further, the classification rate of the GMRF and the multi-scale LBP steadily drops with an increasing variance of Gaussian noise.

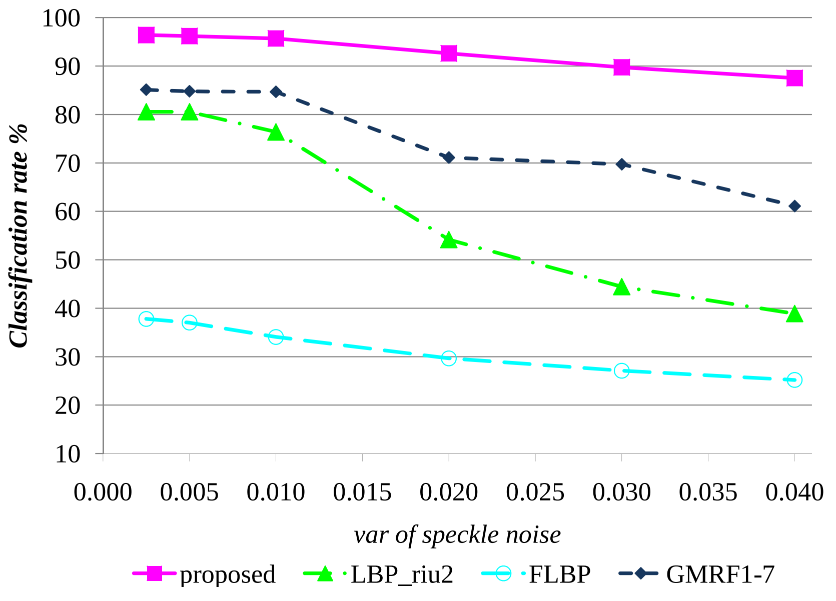
2.7.3 Image Compression

To cope with the high data volume of digital images and video, compression is a popular technique to reduce the size and to optimize the storage. However, the presence of artifacts caused by compression algorithms is an issue for texture analysis. Aschkenasy et al. have investigated the effect of compression on texture analysis of echocardiographic images [111]. They recommend the use of uncompressed or lossless compressed digital images in studies involving texture analysis since lossy compression affects texture parameters.

To test the robustness of the presented texture features for classification against image compression artifacts, we apply compression using a Joint Photographic Expert Group (JPEG) compression algorithm of the Independent JPEG Group (IJG) on the color Vistex textures so that blocking artifacts appear in the test images [112]. There is no direct measure of the degree of image distortion introduced by JPEG compression. We use the so-called ‘quality levels’ (Q) proposed by IJG, which range from 0 (lowest quality) to 100 (highest quality). Remark that the quality level $Q = 100$ is rather a mathematical limit than



(a)



(b)

Figure 2.18: Classification rate of (a) Brodatz and (b) Vistex textures in function of the variance of zero mean speckle noise.

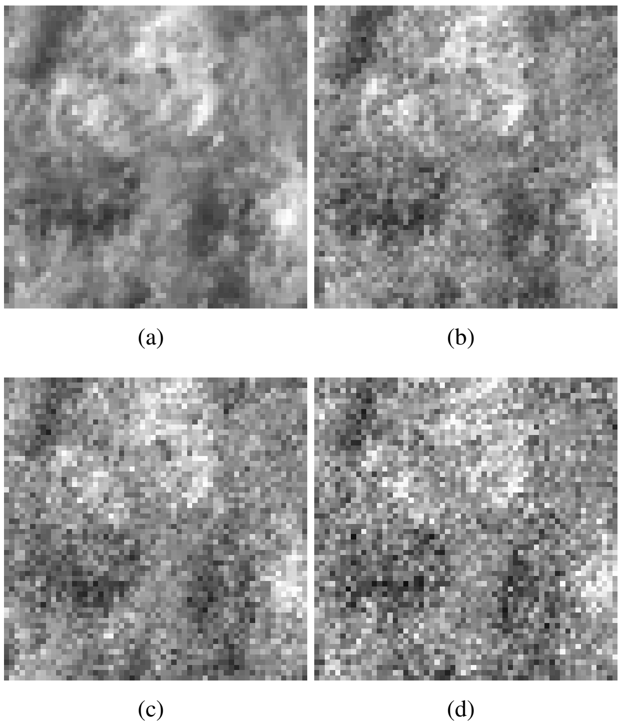
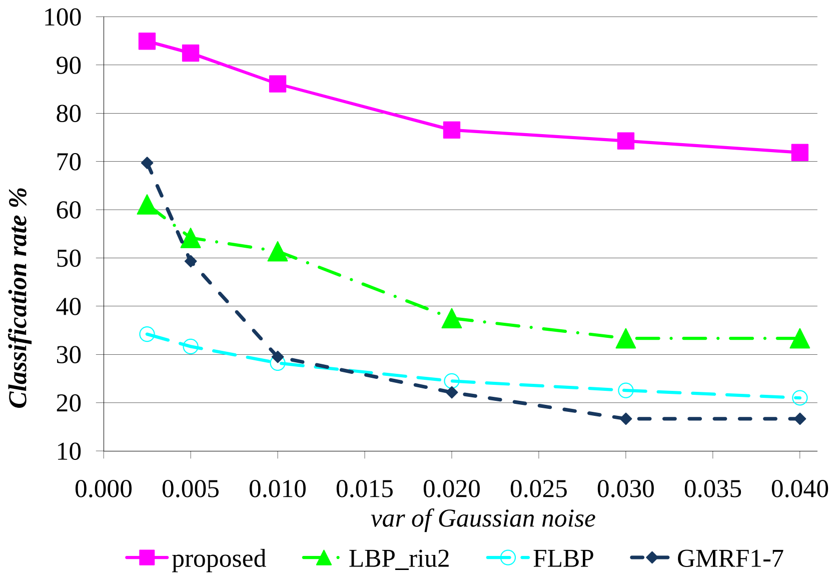


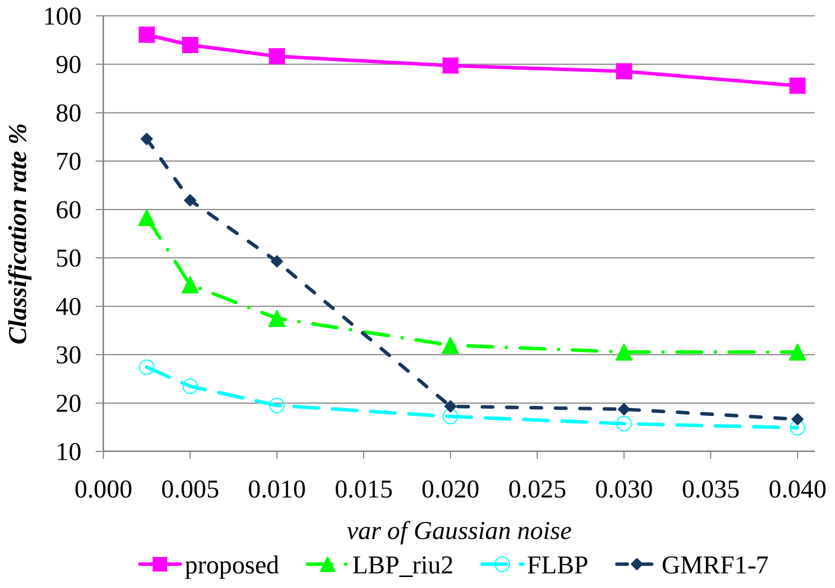
Figure 2.19: (a) Detail of the Vistex Fabric.0015 texture, with zero mean Gaussian noise and variance (b) 0.01, (c) 0.02, and (d) 0.04.

a useful setting. Therefore, the IJG recommends never to go above quality level 95. A quality level 100 will produce a file two or three times as large as $Q = 95$, but of hardly any better quality. The IJG recommends a quality setting of $Q = 75$ for good-quality, full-color source images, without expecting to see defects in a typical image. The lower the quality level, the more compression artifacts that will arise, while the high spatial-frequencies of the texture are more affected as exemplified in Figure 2.21. It is also important to remark that JPEG quality scales are not standardized across JPEG-creation programs: other JPEG implementations use completely different quality scales. Since little or no artifacts appear in gray-scale images using a JPEG compression algorithm (JPEG compresses hue data more heavily than brightness data), the Brodatz pictures are not used in this experiment.

In the first classification experiment, the 10 Vistex test textures are compressed at IJG quality level $Q = 15$, texture features are calculated and are then classified with a classifier that has been trained with the training data which are



(a)



(b)

Figure 2.20: Classification rate of (a) Brodatz and (b) Vistex textures in function of the variance of zero mean Gaussian noise.

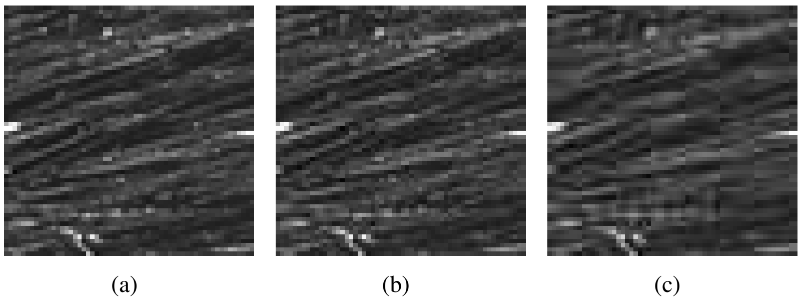


Figure 2.21: (a) Detail of the Vistex Fabric.0005 texture, compressed at quality level (b) $Q = 75$ and (c) $Q = 15$.

not compressed. The test results, which are plotted in Figure 2.22, indicate that the presented texture features obtain the best classification results compared to the other texture features for all values of K (e.g., 90% for $K = 10$). Also the smoothed Gabor filter responses show some robustness against JPEG compression while the other texture features clearly do not. Since the JPEG compression mainly affects the high frequency details in the images, the filter-bank-based methods still achieve satisfactory classification results because the image is decomposed at different scales. On the other hand, the classification rate of the multi-scale LBP rapidly drops as the number of textures increases. The GMRF$_{1-7}$ features perform slightly better than GMRF$_{13+14}$, but the results for both GMRF vectors are unsatisfactory. The FLBP obtain, again, the lowest classification rates. It is clear that the multi-resolution approach of our filter bank can deal better with this loss of information than LBP and GMRF since the latter methods are more sensible to local changes of pixel values.

In the second experiment, we test the impact of the JPEG quality level Q on the classification rate. Therefore, the number of texture classes is kept constant (i.e., $K = 10$). In this experiment, we omitted the GMRF$_{13+14}$ because of their low accuracy in the previous experiments. Also the G$_s$ are omitted since they make part of the proposed texture features which performance is better. As plotted in Figure 2.23, it is obvious that the proposed features are hardly affected by the JPEG compression. The multi-scale LBP can keep up with the proposed features, but for quality levels lower than 75, the classification rate steadily drops and attains for quality level 15 a lower classification rate than FLBP or GMRF. Also, the already low classification rate of the FLBP drops for quality levels lower than 75. The classification rate of the GMRF also starts decreasing from quality level 75.

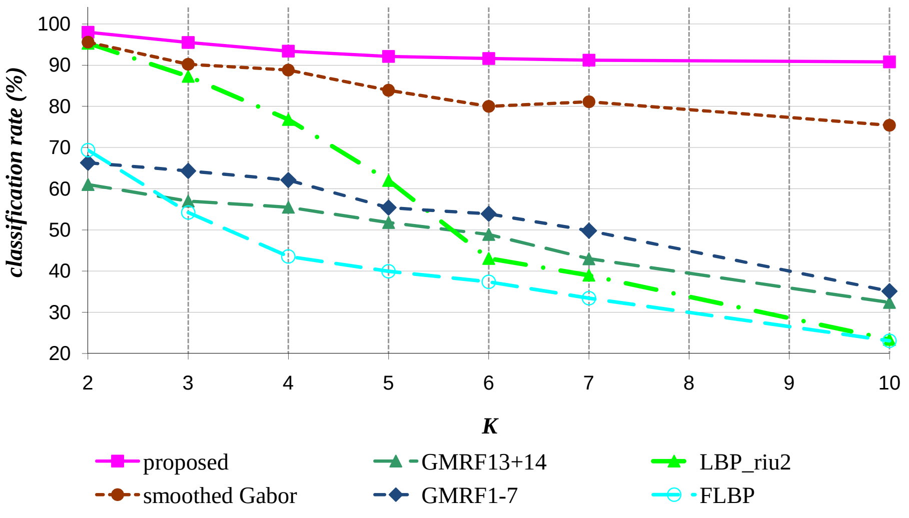


Figure 2.22: Classification rate of JPEG compressed Vistex test images in function of the number textures K. The test images are compressed with an IJG compressor at quality level 15.

2.8 Material Classification

The problem of classifying materials from their imaged appearance, without imposing any constraints on, or requiring any a priori knowledge of the viewing or illumination conditions under which these images were obtained, is an extremely challenging task. This is because small variations in the illumination or camera position can have a huge impact on the acquired image. Classifying materials from a single image obtained under unknown viewpoint and illumination conditions is similar to texture classification since they both involve the classification based on the visual appearance of a surface. In this section, we examine the use of texture information for classifying materials. Therefore, we apply the texture classification approach which is explained in previous section (2.7) on textures from the Vistex library in Section 2.8.1 and on a self-created library of outdoor textures in Section 2.8.2.

2.8.1 Material Classification on Vistex Textures

The Vistex library contains textures which are acquired from photographs of materials, such as bark, bricks, leaves, and water. In a first material classi-

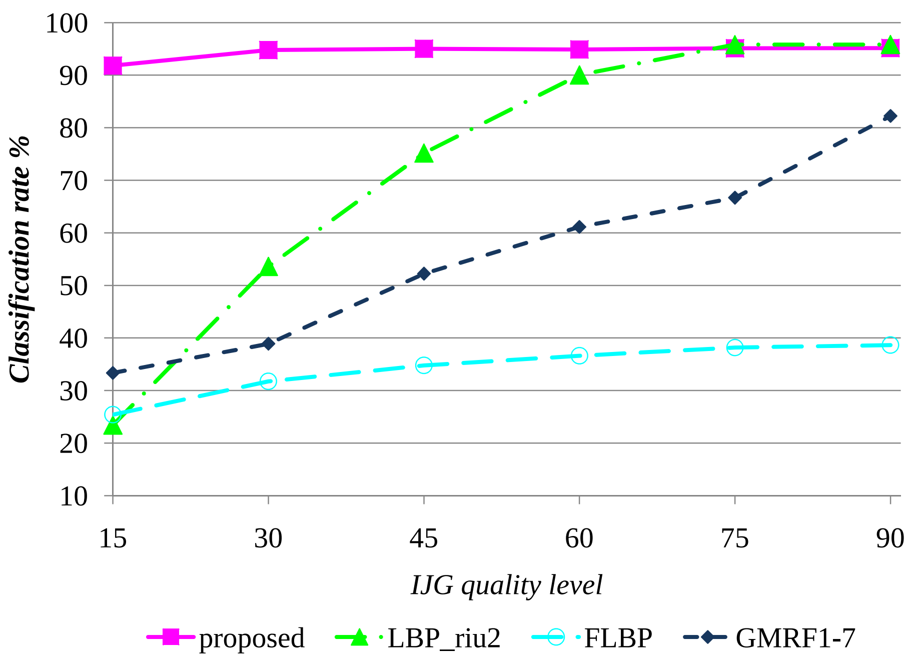
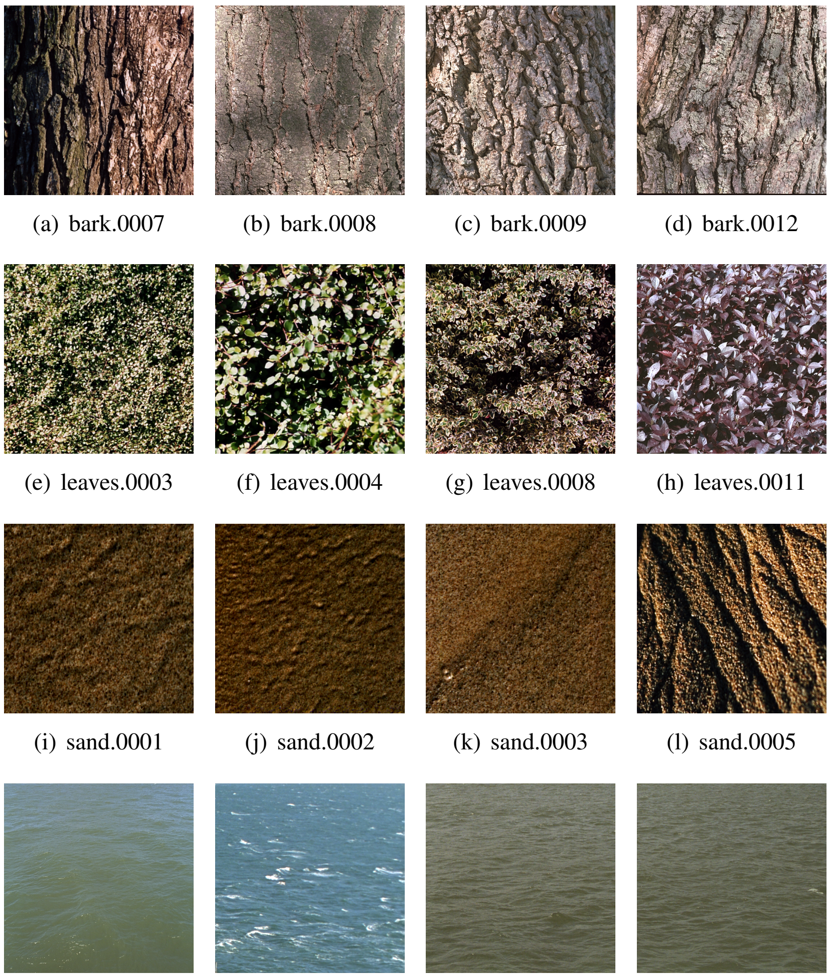


Figure 2.23: Classification rate of JPEG compressed Vistex textures in function of the IJG quality levels.

fication experiment, we group different Vistex images in one class based on the corresponding material or object from which the photograph is taken. In this way, textures that describe the same semantics and therefore are similar to humans, are grouped together. We have considered 4 texture classes: `Water`, `Bark`, `Sand`, and `Leaves` and each texture class contains 4 images as depicted in Figure 2.24. From each texture class, we leave out one image that we use as test image for classification, the other three images are then used to train the classifier. This process is repeated for every image (cross-validation).

As can be seen in Table 2.7, the proposed texture features ($G_s + Grat_{enh}$) achieve the best classification results. However, the texture class `Leaves` obtains a considerably lower classification rate than the other classes. The same finding applies for the smoothed Gabor filter response (G_s) and the multi-scale LBP features. The latter is caused by the fact that the intra-variation of the `Leaves` class is very high. As can be seen in Figure 2.24 (e)–(h), the scale of the texture varies a lot, and consequently, our classifier has some difficulties with these deviations. Our experiments further indicate that GMRF features



(a) bark.0007 (b) bark.0008 (c) bark.0009 (d) bark.0012

(e) leaves.0003 (f) leaves.0004 (g) leaves.0008 (h) leaves.0011

(i) sand.0001 (j) sand.0002 (k) sand.0003 (l) sand.0005

(m) water.0001 (n) water.0002 (o) water.0003 (p) water.0006

Figure 2.24: Textures classes: `Bark` (a)-(d), `Leaves` (e)-(h), `Sand` (i)-(l), `Water` (m)-(p).

are clearly not designed for this kind of task. The high classification rate of the `Leaves` class using GMRF could indicate that GMRF can deal better with the near-stochastic[5] property of this texture. However, the high rate is deceiving. We observe that the `Leaves` class accounts for a high number of the misclassifications using GMRF since a large fraction (i.e., 51%) of the fea-

[5]In this context, a stochastic texture has a more random, noisy appearance, such as randomly scattered dots.

tures belonging to each texture class is misclassified as `Leaves`. The GMRF features are just not sufficiently distinctive for material classification. Also the FLBP attain poor classification results due to the small spatial support of the operator.

	$G_s + \text{Grat}_{enh}$	GMRF_{1-7}	GMRF_{13+14}
`Bark`	99.3	14.8	1.8
`Leaves`	74.9	93.0	90.7
`Sand`	100	55.3	24.4
`Water`	97.3	60.1	43.1
average	92.9	54.8	40.0
	LBP^{riu2}	FLBP	G_s
`Bark`	99.0	57.0	85.9
`Leaves`	72.9	20.1	64.0
`Sand`	100.0	27.8	99.1
`Water`	81.3	98.1	91.7
average	88.3	56.68	85.2

Table 2.7: Classification rate (%) of 4 texture classes obtained from the Vistex album.

2.8.2 Material Classification on Outdoor Textures

To test the material classification of outdoor textures, we create an album of 100 outdoor textures which are manually collected from the World Wide Web (WWW). Each collected texture belongs to one of these five classes: (i) `Branches`, (ii) `Bricks`, (iii) `Grass`, (iv) `Sky`, and (v) `Water`. Every class contains 20 samples. Figure 2.25 depicts some example textures from the self-created texture album. As in the previous material classification experiment, one image is used as test sample while the other ones are used to train the classifier. As depicted in Table 2.8, the $G_s + \text{Grat}_{enh}$ texture features achieve the best classification results over all materials. Also the smoothed Gabor responses (G_s) obtain very good results. However, the multi-scale LBP have some difficulties with the `Water` and `Grass` textures because of the high variation of the data. Our experiments further indicate that GMRF features are clearly not designed for this kind of task. The GMRF have major difficulties to discriminate textures from the class `Water` and `Sky`, and textures from the class `Grass` and `Branches`. Since the FLBP and the GMRF_{13+14} have achieved low classification rates in the previous experiments, they have been omitted in this one.

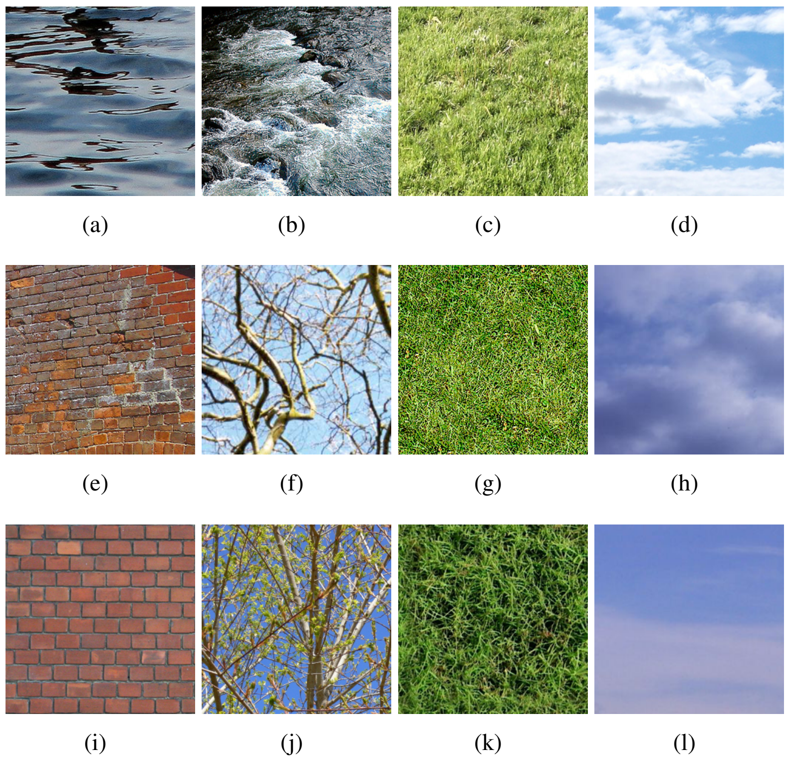


Figure 2.25: Examples of outdoor textures: water (a, b), branches (f, j), sky (d, h, l), bricks (e, i), and grass (c, g, k).

	$G_s + Grat_{enh}$	$GMRF_{1-7}$	LBP^{riu2}	G_s
Branches	96.8	81.6	72.2	87.6
Bricks	98.5	26.5	86.11	81.9
Grass	87.7	15.5	11.1	85.8
Sky	95.9	99.9	100	95.3
Water	85.9	2.3	16.6	85.7
average	93.0	45.2	57.2	87.3

Table 2.8: Classification rate (%) of outdoor textures.

2.9 Outdoor Scene Labeling

This section describes the interpretation of outdoor scenery images using texture information. We demonstrate how the presented HVS-based texture features can be used for labeling regions in images. Before our approach is described in Section 2.9.2, we first give an overview of related work in the next section.

2.9.1 Related Work

Early content-based image retrieval systems were based on the search for the best match to a user-provided query image or sketch [113–115]. Such systems decompose each image into a number of low-level visual features (e.g., color histograms, edge information) and the retrieval process is formulated as the search for the best match to the feature vector(s) extracted from a query image. However, it was quickly realized that the design of a fully functional retrieval system would require support for semantic queries [116]. The basic idea is to automatically associate semantic keywords with each image by building models of visual appearance of the semantic concepts of interest. However, the critical point in the advancement of content-based image retrieval is the semantic gap which makes describing high-level semantic concepts with low-level visual features a challenging task. The first efforts targeted the extraction of specific semantics under the framework of binary classification, such as indoor versus outdoor [117], and city versus landscape classification [118]. More recently, efforts have emerged to solve the problem in greater generality through the design of techniques capable of learning semantic vocabularies from annotated training image collections by applying (both unsupervised and semi-supervised) machine learning techniques, e.g. [119, 120]. Many methods have been proposed for region-based image retrieval. Zhu et al. [121] partition the image into equally sized blocks, index the regions using a codebook whose entries are obtained from features extracted from a block. Finally, images are indexed using this codebook. Image retrieval is then performed based on the indexed images. However, the equal-sized blocks ignore the boundary of image regions and consequently, these blocks cannot represent the objects correctly. Wang et al. use a codebook to segment an image based on the statistics of the regions' color and texture features [122]. At pixel level, color-texture classification is used to form the codebook. This codebook is in the next stage used to segment an image into regions. The context and content of these regions are defined at image level. The method of Li and Wang [123] uses 2-dimensional hidden Markov models to associate the image and a textual description. A major drawback of these approaches is that they cannot integrate the semantic descriptions into the image regions, and therefore they cannot support the high-level querying of images. For that reason, some approaches use an image partitioning as an intermediate step to extract the semantics of scenery images using low-level features. Depalov et al. [124] use a quantized color and texture segmentation algorithm to segment images depicting natural scenes. The features of the obtained regions are used as medium level descriptors to extract semantic labels at region level and later at scene level. Yuan et al. use spatial context constraints to label image regions [125]. Segmented image regions are first regularized into a 2-dimensional lattice layout to represent a graphical model for learning and inference. However, their learning is supervised and the parameters of the support vector machines and conditional random fields are estimated sequentially rather than simultaneously. In [126], a self-organizing map trained with local binary patterns is employed to classify outdoor scene images. As a means of supervision, the user selects the map nodes with similar appearance and the corresponding samples are retrained using a smaller map in order to reveal if some classes are mixed up in the same node. A major weakness of these retrieval systems is their lack of domain knowledge. Consequently, many systems are error prone when it comes to detection of high-level concepts. Athanasiadis et al. associate a region with a fuzzy set of candidate concepts stored in an ontological knowledge base [127]. A merging process is performed based on new similarity measures and merging criteria that are defined at the semantic level with the use of fuzzy set operations. Schober et al. apply domain knowledge for the interpretation of landscape images [128]. They relate the extracted low-level features with concepts and then generate rules which define the coherences between the concepts. An ontology defines the spatial relations between the concepts to remove the incorrect assignments. A detailed overview of content-based image retrieval techniques which include semantics is given by Liu et al. [129]. They divide these methods into five categories:

(i) employing ontologies to define high-level concepts,

(ii) applying machine learning on low-level features,

(iii) using relevance feedback,

(iv) generating semantic templates to assist high-level information retrieval,

(v) using both visual content and surrounding text.

2.9.2 Methodology

Despite the efforts, humans outperform these machine vision systems in many aspects. Humans are very good at getting the conceptual category and layout of a scene within a single fixation. However, it is still unknown how visual information is fully processed by the human brain. Available information about the processing of the HVS indicates that the first stage of a system that is able to automatically interpret visual information, should involve the use of perceptually based features, i.e., features that are based on the HVS.

According to the above five categories by Liu et al, the approach presented in this dissertation uses techniques from the first and second category: we apply machine learning on perceptual texture features, and in the final stage, we ingest domain knowledge. Our method to assign a label to regions of outdoor scenery images consists of 3 successive steps: (i) segmentation, (ii) classification, and (iii) the application of domain knowledge. Thus, our methodology employs two strategies:

(i) a bottom-up strategy to compute semantically relevant information from the low-level image data,

(ii) a top-down strategy that ingests domain knowledge to increase the accuracy of the obtained interpretation.

Image Segmentation

Instead of directly assigning a label to each pixel, we first apply an intermediate segmentation step. Based on the perceptual texture (i.e., enhanced grating cell with Gaussian smoothed Gabor responses) obtained from patches of 4×4 pixels, the image is segmented into similar regions with no supervision as explained in Section 2.6.

Labeling of Image Regions

In the second stage, the same texture features which are used to obtain the image regions, are used for material identification in order to assign a label (see Section 2.7). For that purpose, the hierarchical SOM-based classifier is trained with features obtained from different types of texture classes. The label of an image region is then easily obtained by selecting the label that occurs most frequently. However, the latter implies that only regions consisting of similar textures (i.e., the same texture classes) as those in the training set, can be correctly identified by the classifier. In order to obtain a successful identification of an image region, there are two requirements: (i) a representative training set and (ii) discriminative texture features. However, the creation of a representative training set is a very difficult issue, if not practically impossible. For the labeling of outdoor scenery image regions, we use the texture album presented in Section 2.8.2 which consists of five texture classes, i.e., (i) `Branches`, (ii) `Bricks`, (iii) `Grass`, (iv) `Sky`, and (v) `Water`.

Exploitation of Domain Knowledge

Due to the difficulties described above, any bottom-up approach is error prone. In order to cope with errors and to obtain a more plausible interpretation of the depicted scene, we ingest domain knowledge. For that reason, an ontology is created. An ontology is high level domain knowledge that describes the conditions and restrictions of the depicted concepts (i.e., the concepts that are present in the training set). In this way, image regions can be merged or misclassifications and illogical compositions can be removed or altered. However, one should pay special attention to the knowledge modeling phase to avoid false rules or rules that are not universally applicable within the given context.

The created ontology consists of a few simple, but effective rules. Given the concepts of our training set, the following rules are iteratively applied:

(i) neighboring regions with the same label, are merged,

(ii) no region can exist in `Sky`,

(iii) no region of `Sky` can exist in `water`,

(iv) no `Water` can be above `Sky`,

(v) a region should be at least 8 pixels wide and high,

(vi) regions that not obey to the above rules, are relabeled (the new label is obtained by the ith BMU of the region, where i denotes the iteration number).

2.9.3 Experiments

The image labeling experiments are conducted on ten scenery images of various sizes, e.g., see Figure 2.26(a), Figure 2.26(e) and Figure 2.27(a). These

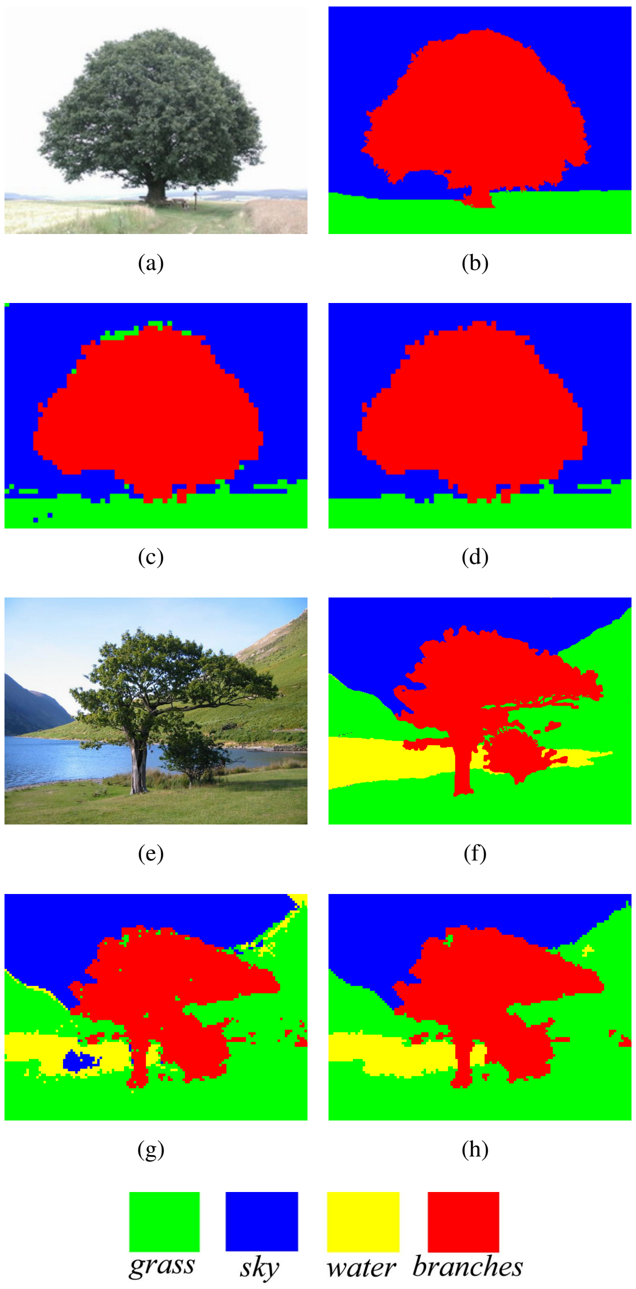


Figure 2.26: (a, e) Scenery image, (b, f) ground truth, (c, g) labeled regions, and (d, h) obtained regions after ingestion of domain knowledge.

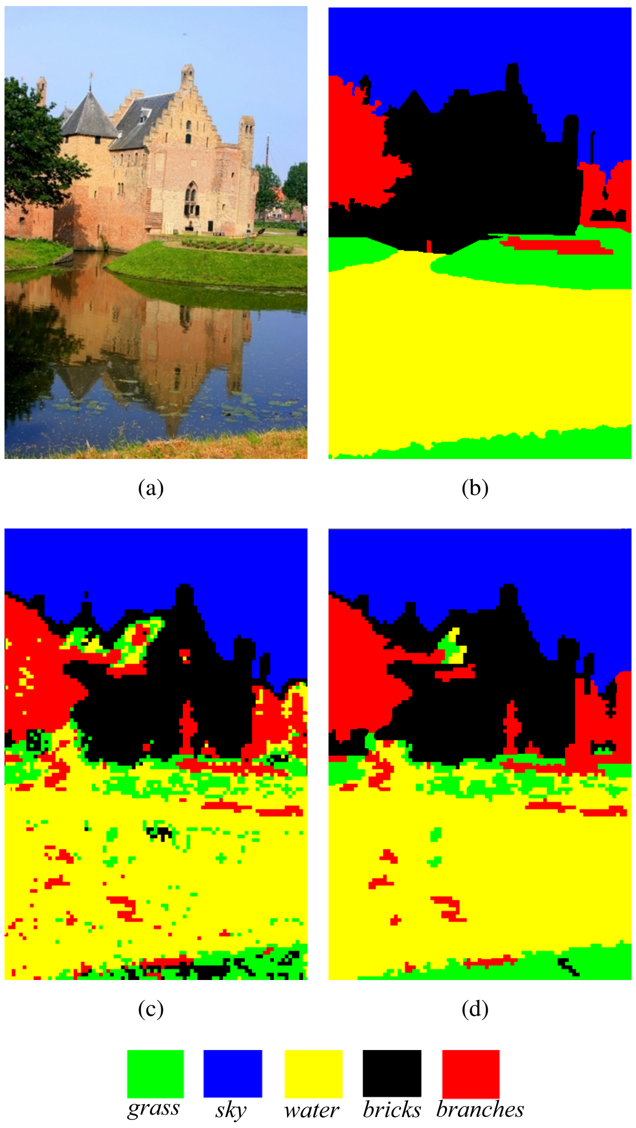


Figure 2.27: (a) Scenery image (a), (b) ground truth, (c) labeled regions, and (d) obtained regions after ingestion of domain knowledge.

images contain no other texture classes than those in our training set. The ground truth (see Figure 2.26(b), Figure 2.26(f) and Figure 2.27(b)) is created manually and therefore it should be interpreted as an approximation rather than a certainty. Texture features (i.e., enhanced grating cell responses with Gaussian smoothed Gabor responses) are computed from patches of 4×4 pixels and are fed to a SOM of 10×10 nodes for segmentation.

After the segmentation process, our trained classifier is used to label the computed image regions. As can be seen in Figure 2.26(c), Figure 2.26(g) and Figure 2.27(c), the result of this labeling process contains different errors. At first, some isolated patches are misclassified. These errors can be removed by applying constraints on the size of the regions. However, some other errors remain, e.g., due to reflection, such as the *Sky*-blob in the lake of Figure 2.26(c). Such type of errors can only be removed by incorporating domain knowledge. Other errors emerge from the fact that the scaling of certain textures alters due to changes in the perspective, e.g., at the edges of the mountains in Figure 2.27(c). Generally, this problem is much harder to tackle.

In the last step, the ontology is applied on the intermediate results. As can be seen in Figure 2.26(d), 2.26(h) and 2.27(d), small misclassified regions are filtered away, and the blob of *Sky* in the lake is removed. After the semi-supervised classification step, 82.7% of the pixels have a correct label. However, employing the domain knowledge enhanced the region labeling with 9.2% up to 91.9%.

Despite these good results, some improvements can still be made. At first, the segmentation step should be enhanced. In our experiments, we have noticed that some region boundaries are falsely detected. The latter could be tackled by also taking edge information into account or, by using color information as well. Indeed, since color is the primary visual stimulus, we expect that introducing color information, next to texture information, could also increase the classification rate. In the HVS, bar and grating cells seem to play an important role in boundary detection [77, 81]. In contrast to grating cells, bar cells respond only to an isolated edge or line but do not respond to any texture edge. Hence, it is possible to distinguish between an edge belonging to a textured region and a non-textured region. This way, the problems that arise due to the perspective can be tackled. Since the scale of the texture is not constant due to the perspective, a segmentation solely relying on texture will distinguish texture samples that are "close" and samples that are "far" away from the position from which the photo was taken. However, if no clear edge can be detected between these regions, they should be considered as one region.

2.10 Conclusions and Future Work

2.10.1 Conclusions

In this chapter, we have presented the use of HVS-inspired features for texture analysis. These features consist of enhanced grating cell responses combined with Gaussian smoothed Gabor responses, and correspond to outputs of cells found in the primary and secondary visual cortex of humans. Enhanced grating cell responses give a stronger response to the salient texture-specific orientations and periodicities than the original grating cell features. Although the use of (enhanced) grating cell responses is not sufficient to obtain a successful texture characterization, their combination with Gaussian smoothed Gabor responses produces a texture feature that has more distinguishing characteristics than existing texture analysis methods. Furthermore, the behaviour of grating cells is an added value for a texture operator since non-texture features (such as isolated pixels or edges) can be distinguished from texture patterns containing some specific orientation.

Using SOMs for the unsupervised segmentation of textured images, we have shown that these features clearly obtain the best segmentation results, up to a recall of 85.7% (without using pixel adjacency information) for the segmentation of nine textured images containing Brodatz textures.

For the classification of texture information, we have employed hierarchical SOMs. Hierarchical SOMs are obtained by retraining nodes which contain texture features from different classes using smaller SOMs. Experiments conducted on different texture libraries indicate that the presented texture features have excellent discriminating capabilities and can compete with state-of-the-art texture analysis methods like multi-scale LBP. Classification rates up to 97.8% and 96.9% have been achieved for the classification of 10 Brodatz textures and 10 Vistex textures, respectively.

For most practical applications the presence of noise in images is a problem in texture characterization and causes difficulties for interpretation. In our experiments, we have shown that the presented approach is highly robust to various levels of uniform, speckle and Gaussian noise. In contrast, the classification rate of well-established texture analysis methods, such as GMRF or LBP, steadily drops as the variance of the noise increases. The proposed texture features obtain classification rates that are, e.g., at least 25% and 35% better than other methods when speckle noise and Gaussian noise is introduced, respectively. Also the presence of compression artifacts in images causes problems for texture characterization. By applying JPEG image compression, we have shown that the presented texture features are highly robust and can deal better with the impact on the high spatial-frequencies of the textures than other texture analysis methods. For the highest tested compression rate (IJG quality level 15), the classification rate obtained by our proposed method is at least 50% higher than the classification rate that is obtained by state-of-the-art methods. The robustness against noise and compression artifacts of the presented texture features has the advantage that no prior knowledge about the obtained images is needed for applying texture analysis.

We also examined the use of texture features for material detection. The presented texture features also achieve the highest classification rates.

Finally, the presented texture features are used for the interpretation of textured outdoor scenery images by first applying segmentation, followed by material detection on the obtained image regions. The experiments point out that the use of perceptually based texture information is an added value for scene interpretation since these low-level features can be related to semantic concepts. To model cognition, we have ingested domain knowledge by applying a top-down approach. An ontology describing the conditions and restrictions of the depicted concepts, can tackle many errors which are related to a bottom-up approach.

2.10.2 Future Work

For the material detection, some issues due to the high variation of the visual appearance of materials have shown up: (i) the altering of the scale, and (ii) the high intra-class variation. Therefore, robustness to scale variations and the capability to generalize across different instances of the same materials, are needed. We are convinced that more sophisticated machine learning techniques and class-specific feature selection methods are necessary to tackle these problems.

To automatically obtain a more accurate scene description, texture information is not sufficient. Color information, which is the primary visual stimulus, will certainly be helpful. We demonstrated in the third segmentation experiment (Section 2.6.3) that the combined use of color and texture information achieves better segmentation results than the single use of color or texture information. However, the recognition by the HVS of visual scenes is yet not fully understood. Moreover, the visual cortex consists of many cell types from which it is believed that their functionality still has to be discovered. Additional models are already available for keypoints [130], dot patterns [131], disparity [132], contour completion [133], and non-classical receptive field inhibition [134]. These models can possibly be improved and must somehow be integrated into a complete cortical architecture.

Finally, a general issue related to texture analysis methods is the computation time of the feature extraction. To achieve real-time processing using Gabor filter banks, it has been shown that a hardware implementation might be inevitable, especially if the resources are limited [135]. However, with the continuously expanding digital photo and video collections, the need for efficient and (semi-)automatic content-based retrieval is even growing faster. Consequently, we believe that optimized (hardware) implementations of state-of-the-art techniques will be indispensable to achieve high retrieval results and to cope with the enormous amounts of data in limited time.

The work that was presented in this chapter is accepted for publication in the Visual Computer Journal:

(i) G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Noise and compression robust biological features for texture classification. *The Visual Computer*, 26(6-8):915–922, June 2010

and is submitted to the Journal of Image Vision and Computing:

(i) G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Semi-supervised classification of robust texture features inspired by the human visual system. *Visual Communication and Image Representation*, 2010. Under review

Furthermore, work in this domain has lead to the following book chapter:

(i) G. Martens, P. Lambert, and R. Van de Walle. *Self-Organizing Maps*, chapter Bridging the semantic gap using human vision system inspired features, pages 261–276. In-Tech, 2010

Additionally, work in this domain can be found in the following publications:

(i) G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Perceptual-based textures for scene labeling: a bottom-up and a top-down approach, 5 2010

(ii) G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Unsupervised texture segmentation and labeling using biologically inspired features. In D. Feng, D. Sikora, W.C. Siu, J. Zhang, J. Guan, L. Dugelay, Q. Wu, and Li W., editors, *Proceedings of the 2008 IEEE 10th workshop on Multimedia Signal Processing*, pages 159–164, Cairns, 10 2008. IEEE Signal Processing Society

(iii) G. Martens, C. Poppe, and R. Van de Walle. Enhanced grating cell features for unsupervised texture segmentation. In W. Ponweiser, editor, *Performance Evaluation for Computer Vision: 31ste AAPR/OAGM Workshop 2007*, pages 9–16, Wien, 5 2007. Osterreichische Computer Gesellschaft

Chapter 3

Musical Audio Mining

After silence that which comes nearest
to expressing the inexpressible is music.
– Aldous Huxley (1894 – 1963)

3.1 Introduction

"*What is music?*" Is it the sound we hear through our speakers when we turn on the radio or when we play a compact disc? Is music what musicians play and what singers sing? Or is music the sound that affects our emotions, what makes us happy, or makes us cry... "the language of emotions"? Sometimes, music makes us want to dance or feel sleepy. Some people say that music is art. However, everybody can recognize pieces of music as we hear them. Furthermore, we know there are different kinds of music and we know which kinds we like or do not. Thus, the question what music is, is a relatively simple one, but the answer is apparently not so trivial. It is a fact that music is present in many aspects of our lives. The question "*What is music for?*" might even be more difficult to answer. A simple answer is that music is enjoyable. We turn to music to uplift us even further in happy times or seek the comfort of music when melancholy strikes. Therefore, music has been used in many domains to intensify sensations, such as advertising and film production while it is also used in therapeutic sessions for the reduction of anxiety and stress, the relief of pain, and as an aid for positive change in mood and emotional states. Music is thus a rich and a powerful medium. What is certain about music is that it is related with the sense of hearing and hearing involves sound and the perception of it. But, it is hard to state which sound is music and which is not. Some people will describe certain sounds as “disturbing” or “noise” while the concatenation of such sounds may be perceived as music. The notion of music, or at least the appreciation of it, is thus highly subjective and culture-specific. However, there are structures or musical properties that can universally be recognized, such as *melody* and *rhythm*. Humans are capable to recognize and reproduce these temporal arrangements of sound. People often only remember the melody and/or the rhythm of a piece they’ve heard, rather than the name of the song and performer. By imitating these patterns, e.g., by singing, whistling, or humming, one would be able to retrieve information from a musical collection in a natural way. However, consulting large music collections and databases with audio information occurs still in a textual manner as already stated in Section 1.3. There is an urging need for techniques that are able to extract information directly from the musical content, rather than relying on human-inputted metadata. The domain that deals with the retrieval of information related to music, is called *Music Information Retrieval* (MIR).

The perception of (musical) sound involves many layers of processing in the auditory system. Therefore, appendix D elaborates on the processing of sound by the human auditory system and gives an overview of the most important – within the context of this dissertation – aspects of the auditory system as certain aspects will later be mentioned in the text again. The remainder of this chapter is organized as follows. In Section 3.2, we describe the context of MIR and its relation with this thesis. Furthermore, we will explain the notion of sound in a musical context and define the concept “melody” in this dissertation. In Section 3.3, we will deal with the extraction of melody lines from musical audio recordings. Unlike many melody extraction approaches, we will aim to explicitly distinguish individual musical notes, characterized by specific temporal boundaries. Furthermore, we will intend to obtain the melody information from both polyphonic and monophonic recordings and the described system does not impose any restrictions on the audio recordings concerning the instrumentation nor the number of instruments (simultaneously) playing. Next to the extraction of melodic information, we will also consider the extraction and classification of tonality information in Section 3.4. Like melody, tonality also entails pitch information, but it manifests over a relatively larger time interval than melody. We will present a novel method to obtain the key of a musical audio piece by using a classification-tree-based approach instead of the widely used metric-based approach. Finally, in Section 3.5, concluding remarks about the presented approaches will be formulated.

3.2 Music Information Retrieval

Music information retrieval (MIR) is an interdisciplinary research area which deals with a variety of tasks. Basic research includes many topics, such as representation of music, interaction, indexing, retrieval, content analysis, music recommendation, audio compression, metadata, ontology modeling, perception, psychology, and intellectual property rights. According to Futrelle and Downie, these research themes can be grouped according to the kind of music representation they employ [136]. This way, a distinction can be made between:

(i) *Metadata MIR* tackles the representation of metadata for tasks such as cataloguing, ontology building and reasoning.

(ii) *Symbolic MIR* requires a symbolic representation of music, such as MIDI, for topics like music analysis, melody matching, and score following.

(iii) *Audio MIR* deals with the retrieval of information from audio recordings in analog or digital format. Examples are genre classification, drum detection, melody detection, audio compression, instrument recognition, digitalization, noise removal, etc.

However, we can add a fourth category to the above list, namely *multi-modal MIR*, which uses different representations, such as in music recommendation where both metadata and symbolic or audio representations are used for analysis.

Using the above categorization, the research presented in this dissertation falls within the audio MIR category. In this thesis, research in the domain of tonality and melody detection from musical audio are presented.

As far as musical audio is concerned, the content can be considered as explicit audio information or the implicit information related with the signal. The former relates to the sound, i.e., the signal level while the latter relates to the characteristics of, e.g., the rhythm, melody, structure, etc. Analyzing characteristics of musical audio, such as melody and rhythm, has a strong association to physiological and perceptual issues. Human listening comprises many stages and layers of information processing, from the treatment of low-level auditory stimuli in the inner-ear to the data handling in the brain. However, the latter can be influenced by high-level factors such as the memory, the context and personal experiences.

3.2.1 Musical Sound

Sound is the audible band of the mechanical wave spectrum, similar to the band of visible light within the electro-magnetic wave spectrum. Sound propagates as a traveling wave transmitted through a medium (a solid, liquid or gas).

A *musical sound* is often characterized with four perceptual attributes, *pitch*, *duration*, *loudness*, and *timbre*, which make it possible for the listener to distinguish musical sounds from each other:

- *Pitch* allows the tonal ordering of sounds from low to high on a scale. A sound has a certain pitch if it can be matched by adjusting the frequency of a sinusoidal wave of arbitrary amplitude [137]. The pitch often represents the perceived *fundamental frequency* of a sound [138]. However, the actual fundamental frequency may differ from the perceived pitch because of *harmonics* which have a frequency that is an integer multiple of the fundamental frequency.
- *Duration* corresponds to the time of the vibration which leads to the production of the sound.
- *Loudness* is the attribute of auditory sensation in terms of which sounds can be ordered on a scale extending from quiet to loud. It is related to the physical strength, i.e., the amplitude, of the signal.
- *Timbre*, also referred to as “sound color”, is closely related to the recognition of sound sources. If two musical sounds have equal pitch, duration and loudness, timbre is the property which enables us to distinguish both sounds from each other. Timbre is a multi-dimensional concept and depends mainly on the spectral energy distribution and its temporal evolution.

A *harmonic sound* can be decomposed into a sum of sine waves, i.e., the *harmonics*, whose frequencies are integer multiples of the lowest frequency as exemplified in Figure 3.1. The lowest frequency is named the *fundamental frequency* (F0) which is in the literature often referred to as the first harmonic. The integer multiple frequencies are then named the overtones. This way, harmonic sounds are periodic. However, many real-world instrumental sounds are not perfectly harmonic in the sense that the frequency components are almost integer multiples of the F0. Examples of such nearly-harmonic instruments are the piano, violin, flute, and also the human voice. These instruments are referred to as ‘pitched’ instruments since a distinctive pitch can be perceived.

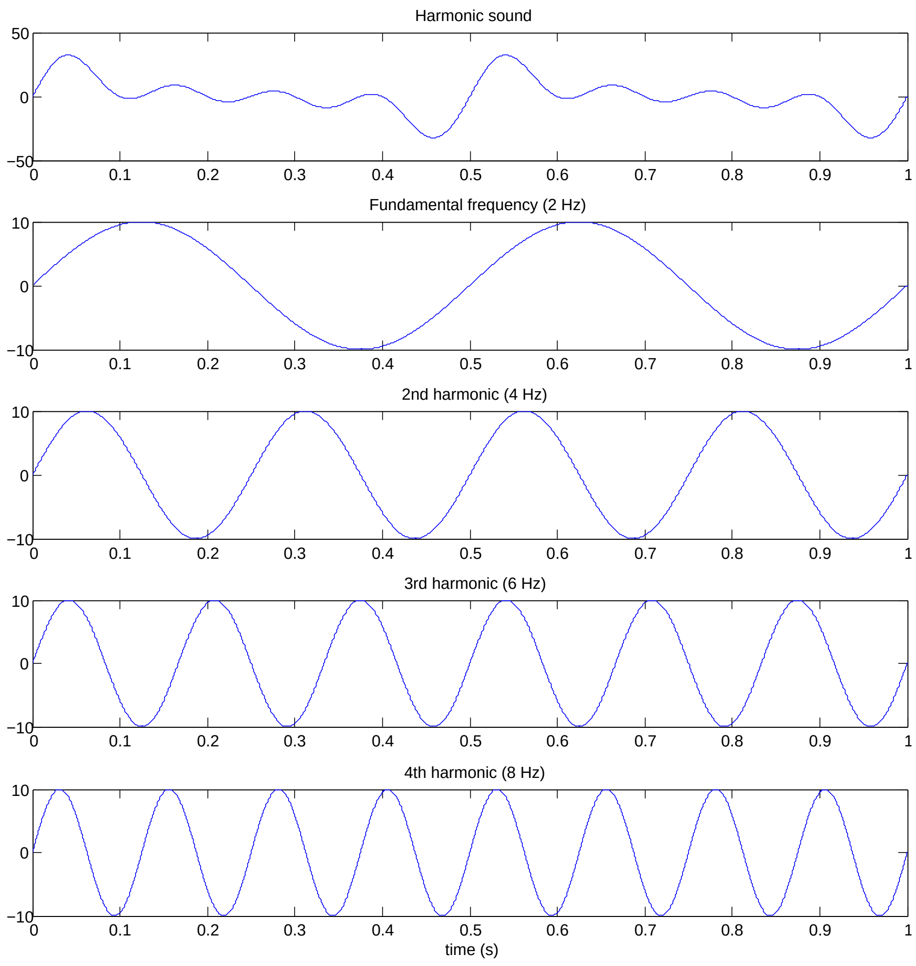


Figure 3.1: A harmonic sound decomposed into its harmonics.

Non-harmonic sounds have partials that are no integer multiples of the F0, such as percussive instruments. Therefore, these instruments are often called ‘non-pitched’ since no clear pitch can be perceived. Apart from that, *noise* has no characteristic frequency at all.

A pitch can be measured over a “short” time interval where it represents a *tone*. Remark that in the literature, the concept of a tone is often replaced by the concept *note*. In this dissertation, we make a clear distinction between these two concepts:

- A *tone* is a regular sinusoidal wave of a single frequency and a given duration.
- A *note* is the conceptualization of a tone on a musical scale (e.g., a chromatic scale). For example, a tone of 440 Hz corresponds to the note `A4`, i.e., the middle "la" on a piano clavier.

Over a "long" time interval, pitch corresponds to *tonality*, which is specified by hierarchic pitch relationships based on a harmonic center (see Section 3.4).

In the remainder of this dissertation, we make no distinction between sound and music because the difference between both concepts can be very subjective and is often driven by the musical background and the culture.

3.2.2 Music Transcription

Within audio MIR, music transcription is a process that aims to convert a musical audio signal, e.g., a musical recording on a magnetic tape or a digital recording stored as an mp3-file, into a symbolic representation (e.g., a musical score) that identifies the pitch, timings, rhythmic structure, and other features.

The usual process of manual music transcription typically proceeds in a top-down order. At first, the piece is segmented into meaningful parts and the rhythmic structure is recognized. Next, the instruments or musical objects of interest (e.g., the melody or the chords) are written down by repeatedly listening to the piece. Therefore, musicians commonly use an instrument to determine the note names while doing the transcription. People having a musical background can perform certain tasks, such as music transcription, generally with less effort than people lacking musical background. Furthermore, only a few people can actually directly name the absolute pitch of a sounding note or the tempo of the piece. Especially, the human ability to focus on a single instrument at a time, provided that it is somewhat audible in the mixture, is a useful property in music transcription. Manual music transcription is thus a highly specialized ability of the auditory system and it requires more effort than, e.g., visual scene analysis which occurs almost instantly.

According to the *polyphony* of the recording, music transcription can be more or less complex. The polyphony indicates the number of simultaneously sounding voices (whereas monophonic denotes that there is only one voice present at each time). For a higher number of simultaneous voices, it becomes more difficult to distinguish them. For manual music transcription, a monophonic recording may, e.g., require only one iteration whereas transcribing an entire symphony with instruments with similar timbres may turn out to be impossible [139]. Particularly, tonally fused (i.e., chimeric) sounds, which form a unified musical perception, may be difficult to distinguish and to separate into the corresponding individual musical tones. Actually, it is argued that trying to explicitly unveil the musical tones that are hidden in a chimeric sound is perceptually unnatural. In this sense, the mechanism of human music transcription must draw from a conscious mental effort, which demands substantial training and musical proficiency. Manual music transcription is thus rather time-consuming, error-prone, iterative, and requires specialized skills. Therefore, composers, music amateurs, and professionals could gain from automatic music transcription systems since these would free them for other more creative jobs.

Automatic music transcription systems are proposed as a means to overcome the above difficulties. In contrast to manual music transcription which occurs in top-down order, automatic music transcription is rather a bottom-up process. This is due to the fact that modeling such analytic listening and organization of musical sounds into entities is a very challenging problem. Generally, the architecture of such transcription systems comprises three main stages:

(i) *Spectral or frequency analysis* in which features are extracted from a frame of the original musical audio signal.

(ii) *Detection* of, e.g., pitches that represent the fundamental frequencies of melodic notes or classification of spectral data for drum detection.

(iii) *Event generation*, where the detected mid-level features are transformed into a desired symbolic representation, e.g., a score.

There exists a wide variety of different research topics in the domain of audio MIR processing concerning the analysis of music signals. In general, automatic music transcription tries to extract information from the musical content and includes several topics, such as pitch and multi-pitch estimation, the transcription of pitched instruments (detecting melody lines), transcription of percussive instruments (e.g., the snare drum, hihats), beat tracking and meter analysis, instrument recognition, and musical structure analysis. An important aspect of (Western) music is the melody. In the next section, we elaborate on this topic.

3.2.3 Melody

Successive sounds create temporal auditive patterns. Many problems in musical content retrieval center on the recognition, identification, and classification of patterns. Musicians as well as non-musicians seem to perform such tasks effortlessly and often unconsciously, such as having a sense of musical key [140], being able to follow the beat [141], and also following the melody of a song. A lot of research has been carried out about the way people build mental structures while listening to music and how musical patterns are remembered. Francès found that a figure (a short succession of pitches) is heard if it is higher in pitch than the rest. However, if the higher pitch is relatively constant, the lower figures form a more interesting pattern [142]. Another factor that influences the perception is the loudness. Furthermore, a listener can shift its attention to other musical parts.

Dowling discovered that the contour of a melody is easier to memorize than the exact melody [143]. The contour refers to the shape of the perceptual pitch (i.e., the F0) of the melody: the pitch goes up, stays the same, or goes down. In contrast, a musical scale is learned over a much longer period in life through listening to music. So perceptually, the relative changes of the pitch are more important than the absolute values. Other findings that are made concerning the perception of melody, are:

- Two melodies having the same contour and the same tonality (see Section 3.4) are perceived as more similar than two melodies having the same contour but a different tonality [143].
- It is harder to distinguish between two atonal melodies with the same contour than atonal melodies with a slightly different contour [144].
- When the octave of some notes are changed, it is harder to recognize the melody [145].
- As the pitch intervals (i.e., the timing between successive pitches) are altered, it becomes more difficult to identify the melody [144].
- Excact transpositions of melodies with a greater difference in pitch - regardless of relatedness of key- are considered less similar than those with a small difference in pitch [146].
- As the melodies become longer, pitch intervals become more important than the contour [147].

The timing between the pitches is thus also of importance. Consequently, two musical figures with the same contour, but with different intervals are perceptually different.

Thus, defining the notion melody is not so trivial. In effect, the concept of melody entails a certain subjectivity. Different people can have diverging perceptions of the same song about what the melody actually is. Furthermore, the concept of melody includes various aspects: melodies can be monophonic, contrapuntal (with two or more independent melody lines), pitched or rhythmic, tonal or atonal [148]. As a result, many definitions of melody relying on perceptual or musicological issues have been proposed. In the following, we list up various definitions of melody found in the literature, but we are not aiming to cover them all:

- Oxford Music Online[1] defines melody as: “A succession of notes, varying in pitch, which have an organized and recognizable shape. Melody is horizontal, i.e., the notes are heard consecutively”, but also as “the result of the interaction of rhythm and pitch”, and further as: “pitched sounds arranged in musical time in accordance with given cultural conventions and constraints”.
- “Musical sounds in a pleasant order and arrangement”[2].
- “A succession of rhythms and pitches” [149].
- “Melody is an organized sequence of consecutive notes and rests, usually performed by a lead singer or by a solo instrument” and also, “the melody is the part one often hums along when listening to a music piece” [150].
- “Melody is the dominant individual pitched line in a musical ensemble” [151].
- “A horizontal musical line of notation on the staff” and “melody is to a musical work what a paragraph is to a composition”. [149].
- “An auditory object that maintains its identity under certain transformation” [152].
- “A series of individual pitches, one occurring after another, so that the composite order of pitches constitutes a recognizable entity” [153].

[1]http://www.oxfordmusiconline.com
[2]http://www.wordsmyth.net/

- According to Wikipedia[3], a melody is: "A linear succession of musical tones which is perceived as a single entity. In its most literal sense, a melody is a sequence of pitches and durations, while, more figuratively, the term has occasionally been extended to include successions of other musical elements such as tone color".

As can be observed in the above list, various musicological, perceptual, and subjective properties are used to define the concept melody: (i) a sequence of pitched sounds, (ii) the musicological characteristics of a melody are exploited, (iii) the relation with rhythm is used, but also (iv) perceptual and emotional features are taken into consideration.

In the context of this dissertation, we define a melody as:

Definition. *Within a given time frame of a musical signal, the melody is defined by the time-pattern of the most dominant, and non-overlapping pitches.*

This definition reflects some of the above listed properties. At first, a melody is related to music. Second, an association with pitch is made. Furthermore, it also concerns the dominant pitch which relates our definition of melody with perception and the fact that there's only one dominant pitch at a time (i.e., non-overlapping). Third, melody concerns a pattern in time. The latter property relates melody with the notion of rhythm as it concerns the timing of successive pitches. Finally, a melody occurs in a given time frame. The latter is a very important aspect of our definition since the notion of melody is now not related with a musical piece as "a whole". This means that our definition of melody does not indicate we are looking for the pattern of pitches that is most easily remembered nor the tune that "stays in our head" after hearing a song, nor the pattern that occurs most often in a song. Thus, for each excerpt that can be characterized by pitch information, a melody can be found: one single tone, a continuously rising pitch, a succession of tones, and so forth. Remark that since noise has no characteristic pitch, it is not related with melody. Also according to our definition, a melody is not per se related with one instrument. Another effect of our definition of melody is that a melody can be represented as a timed series of tones, the frequency of each tone corresponds then to the F0.

Melody transcription (or melody detection) can thus be seen as a subtask of music transcription. Instead of identifying all musical events in an audio recording, melody transcription aims at finding the pitch information that is

[3]http://en.wikipedia.org/wiki/Melody

perceived by the listener as melody. Melody detection has a potential number of applications:

- *Query-by-melody* (QBM) is a process by which the melody is used as query and aims at finding musical items in a database that contain a similar melody. The latter requires that the melody information must be extracted from an audio recording. Remark that QBM can "easily" be performed on symbolic music representations, e.g., when the database contains score information (such as a collection of MIDI files). Query-by-melody could then be an intuitive way to query music collections in, e.g., a store. Such a query could either be aural, e.g., by humming which is then called query-by-humming (QBH) or by singing, i.e., query-by-singing (QBS), or a query could also contain symbolic information to represent a melody, e.g., MIDI data obtained from a synthesizer, or the Humdrum format [154] that describes the pitch contour.

- *Plagiarism detection* could gain from melody transcription as well. Indeed, both authors and copyright instances would have the possibility of automatically comparing songs based on melodic similarity measures, similarly like queries that are matched to melodies in QBM.

- *Performance* and *expressiveness analysis* by comparing the written and score could gather much information about the style of music performers. This is especially useful for analog instruments that cannot output symbolic (e.g., MIDI) information.

- For *music analysis*, the melodic part contains useful information for the detection of motives and themes. Hence, its automatic transcription could support this task.

3.2.4 Pitch Detection

The first major step in many music transcription methods and especially in melody transcription, is the detection of pitch information from the musical signal (as will also be later explained in Section 3.3.2). Pitch is the main low-level feature in melody detection and is one of the four characteristics of musical sound (see Section 3.2.1).

As a sound is usually made up of multiple sine waves, there's is no actual pitch. However, the human auditory system is anyway able to perceive a pitch which corresponds to its F0. Thus, to obtain the perceptual pitch of an arbitrary sound (over a certain time interval), it suffices to find the F0. An important property of the human auditory system is that the F0 does not need to be present in the signal to be perceived if the overtones are present. This concept is known as the *missing fundamental*. For example, from a signal that consists of 200 Hz, 300 Hz, 400 Hz and 500 Hz harmonics, the 100 Hz F0 can still be heard. Apparently, the brain processes the information present in the overtones to calculate the F0. For pitch perception, this implies that the frequency spectrum of the signal is at least as important as the F0. Licklider [155–157] proposed a neural processing model to explain how this information could be extracted from the auditory nerve. Others have elaborated this model such as Lyon [158], van Noorden [159], Slaney [160], Meddis, and Hewitt [161].

A straightforward way to obtain the frequency components of a signal, is by applying a Fourier transformation (FT). An FT converts a signal from the time-domain to the frequency-domain by decomposing the signal into complex exponential functions of different frequencies. However, an FT does not provide any information on the time at which a frequency component occurs. This is not a problem for stationary signals, but it is for non-stationary signals, such as speech and music since their frequency content changes over time. A way to overcome the latter and to obtain the time-frequency information, is to assume that the signal is stationary over a short time interval. For this purpose, a window function $w(t)$ is chosen. The width of this window must be equal to the segment of the (continuous) signal $s(t)$ where its stationarity is valid. Consequently, a moving window is then applied to the signal and the FT is applied to the signal within the window as the window is moved. This process is known as the short-time FT (STFT), with a window function w centered around τ:

$$STFT\{s(t)\} = \int_{-\infty}^{+\infty} w(t-\tau)s(t)\exp(-j\omega t)dt. \tag{3.1}$$

For a non-continuous signal $s[x]$, (3.1) becomes:

$$STFT\{s[x]\} = \sum_{x=-\infty}^{+\infty} w[x-m]s[x]\exp(-j\omega x), \tag{3.2}$$

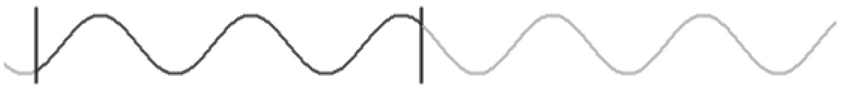
which is called the discrete STFT.

The size of the window has an important impact on the obtained results. A wide window gives better frequency resolution but a poor time resolution. On the other hand, a narrower window gives good time resolution but poor frequency resolution which is related to Heisenberg' s uncertainty principle.

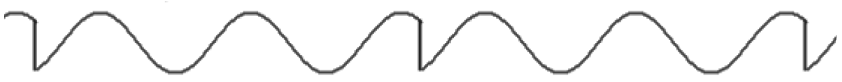
A more computationally efficient method is the Fast Fourier Transform (FFT). The FFT performs a Discrete Fourier Transform (DFT) on the discrete signal

$s[x]$ using a more efficient algorithm with time complexity $O(nlog(n))$ instead of $O(n^2)$ (n represents the number of data points) [162].

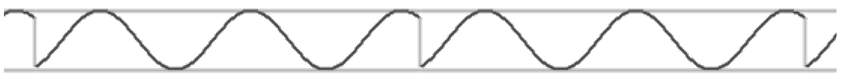
One of the major problems caused by the assumption that a signal is stationary over a short time interval (and thus of DFT algorithms in particular), is *spectral leakage*. When the end-points of a measured signal interval do not match up perfectly, discontinuities are introduced as illustrated in Figure 3.2(b). Consequently, the FT will introduce sharp discontinuities which are then processed into sinusoids and the spectral energy from these sinusoids spreads across the DFT bins from the fundamental frequency peak. A perfect sine does not have spectral leakage when the measured signal interval is in phase with the signal. The effects of spectral leakage can be reduced (or even increased) by



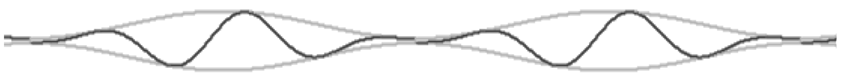
(a) The signal.



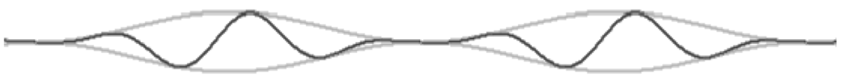
(b) The perceived signal without windowing.



(c) The perceived signal using a rectangular window.



(d) The perceived signal using a Hamming window.



(e) DFT perceived signal using a Hann window.

Figure 3.2: The effect of windowing functions on a sinusoidal signal.

the windowing function. Selecting an appropriate window function is not a simple task. Each window function has its own characteristics and suitability for different applications. To choose a window function, one should be able to estimate the frequency content of the signal. If the wavelength is shorter than the window, a rectangular window can be used. A rectangular window is the simplest window which just takes a chunk out of the signal: $w(n) = 1$. However, a rectangular window leads to discontinuities at the end-points (see Figure 3.2(c)). Therefore, windowing functions typically taper to zero near the end-points. In this way, the measured signal is attenuated at the end-points to ensure that the sharpness of discontinuities is dampened. Examples of commonly used window functions are the Hamming window ($\alpha = 0.54$, see Figure 3.2(d)) and the Hann window ($\alpha = 0.5$, see Figure 3.2(e)):

$$w(n) = \alpha - (1 - \alpha)\cos\left(\frac{2\pi n}{N-1}\right),$$

for $n = 0..N - 1$. For more information concerning the use of windowing functions for harmonic analysis, we refer to [163].

Many techniques for pitch detection from audio signals have been proposed in the literature, especially for speech analysis, e.g., [160, 164–171], but, no universal pitch detector currently exists. According to the way spectral information is handled, pitch detection techniques can be divided into three classes [172]:

(i) *Spectral location algorithms* are based on harmonic pattern matching which look for frequency components at harmonic locations. Popular examples are time-domain autocorrelation-based algorithms, and cepstrum-based frequency estimators [164, 173]. However, a major trade-off of spectral location algorithms is their inability to appropriately cope with non-ideal harmonic sounds. Inharmonicity is not a big concern in speech processing, but is immediately met when analyzing musical sounds at a wide frequency band.

(ii) *Spectral interval algorithms* are based on measuring the spectral intervals between frequency partials by applying autocorrelation on the spectrum, e.g., [166, 174]. These methods can cope relatively well with non-harmonic sounds. The idea is derived from the observation that a periodic but non-sinusoidal signal has a periodic magnitude spectrum, i.e., the period which is the F0. Spectral interval algorithms work better for sounds that exhibit inharmonicities. Even though the intervals do not remain constant, they are more stable than the locations of the harmonics.

(iii) *Unitary algorithms* provide a trade-off between the spectral interval and spectral location algorithms and evaluate the periodicity of the time-domain amplitude envelope. In the unitary approach, both timing and place information are taken into consideration through bandwise signal analysis followed by periodicity evaluation in each frequency channel. Auditory models typically apply this approach. The idea is derived from the observation that any signal with more than one frequency component exhibits periodic fluctuations in its time-domain amplitude envelope (i.e., beating). The frequency components alternatingly amplify and cancel each other. The rate of beating depends on the frequency difference between each two frequency components.

As explained in appendix D, the human auditory system performs pitch detection in the cochlea by the stimulation of hair cells in the organ of Corti that resides on the basilar membrane. Furthermore, there are two theories that explain the perception of pitch in the human auditory system. The place theory uses the evidence that different places in the basilar membrane of the inner ear respond to different frequencies. On the other hand, the frequency theory is based on the fact that the part of the basilar membrane that responds best to a given frequency component tends to vibrate at the frequency of that component as well. The auditory system could then use this information to infer the spectrum of the analyzed sound. Models of human pitch perception attempt to unify these theories into one model that is able to reproduce a wide range of phenomena in human pitch perception [160, 175, 176]. Both timing and place information are taken into consideration by separating the analysis into different frequency bands. As such, increased robustness to corrupted signals can be achieved.

3.3 Melody Transcription

In this section, we present an automatic transcription system for the detection of the melody from musical audio. We aim to distinguish individual tones characterized by specific temporal boundaries without imposing any restrictions upon the instrumentation nor the polyphony. In Section 3.3.1, we outline related work already done in this field. Next, in Section 3.3.2, we describe the front-end which is a pitch detector based on an auditory model. The second phase of our approach consists of the estimation of the F0 in each time-frame and the creation of pitch trajectories over successive frames. This is explained in Section 3.3.3. Next, the creation of melodic tones is clarified in Section 3.3.4. The final phase of the presented system is a post-processing to remove redundant tones and is explained in Section 3.3.5. In Section 3.3.6, we give an overview of the presented system. Finally in Section 3.3.7, the evaluation of the presented melody transcription system is discussed, the main encountered problems are addressed, and possible improvements are suggested.

3.3.1 Related Work

The first melody transcription mechanism focused on monophonic recordings of instruments with a relative strong first harmonic (e.g., flutes), playing at a consistent tempo [177]. This system used a STFT as front-end and formulated note hypotheses based on amplitude information for automatic score generation. Currently, tracking the pitch of a monophonic musical piece is practically a solved problem, but the quantization of pitches into notes is still a difficult problem (especially for singing) [172]. The latter is mainly due to performance aspects, such as vibrato and portamento.

In contrast, performing pitch detection on a polyphonic musical piece is a much more demanding task. In a polyphonic context, several instruments are usually playing at the same time. As a result, harmonic components of different concurrent sounds coincide in frequency. The transcription of the melody from polyphonic audio data is still recognized as an unsolved problem. While humans easily spot the melody from various musical pieces, there is still no reliable method for the automatic extraction of the melody.

To locate melodic fragments, the predominant pitches in the input signal need to be estimated. Consequently, the first step in a melody transcription system is the detection of this pitch information.

Over the last decade, there has been a remarkable progress in the area of melody transcription from polyphonic music. The work of Goto received particular attention [178–180]. The front-end of these systems consists of a STFT-based multirate filter bank which generates a number of pitch candidates for every 10 ms. In the second stage, a probabilistic model for the detection of melody and bass lines is devised. This model assumes that the obtained pitch distribution at each frame is generated from a weighted mixture of tone models. Each tone model has a harmonic structure and is modeled as a Gaussian distribution centered at integer multiples of the corresponding F0. Further, the melody then corresponds to the most predominant tone model with the highest weight. The weights are iteratively updated using the Expectation Maximization algorithm. By tracing the F0 candidates at consecutive frames, the melody line is formed. One of the shortcomings of this method is that no discrimination of the melody and the accompaniment is performed. This limitation was addressed by Marolt who used features such as loudness, pitch stability, and onset steepness in order to perform sound source separation [181]. However, the accuracy varied considerably across different musical excerpts. Tapert and Batke also made some adaptations on the F0 estimation method of Goto. Their F0 tracking agents are implemented similarly to those in Goto's work. The agents contain four time frames of F0 probability vectors (two of the past, the actual and the upcoming frame). To find the path of the predominant frequency, all maximal values over four frames within an agent are added and discontinuities are punished by diminishing. Finally, the agent with the highest score decides the F0.

Eggink and Brown suggest a methodology for extracting the melody line played by a solo instrument in a mixture [182]. After the identification of F0 candidates using an STFT-based front-end, pitch tracks are formed. The main melodic path is looked up in a network comprising all possible candidates over time. This is supported by various local and temporal knowledge sources (e.g., F0 strength, instrument likelihood, interval likelihood) and subject to some constraints. Instrument recognition receives particular attention here, and the likelihood that a particular tone corresponds to the solo instrument is estimated in each frame. A major drawback of this mechanism is that it requires the solo instrument to be known in advance. Further, the frame-based instrument recognition does not perform accurately.

Paiva et al. apply the auditory model of Slaney and Lyon [176] for the pitch detection phase [183]. Next, pitch trajectories are obtained using the peak continuation algorithm of Serra [184]. The method of Serra looks for regions of stable sinusoids in the signal's spectrum, which leads to a trajectory for each harmonic component found. Next, a frequency-based segmentation is performed, which aims to split notes of different pitches that may be present in the same trajectory. Next, a segmentation based on pitch salience minima is performed to split consecutive notes at the same pitch and to mark the limits of each note. Since the author aims to find a MIDI representation, the pitches are in an early stage quantized into (MIDI) notes. However, this early quantization of frequencies to MIDI notes is only valid if the instruments are well-tuned and, furthermore, nowadays music production systems can easily change the pitch of an (instrumental) voice so it matches the accompaniment which does not necessarily need to be 'correctly' tuned.

In Dressler's approach, sinusoidal-tracks are used as the front-end for predominant F0 extraction [185]. First, spectral analysis is performed via an STFT filter bank and candidate spectral peaks are detected in each frame. The kernel of the approach is the pitch estimation module where a perceptually-based magnitude weighting is carried out and the harmonic structure is examined. Next, perceptual cues of sound organization, namely frequency and magnitude proximity, are used to connect consecutive pitches. The melodic pitch line is then identified by a rule-based scheme, (e.g., intervals above the octave are avoided) and notes from middle or higher pitch registers are preferred. This method also resorts to the identification of the most active frequency regions. In this way, the weights of the pitch lines belonging to such regions are increased. In [186], a more elaborate peak selection method has been applied relying on a magnitude threshold which depends on the signal. Then the instantaneous frequency for the selected peaks is computed. In order to obtain more stable frequency measures, the average of two estimation methods is used. Furher, the actual estimation of tone height and tone magnitude is performed as an independent computation: harmonic peaks are added to existing tone objects and after a few frames, a timbre representation for that tone is established. This way the impact of noise and other sound sources can be decreased.

Seokhwan and Chang [187] also rely on a STFT as the first phase for the melody extraction. They assume that melody follows a Markov process. They estimate melody parameters using sequential importance sampling which is a conventional particle filter method. Their method outperforms other well-known methods using the ISMIR 2004 Audio Description Contest data set [188]. However, they only use 16 (out of 20) songs of this data set and thus more tests on realistic data are needed.

Joo et al [189] first extract a fixed number of pitch candidate, and then apply a rule-based algorithm to extract the melody. A major limitation of their approach is that they limit the melody range to one octave.

Poliner and Ellis tackle the melody detection problem as a classification task [190]. At first, normalized STFT coefficients are acquired in each time frame. Next, these data are classified by a support vector machine which is trained on multi-instrument recordings as well as synthesized MIDI audio. As a result, the input data are mapped to the corresponding target frequencies. Then, melodic vs. non-melodic discrimination is performed by energy thresholding. The uniqueness of this approach is that no assumptions about the spectral structure are made, which is in contrast to the other methods which rely on (harmonic) frequency structures.

Despite all these attempts, melody transcription from polyphonic audio is still an unsolved problem. The methods described above mainly rely on a STFT-based pitch extraction as the first step. However, since melody is a perceptual phenomenon, we opt to start from an auditory model like Paiva et al. [183]. This way, a more robust pitch detection can be obtained. Then, instead of ap-

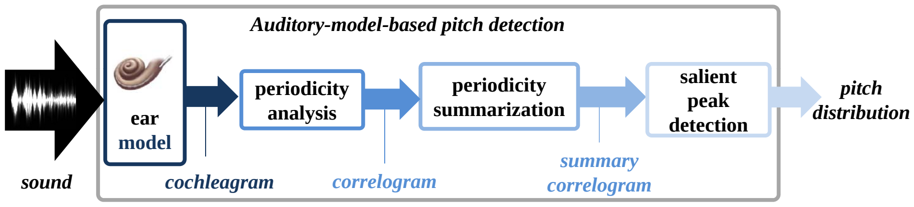


Figure 3.3: Schematic overview of our auditory-model-based pitch detector.

plying a peak continuation algorithm, we propose a method to estimate the F0 relying on Gaussian mixture models. As such, the melody detection problem can be viewed as an F0 estimation problem. The continuation of the F0 over adjacent frames can be considered as a high-order Markov process. But, instead of using a probabilistic framework, we apply a rule-based approach using an adapted version of Goto's agent-based approach. In the next sections, we describe our auditory-model-based melody detection system.

3.3.2 Auditory-Model-Based Pitch Detection

The first phase in the proposed melody transcription system is the detection of pitch information from the musical signal. The pitch detector presented in this dissertation is based on the auditory model of Slaney and Lyon [176]. This auditory model outputs a correlogram which corresponds to the output of Licklider's duplex theory which states that, next to a frequency analysis, the auditory system also performs an autocorrelation analysis [191]. Briefly summarized, our pitch detector consists of four stages, as depicted in Figure 3.3:

(i) The application of an auditory model (or *ear* model) on a sound wave results in a *cochleagram* which consists of auditory nerve patterns for each frequency channel.

(ii) A *correlogram* is obtained by analyzing the periodicities using autocorrelation in each frequency channel.

(iii) The global periodicities are calculated by a summation which results in a summary *correlogram*.

(iv) To obtain the n most salient pitch candidates, the highest n peaks in the summary correlogram are selected.

Computation of Cochleagram

The model of Slaney and Lyon implements three main tasks: filtering, detection, and compression [176]. The cochlear model described by Lyon combines a series of band-stop filters which model the traveling pressure waves with resonators into basilar membrane motion or velocity. The acoustic wave is filtered by a band-stop filter at each point in the cochlea. Each filter operates at successively lower frequencies in order to low-pass the pressure wave. At the same time, resonators pick out a small range of the traveling energy and model the conversion into basilar membrane motion which is detected by the inner hair cells. The filters act as a frequency analyzer and each filter corresponds to a cochlear channel that best responds to a particular frequency range. Furthermore, front filters are also implemented, which constitute a simple model of the responses of the outer and middle ear.

After filtering, the movements of the basilar membrane are converted into auditory nerve responses. Since inner hair cells only respond to movement in one direction, an array of half-wave rectifiers is employed to detect the output of each filter. Finally, four stages of automatic gain control compress the dynamic range of the input into a limited level that the auditory nerve can deal with. The automatic gain control is, in fact, a model of ear's adaptation: the response to a constant stimulus is first large and then, as the auditory system adapts to the stimulus, the response becomes smaller. For a more detailed overview of the auditory model, we refer to the work of Licklider [191] and Lyon [192].

We use the Auditory Model Toolbox which is a Matlab implementation by Slaney of the ear model of Lyon [192, 193]. Regarding parameterization, the parameters proposed by Slaney are used again [193]. The output of this auditory model is a multi-channel representation of auditory nerve firing patterns and permits the visualization of the sound as a time-frequency image. In this image, the *cochleagram* or *auditory nerve image*, each line contains information regarding auditory nerve responses for the corresponding frequency channel. Figure 3.4 shows the cochlear data obtained from this auditory model of the note `C4` sampled at 44.1 kHz from a piano sound on a Yamaha QS300 synthesizer.

Computation of the Correlogram

After computing the auditory nerve firing responses for each frequency channel, the main periodicities are detected in a correlogram. The correlogram summarizes the temporal activity at the output of the cochlea [176]. This is accomplished by computing the autocorrelation in each channel, which results in a two-dimensional image of the sound signal. The horizontal axis represents

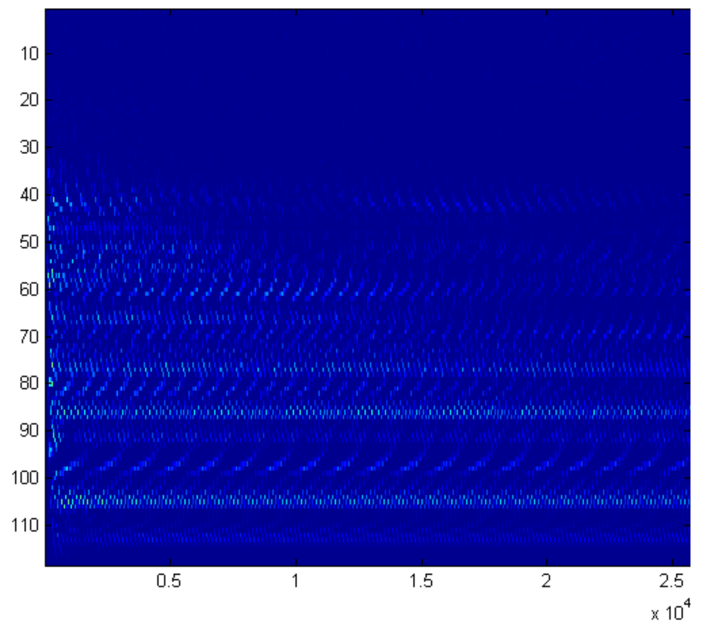


Figure 3.4: The corresponding cochleagram of the note `C4` from a piano sound at 44.1 kHz – the vertical axis represents the filters (frequency channels) and the horizontal axis is the time lag ($\times$10 ms).

the correlation lag and the vertical axis represents the frequency channels. All channels will show peaks at the horizontal positions corresponding to correlation lags which, on their turn, correspond to the periods of repetition present in the signal. Slaney and Lyon argue that the correlogram is biologically plausible. In reality, few researches suggest that the brain measures periodicities using a neural delay line. This case is supported by the cross-correlator structures found in the brains of owls and cats for spatial localization, but the structures that could compute the correlogram for pitch have yet to be found [160]. Other schemes, e.g., based on mechanical delays in the cochlea, have been proposed to implement a correlation [194]. Figure 3.5 depicts the corresponding correlogram of the cochleagram depicted in Figure 3.4, computed using a Hann window of width 40 ms and a step size of 10 ms.

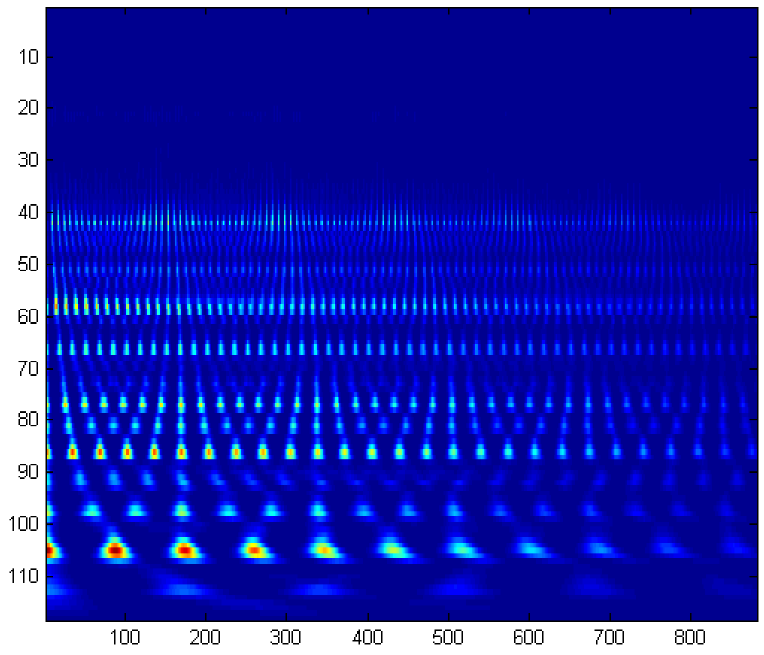


Figure 3.5: Correlogram from the note C4 from a piano sound – the vertical axis represents the filters (frequency channels) and the horizontal axis is the correlation lag.

Periodicity Summarization

The next step is the creation of a summary correlogram by summing up the autocorrelation functions across all channels at each time lag. The summary correlogram quantifies the likelihood that a periodicity corresponding to a particular time lag is present in the sound wave.

Salient Peak Detection

The next task is to pick the most salient peaks in the summary correlogram. For a sample rate s, a peak at time lag L (ms) in the summary correlogram indicates the presence of a pitch with frequency $f_L = s/L$. This process is exemplified in Figure 3.6 which depicts the summary correlogram that is calculated from the correlogram shown in Figure 3.5 and its $n = 4$ most salient peaks. Remark that the peak at zero time lag is not considered since the autocorrelation function has a maximum when the signal is compared to itself. For a sample rate of 44.1 kHz, the highest peak occurs at time lag 167 ms which corresponds to the frequency 264.1 Hz. The other three peaks, in decreasing

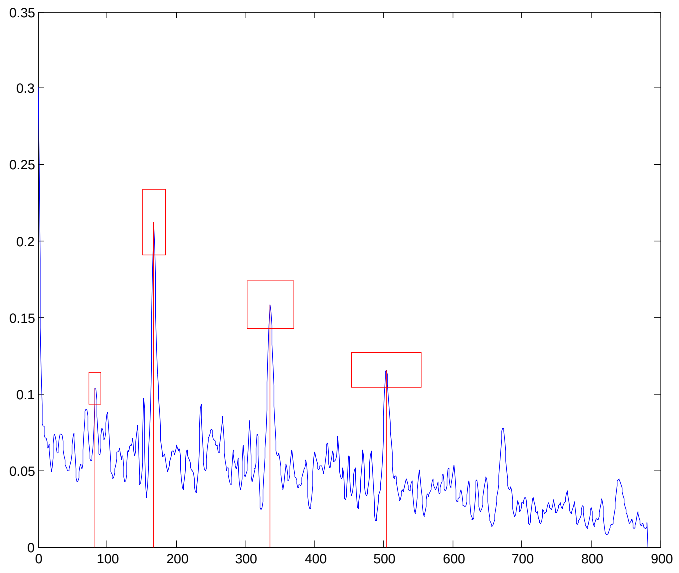


Figure 3.6: Summary correlogram of the note C4 from a piano sound sampled at 44.1 kHz and its 4 most salient peaks – the horizontal axis represents the time lag in ms and the vertical axis is the self-correlation.

salience order, correspond to 131.6 Hz, 87.7 Hz, and 537.8 Hz and thus occur at nearly (sub)harmonic positions.

The pitch evidence is calculated by dividing the value of the summary correlogram at the corresponding peak position with the value of the summary correlogram at zero time lag. Highly periodic sounds with an easily perceivable pitch will have a salience close to 1 while aperiodic sounds will have a salience closer to 0.

Subharmonic Summation

In the next stage, a post-processing is applied on the pitch evidences. According to the subharmonic summation theory, salient peaks in the summary correlogram often occur at positions that correspond to subharmonics of the fundamental frequencies. The subharmonic summation theory states that each spectral component generates a series of subharmonics that give rise to the perception of pitch [195]. To reduce the evidence of peaks in the summary correlogram at positions that correspond to subharmonics, post-processing is applied in the following way. In each time frame t, the evidence $ev_{t,i}$ of the ith pitch (i.e., the ith peak in the summary correlogram) having a frequency $f_{t,i}$ is adjusted if a subharmonic $f_{t,j}, i \neq j$ is present, by:

$$\overline{ev}_t(i) = ev_t(i)(1 + \gamma^{1/\eta}),$$

where $0 < \gamma < 1$ is a constant and η is the subharmonic degree:

$$\eta = \frac{f_{t,i}}{f_{t,j}}.$$

This adjustment will thus increase the evidence of a frequency if there are peaks found in the summary correlogram that correspond to subharmonics.

Evaluation

Due to perceptual phenomena (e.g., the missing fundamental), evaluating a pitch detector which is intended for melody transcription, is not a trivial task. In this evaluation, we test the accuracy of the pitch detector to find the F0 in polyphonic musical audio since its significance in the melody detection process. For the evaluation of the proposed auditory-model-based pitch detector, we have used the MIREX 2005 training set. This set consists of 13 polyphonic musical audio files together with their annotated F0s at a resolution of 10 ms. At the positions where the instrument that plays the melody line (i.e., the leading instrument) is silent, the annotated F0 is 0. Therefore, the detected pitches at these positions are not taken into account. For evaluation, a detected pitch f_{t_1} at time t_1 matches with the ground truth pitch g_{t_1} at time t_1 if $\frac{|g_{t_1} - f_{t_1}|}{g_{t_1}} \leq 0.05$.

Besides the auditory-model-based pitch detector, we also considered an FFT-algorithm [196] for pitch detection. Table 3.1 depicts the F0 detection results by considering the most salient pitch using an FFT-based and the auditory-model-based pitch detector. It can be observed that the FFT-based pitch detector has a remarkably lower accuracy compared to the auditory-model-based approach.

Furthermore, it is noticeable for both methods that the accuracy for the last four training files is remarkably lower than for the other ones. This is mainly because the proportion between the intensity of the melodic instrument and the intensity of the background music in these musical audio files is much lower. Consequently, the spectral information of the melodic instrument is less prominent in the final sound spectrum.

The results in Table 3.1 indicate that the extraction of a single pitch per frame is thus not enough to accurately detect the F0. Therefore, we have to extract

ID	training file	FFT	proposed
1	*train01.wav*	0.43	0.76
2	*train02.wav*	0.44	0.56
3	*train03.wav*	0.26	0.72
4	*train04.wav*	0.22	0.67
5	*train05.wav*	0.62	0.75
6	*train06.wav*	0.40	0.56
7	*train07.wav*	0.64	0.73
8	*train08.wav*	0.29	0.79
9	*train09.wav*	0.31	0.74
10	*train10.wav*	0.43	0.21
11	*train11.wav*	0.13	0.44
12	*train12.wav*	0.05	0.30
13	*train13MIDI.wav*	0.03	0.28
average		0.33	0.58

Table 3.1: Accuracy of F0 detection from the MIREX 2005 training set by detecting the most salient pitch.

multiple salient pitches per frame. Table 3.2 lists the accuracies of the F0 detection using the auditory-model-based pitch detector if the 2, 3, 4, and 5 most salient pitches are considered. We immediately notice that by considering the 2 most salient pitches, the accuracy already increases with an average of 14% over the 13 test files. When more pitches are considered per frame, the accuracy increases (compared to the extraction of one pitch) with 19%, 22% and 24% for 3, 4, and 5 pitches, respectively.

Figure 3.7 shows the pitch extraction results using the proposed pitch detector applied on a sinusoidal wave of 200 Hz. As can be seen, the most salient pitch has indeed a frequency of 200 Hz, but also its subharmonics of 100 Hz (2nd highest salience), 66.7 Hz (3rd highest salience), and 50 Hz (4th highest salience) are detected, but also 327 Hz (which is an overtone of 66.7 Hz). The detection of the subharmonics of 200 Hz is in agreement with the subharmonic summation theory. However, this simple example also indicates that, next to subharmonics, it is also possible that overtones (of subharmonics) are detected. So, by considering multiple peaks from the summary correlogram, peaks that are not related to (sub)harmonics may be selected, so special attention has to be paid.

	number of pitches			
ID	2	3	4	5
1	0.87	0.89	0.93	0.94
2	0.71	0.79	0.86	0.93
3	0.79	0.84	0.89	0.91
4	0.78	0.83	0.85	0.88
5	0.91	0.94	0.97	0.98
6	0.74	0.80	0.85	0.89
7	0.82	0.90	0.98	0.99
8	0.87	0.89	0.91	0.93
9	0.82	0.87	0.92	0.95
10	0.28	0.31	0.32	0.33
11	0.67	0.69	0.70	0.71
12	0.49	0.54	0.55	0.56
13	0.56	0.62	0.64	0.66
average	0.72	0.76	0.80	0.82

Table 3.2: Accuracy of the presented auditory-model-based pitch detector for the detection of the F0 by considering the 2, 3, 4 and 5 most salient pitches on the MIREX 2005 training set.

3.3.3 F0 Estimation

The previous section made clear that we have to extract multiple pitches per frame in order to increase the probability of the F0 detection. However, in many cases, the F0 is not always found. Furthermore, the F0 can be lacking in the signal due to the missing fundamental phenomenon. Thus, whether the F0 is detected or not, the F0 has to be estimated from the extracted pitches. Consequently, the second phase of the melody transcription is the estimation of the F0 in each time frame. The method presented in this dissertation is based on the approach of Goto [178]. We start from the assumption that the pitches (obtained from the presented auditory-model-based pitch detection method) are generated from a model that is a weighted mixture model of harmonic-structure tone models. Higher weights indicate more dominant tone models. So, the maximum weight model should correspond to the most dominant harmonic-structure model with a known F0.

At first, the observed pitches in Hz are transformed into cents so that we use a linear scale. A frequency f_{Hz} is converted into a frequency f_{cents} by using the

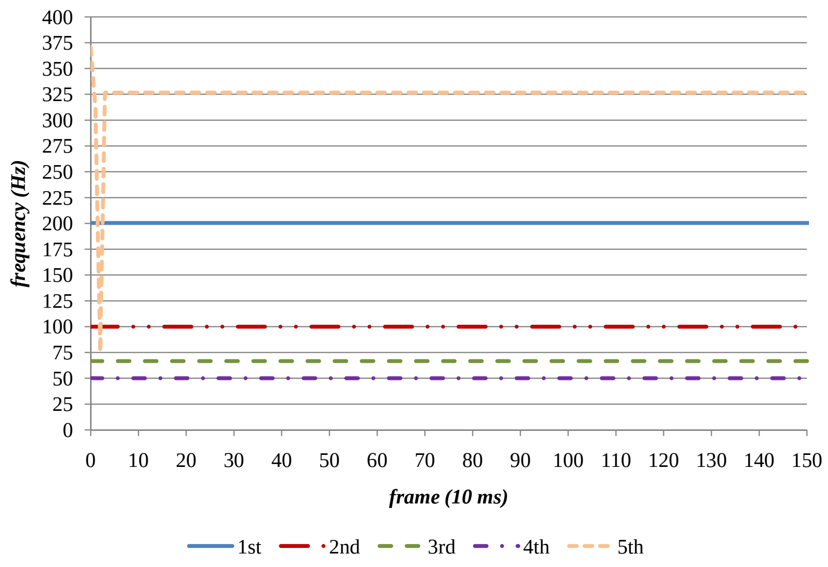


Figure 3.7: Detection of the five most salient pitches of a 200 Hz sinusoidal wave using the auditory-model-based pitch detector.

middle A (440 Hz = 5700 cents) as a reference:

$$f_{cents} = 1200 \log_2 \frac{f_{Hz}}{440 \times 2^{3/12-5}}. \tag{3.3}$$

This way, an octave consists of 1200 cents and a semitone is 100 cents. In the remainder of this dissertation, the notation f is used instead of f_{cents}.

Let Ω_t denote the set of frequencies that are found in frame t (where $|\Omega_t| = n, \forall t$) and $\overline{ev}_t(f)$ the adjusted evidence of the frequency f as explained in the previous section.

Next, the n pitches (in cents) f at time frame t, i.e., $f \in \Omega_t$, are represented as a PDF $p_t(f)$:

$$p_t(f) = \frac{\overline{ev}_t(f)}{P_t}, \tag{3.4}$$

where

$$P_t = \sum_f \overline{ev}_t(f). \tag{3.5}$$

Remark that $\overline{ev}_t(f) = 0$ for $f \notin \Omega_t$, and $\sum_f p_t(f) = 1, \forall t$.

Further, we consider a following harmonic tone model that indicates where the subharmonics of a fundamental frequency F occur:

$$p(f|F) = \sum_{h=1}^{N} c(h) G(f; F - 1200 \log_2 h, W), \tag{3.6}$$

where $G(x; \mu, \sigma)$ is a Gaussian distribution with mean μ and standard deviation σ:

$$G(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right).$$

The parameters of the tone model are: N the total number of subharmonics and $c(h) = G(h; 0, H)$ denotes the (relative) amplitude of the hth subharmonic where H is a constant. Remark that if a pitch detection method mainly detects the overtones of the F0 (instead of its subharmonics), the mean of the Gaussian should become $F + 1200 \log_2 h$.

The more dominant a tone model $p(f|F)$ in the sound mixture, the higher the probability of the F0 of its model. Therefore, the probability of each tone model represents the relative dominance of its harmonic structure, i.e., its weight in the mixture of tone models. We consider that each observed PDF $p_t(f)$ has been generated from a weighted-mixture model $p(f|\theta(t))$ of tone models $p(f|F)$ of all the possible F0s, i.e.:

$$p(f|\theta(t)) = \int_{F_{min}}^{F_{max}} w_t(F) p(f|F) dF, \tag{3.7}$$

where $\theta(t) = \{w_t(F) | F_{min} \leq F \leq F_{max}\}$ with F being the F0 at time t, and $w_t(F)$ is the weight of the tone model $p(f|F)$ at time t so that:

$$\int_{F_{min}}^{F_{max}} w_t(F) dF = 1, \forall t. \tag{3.8}$$

Further, F_{min} and F_{max} represent the minimum and maximum allowed F0, respectively (the melody typically occurs in a range so it is not necessary to analyze the complete frequency range). We need thus to estimate the parameter $\theta(t)$ (i.e., the weigths of each tone model in the mixture) so that the observed distribution $p_t(f)$ is most likely being generated from the model $p(f|\theta(t))$.

The maximum likelihood estimator of $\theta(t)$ is achieved by maximizing the mean log-likelihood [197]:

$$\int_{-\infty}^{+\infty} p_t(f) \log p(f|\theta(t)) df. \tag{3.9}$$

This maximization problem can be solved using the Expectation-Maximization (EM) algorithm which is a method for finding maximum likelihood estimates of parameters [198]. The EM algorithm is an iterative procedure which consists of two successive steps: (i) the expectation step (E-step) and (ii) the maximization step (M-step). In our case, the observed data are $p_t(f)$ and $\theta(t)$ is iteratively updated using the previous estimates $\theta'(t)$.

The E-step calculates the expected value of the mean log-likelihood function:

$$Q(\theta(t)|\theta'(t)) = \int_{-\infty}^{+\infty} p_t(f) E_F[\log p(f, F, \theta(t))|f, \theta'(t)]df, \tag{3.10}$$

where $E_F[g|f, c]$ denotes the conditional expectation of g with respect to F and conditional distribution c given f:

$$E_F[\log(p(f, F, \theta(t)))|f, \theta'(t)] = \int_{F_{min}}^{F_{max}} p(F|f, \theta'(t)) \log p(f, F, \theta(t))dF.$$

Consequently,

$$\begin{aligned} Q(\theta(t)|\theta'(t)) &= \int_{-\infty}^{+\infty}\int_{F_{min}}^{F_{max}} p_t(f)p(F|f, \theta'(t)) \log(p(f, F, \theta(t)))dFdf \\ &= \int_{-\infty}^{+\infty}\int_{F_{min}}^{F_{max}} p_t(f)p(F|f, \theta'(t)) \log(w_t(F)p(f|F))dFdf \\ &= \int_{-\infty}^{+\infty}\int_{F_{min}}^{F_{max}} p_t(f)p(F|f, \theta'(t)) \log(w_t(F))dFdf \\ &+ \int_{-\infty}^{+\infty}\int_{F_{min}}^{F_{max}} p_t(f) \log(p(f|F))dFdf. \end{aligned}$$

The M-step maximizes $Q(\theta(t)|\theta'(t))$:

$$\overline{\theta(t)} = \underset{\theta(t)}{\operatorname{argmax}}(Q(\theta(t)|\theta'(t))). \tag{3.11}$$

The prior condition is given by equation (3.8), i.e.:

$$\begin{aligned} &\int_{F_{min}}^{F_{max}} w_t(F)dF = 1, \\ &\int_{F_{min}}^{F_{max}} w_t(F)dF = \int_{F_{min}}^{F_{max}} \frac{dF}{F_{max} - F_{min}}, \\ &\int_{F_{min}}^{F_{max}} \left(w_t(F) - \frac{1}{F_{max} - F_{min}}\right) dF = 0. \end{aligned}$$

So, we can introduce a Lagrange multiplier λ and the Euler-Lagrange equation Λ to solve this conditional maximization problem:

$$\begin{aligned}
\Lambda(w_t, F, \lambda) &= Q(\theta(t)|\theta'(t)) - \lambda \int_{F_{min}}^{F_{max}} \left(w_t(F) - \frac{1}{F_{max} - F_{min}} \right) dF, \\
\frac{\partial^2 \Lambda(w_t, F, \lambda)}{\partial F \partial w_t} &= \frac{\partial}{\partial w_t} \left(\int_{-\infty}^{+\infty} p_t(f) p(F|f, \theta'(t)) \log\left(w_t(F)\right) df \right) \\
&+ \frac{\partial}{\partial w_t} \left(\int_{-\infty}^{+\infty} p_t(f) p(F|f, \theta'(t)) \log\left(p(f|F)\right) df \right) \\
&- \frac{\partial}{\partial w_t} \left(\lambda \left(w_t(F) - \frac{1}{F_{max} - F_{min}} \right) \right) \\
&= 0.
\end{aligned}$$

Consequently,

$$w_t(F) = \frac{1}{\lambda} \int_{-\infty}^{+\infty} p_t(f) p(F|f, \theta'(t)) df.$$

Given equation (3.8), $\lambda = 1$. By applying Bayes' theorem, we further obtain:

$$\begin{aligned}
p(F|f, \theta'(t)) &= \frac{p(F)p(f|F)}{p(f, \theta'(t)|F)} \\
&= \frac{w'_t(F)p(f|F)}{\int_{F_{min}}^{F_{max}} w'_t(y)p(f|y)dy}.
\end{aligned}$$

So, a new value w_t^i for the weight w_t^{i-1} at frame t can be iteratively computed by:

$$w_t^i(F) = \int_{-\infty}^{+\infty} p_t(f) \frac{w_t^{i-1}(F)p(f|F)}{\int_{F_{min}}^{F_{max}} w_t^{i-1}(y)p(f|y)dy} df. \tag{3.12}$$

with the initial weigths $w_t^0(F) = \frac{1}{F_{max} - F_{min}}$, $F \in [F_{min}, F_{max}]$.

The F0 of a frame t is then characterized by the frequency F with the maximum weight $w_t(F)$ as given by equation (3.12) in the last iteration.

3.3.4 Tone Creation

The final result of the fundamental frequency estimation process is not stable over adjacent frames because peaks corresponding to the F0 of several simultaneous instruments sometimes compete in the PDFs. Therefore, it is necessary to consider the global temporal continuity of these peaks. Hence, our tone detection process consists of two parts: (i) a peak tracing algorithm and (ii) a tone segmentation algorithm.

Peak Tracing

This method sequentially traces for peak trajectories in the temporal transition of the weights of the tone models, in order to select the most dominant and stable frequency trajectories. To perform this, an architecture consisting of a salient peak detector and multiple agents is used.

Salient Peak Detection

A peak detection algorithm detects the local maxima in the PDF at frame t to find the frequencies with the highest probability $w_t(F)$. Therefore, the peakmap $M_t(F)$ is created as follows:

$$M_t(F) = \begin{cases} P_t w_t(F) & \text{if } \frac{\partial w_t(F)}{\partial F} = 0 \text{ and } \frac{\partial^2 w_t(F)}{\partial F^2} < 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3.13}$$

where P_t is given by equation 3.5. The set of salient peaks Φ_t at frame t is given by the peaks that are higher than a threshold T_p of the maximum peak in the PDF:

$$\Phi_t = \left\{ F | M_t(F) \geq \max_F (M_t(F)) T_p \right\}. \tag{3.14}$$

Then, we define the salience degree of a peak by tracing a temporary trajectory in the near future. The traced peak frequency at time t is $N_t(F) = F$ and at time $t + \tau, \tau > 0$ is given by:

$$N_{t+\tau}(F) = N_{t+\tau-1}(F) + \underset{f<2W}{\operatorname{argmax}} (\phi_{t+\tau}(f; F)). \tag{3.15}$$

W is the standard deviation of a Gaussian distribution $G(f; 0, W)$ with zero mean and $\phi_{t+\tau}(f; F)$ expresses the possibility which indicates how much the peak frequency is likely to change by f from the previous allocated peak frequency $N_{t+\tau-1}(F)$:

$$\phi_{t+\tau}(f; F) = M_{t+\tau}(N_{t+\tau-1}(F) + f) G(f; 0, W). \tag{3.16}$$

Finally, we propose to compute the salience degree of a salient peak $F \in \Phi_t$ at frame t by:

$$S_t(F) = \frac{M_t(F)G(0;0,W)}{(1+Per)SP_t} + \sum_{\tau=1}^{Per} \max_{f<2W} (\phi_{t+\tau}(f;F)), \tag{3.17}$$

where Per is the number of frames for tracing a temporary trajectory in the near future and

$$SP_t = \frac{1}{1+Per} \sum_{\tau=1}^{Per} P_{t+\tau}. \tag{3.18}$$

Agent-Based Pitch Tracing

Salient peaks in close proximity of each other are traced by agents. An agent has a set of parameters which are updated every frame:

- Fr_t, the frequency of the current peak.
- Sa_t, the salience degree of the current peak $S_t Fr_t$.
- Pen_t, the penalty; the agent is inactivated if $Pen_t \geq T_{pen}$ (a threshold).
- $SPow_t$, the cumulative sum of SP_t since the agent is active.
- Re_t, the reliability calculated as $SPow_t Sa_t$.

In the first (non-empty) frame, an agent is created for each salient peak $F_t \in \Phi_t$. For the next frames, a peak candidate is selected using the following procedure:

(i) A candidate peak C_t is looked up in Φ_t.

(ii) If no peak can be found in Φ_t, the penalty Pen_t is increased and a candidate peak is looked up in M_t.

(iii) If no peak can be found in M_t, the penalty Pen_t is increased.

And, the following conditions apply to C_t:

(i) C_t is closest to the previously allocated peak frequency Fr_{t-1}: $C_t = \text{argmax}_F(S_t(F)G(F;Fr_{t-1},W))$.

(ii) C_t should be close enough to Fr_{t-1}: $|C_t - Fr_{t-1}| < 2W$.

If a candidate peak is claimed by more than one agent, the peak is allocated by the most dominant agent. In this context, we define the most dominant agent with the highest reliability Re_t. Further, active agents can allocate at most one peak at a time. Upon allocation of a candidate peak C_t at frame t, the agent's parameters are updated accordingly:

- $Fr_t = C_t$,
- $Sa_t = S_t(C_t)$,
- $SPow_t = SPow_{t-1} + P_t$,
- $Re_t = SPow_t Sa_t$.

If after processing of the current frame t, the most salient peak $F_1 = \text{argmax}_F\,(\Phi_t)$ and second most salient peak $F_2 = \text{argmax}_{F \neq F_1}\,(\Phi_t)$ are not allocated, a new agent is created for those peaks. The process of salient peak detection and agent-based peak tracing is exemplified in Figure 3.8.

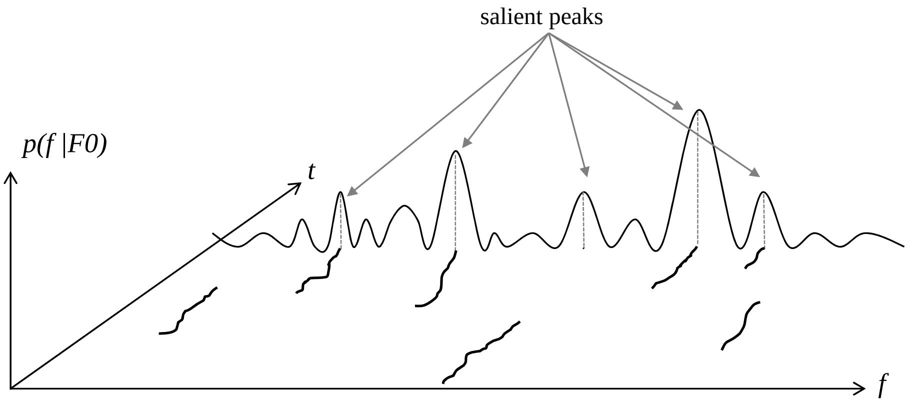


Figure 3.8: Tracing of salient peaks in the PDF of the current frame.

Tone Segmentation

As explained in Section 3.2.1, a tone is defined by a (constant) frequency, start time, and duration. Each agent outputs a trajectory of frequencies between its start and end time. Due to the fact that the frequency-estimation is not stable,

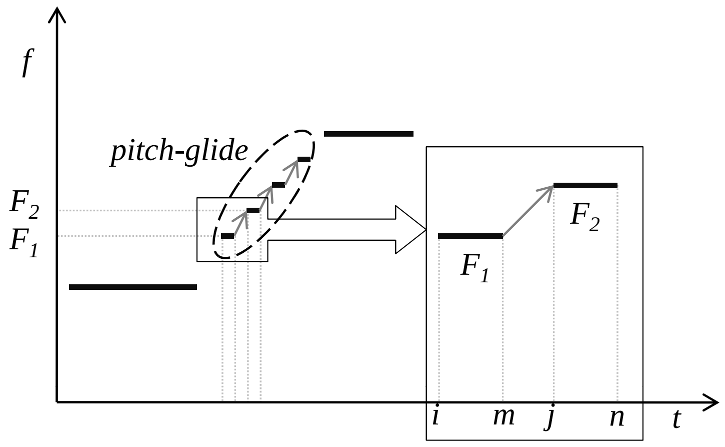


Figure 3.9: Detection of a pitch-glide from frequency F_1 to frequency F_2.

a trajectory can still contain small local fluctuations. Thus, in order to obtain a note representation, these fluctuations must be eliminated. Furthermore, a trajectory can slide from a tone to another adjacent tone. Also pitch-glides can be traced by agents. A pitch-glide is characterized by a continuous increase or decrease in frequency from one frequency to another. Consequently, a segmentation algorithm must take these cases into account.

Pitch-Glide Detection

A pitch-glide between two frequencies F_b and F_e is characterized by a continuous increase or decrease in frequency from time b to time $e, b < e$ and $F_b \neq F_e$. To detect pitch-glides, we introduce a maximum frequency difference ΔF between F_b and F_e and, a maximum allowed inter-frame distance Δt are introduced. Now, suppose that in some frame m a frequency F_1 was lastly allocated by some agent A^i_m with start frame i, and in another frame $n, m < n$, frequency F_2 was lastly allocated by an agent A'^j_n such that $i \leq m < j \leq n$ as depicted in Figure 3.9. If $n - m \leq \Delta t$ and $|F_e - F_b| \leq \Delta F$, then both trajectories are considered as a pitch-glide and are linked together. Pitch-glides are excluded from the segmentation and the averaging phase (see below). After the pitch-glide detection, trajectories are segmented into tone(s) using the segmentation approach presented in next section.

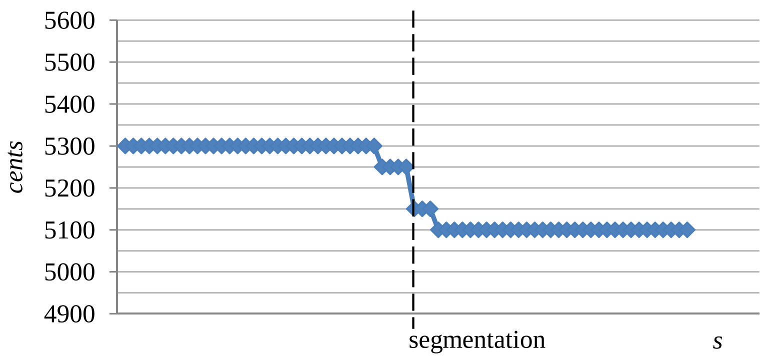


Figure 3.10: The pitch trajectory is segmented with $\delta = 150$ cents so that two new segments, $\{5300{+}5250\}$ cents and $\{5150{+}5100\}$ cents, are created.

Segmentation

Denote the maximum and minimum allocated peak frequency by an agent as F_{max} and F_{min}, respectively. Further, let A_j denote the set of all allocated frequencies by the agent with start frame t_b and end frame t_e, $t_b \leq t_e$, and δ the maximum allowed frequency fluctuation. At any frame t, let $F_{max,t} = \max(F_i | i \leq t)$ and $F_{min,t} = \min(F_i | i \leq t)$, i.e., the maximum and minimum frequency traced so far. If there exists a frequency $F_k, k \leq t_e$ for which $|F_{max,k} - F_k| \geq \delta$ or $|F_{min,k} - F_k| \geq \delta$, then a segment $\Theta_{t_b,k} = \{F_i | t_b \leq i \leq k\}$ is created. If there is no such frequency F_k, the whole trajectory is considered as a segment Θ_{t_b,t_e}. For every segment, both conditions $|F_i - F_{max,i}| < \delta$ and $|F_i - F_{min,i}| < \delta$ are satisfied. The outcome of this process is shown in Figure 3.10 for $\delta = 150$ cents.

Averaging

After the segmentation of the trajectories, the frequency of a segment Θ_{t_1,t_2} can be safely set to a constant value $F(\Theta_{t_1,t_2})$ by calculating its integer average value, see in Figure 3.11:

$$F(\Theta_{t_1,t_2}) = round\left(\frac{\sum_{F_i \in \Theta_{t1,t2}} F_i}{|\Theta_{t_1,t_2}|}\right). \tag{3.19}$$

$|\Theta_{t_1,t_2}|$ denotes the number of allocated frequencies. Remark that $|\Theta_{t_1,t_2}|$ is not equal to the total number of frames $(t_2 - t_1 + 1)$ since a trajectory can contain small gaps. These gaps correspond to frames where no (salient) peaks

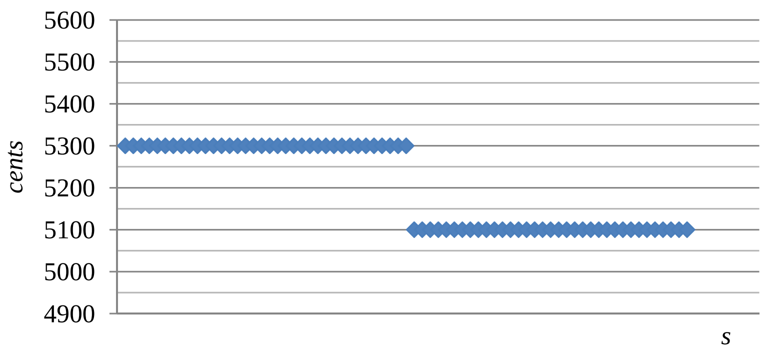


Figure 3.11: The previously obtained segments are averaged. Consequently, two tones of 5300 and 5100 cents are created.

could be found after the F0-estimation or by frames where no pitches were extracted.

3.3.5 Post-processing

The final result of the tone segmentation phase is an ordered set of tones and pitch-glides. However, some of those tones and pitch-glides may be redundant. The latter can be caused by:

- The fundamental frequency estimation created peaks at harmonics of the extracted frequency components which on their turn could be caused by the pitch extraction that produced frequency components at harmonic positions.
- Too short tones.
- Octave errors.
- Concurrent tones at octaves or at harmonic-related positions.

Therefore, we propose a collection of post-processing methods that are run in the following order to filter out redundant parts:

(i) *Keep the most reliable tones*. If simultaneous tones exist, tones with the highest accumulated reliability, i.e., $\sum_{t=t_b}^{t_e} Re_t$, are kept.

(ii) *Keep the highest temporal reliability Re_t*. Simultaneous tones are split up such that at each frame t, the corresponding Re_t is maximal.

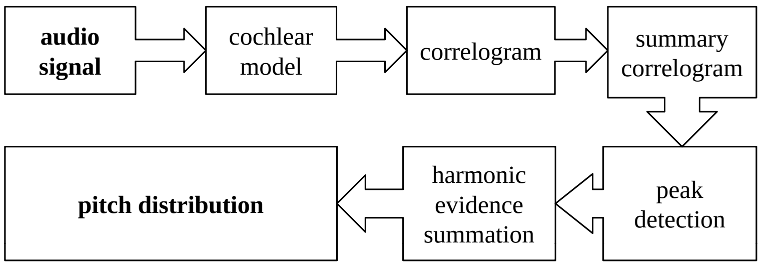


Figure 3.12: Schematic overview of the auditory-model-based pitch detector.

(iii) *Remove short tones.* Tones (or pitch-glides) with a length smaller than a predefined threshold L_{min} are removed.

(iv) *Remove tones with a low reliability.* Tones (or pitch-glides) with $\sum_{t=t_b}^{t_e} Re_t$ that is lower than a predefined fraction $0 < re_{min} < 1$ of the average Re_t of all agents are removed.

3.3.6 System Overview

The architecture of our system that transcribes a raw musical audio signal into a melody consists of four phases: (i) pitch detection, (ii) F0 estimation and pitch tracing, (iii) tone creation, and (iv) filtering. The first phase of the system is an auditory-model-based pitch detector that takes as input an audio signal and generates a pitch distribution.

The auditory-model-based pitch detection algorithm consists of the following stages as outlined in Figure 3.12:

(i) The application of an ear model that mimics the operation of the ear, and in particular of the cochlea in the inner ear. The output of this stage is a cochleagram which consists of auditory nerve patterns for each frequency channel.

(ii) A correlogram is obtained by analyzing the periodicities using autocorrelation in each frequency channel of the cochleagram,

(iii) The global periodicities are calculated by a summation over all channels in the correlogram which results in a summary correlogram.

(iv) Pitch candidates are obtained from the summary correlogram by detecting its highest peaks.

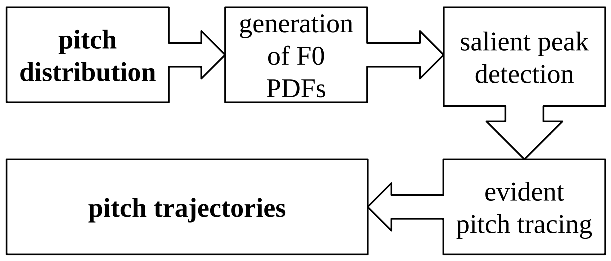


Figure 3.13: Agent-based tracing of evident pitches.

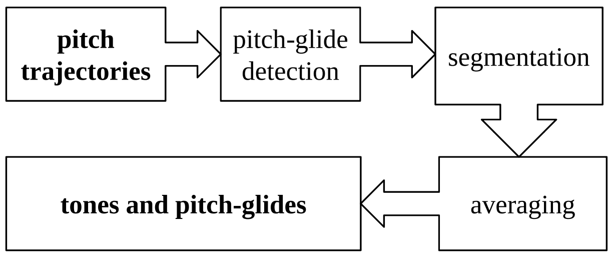


Figure 3.14: Schematic overview of the tone creation process.

(v) The evidence of a pitch is adjusted according to the presence of subharmonics.

The second phase in the melody transcription system concerns the estimation of the fundamental frequency and the tracing of these frequencies over multiple frames. This phase takes as input the previously calculated pitch distribution and outputs a set of pitch trajectories. This process is schematically represented in Figure 3.13 and consists of the following stages:

(i) The generation of a pitch probability density function in each frame.

(ii) The detection of salient peaks in the pitch probability density functions.

(iii) The creation of pitch trajectories by using an agent-based approach over different time frames.

The third phase of our architecture is a tone creation algorithm that takes as input the pitch trajectories and creates a set of tones and pitch-glides.

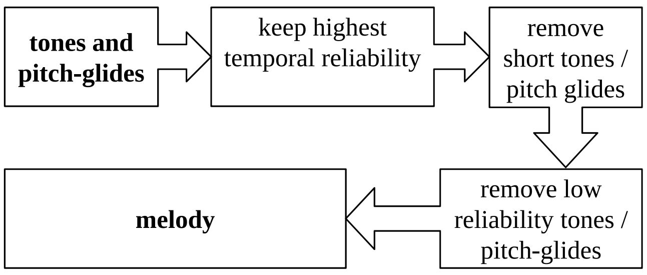

tones and pitch-glides
keep highest temporal reliability
remove short tones / pitch glides
remove low reliability tones / pitch-glides
melody

Figure 3.15: Post-processing applied on the obtained tones and pitch-glides.

As outlined in Figure 3.14, the tone creation process consists of the following stages:

- Pitch-glide detection.
- Segmentation of trajectories.
- Averaging the frequency of a segment.

The final phase of the melody transcription is a post-processing which keeps the highest temporal reliability and includes the filtering of redundant segments. In case of (partially) simultaneoussegments, the parts with the lowest reliability are removed, short segments and segments with a low reliability are also deleted as outlined in Figure 3.15.

3.3.7 Experimental Results

The results and evaluation of the above presented melody transcription system are discussed in the next paragraphs. The main problems are addressed and possible improvements are suggested.

The melody transcription system is evaluated using two different data sets: (i) a self-created set of melodies acquired from popular songs (further referred to as GMEL) and (ii) the MIREX 2005 training set which consists of 13 melodies (see also Section 3.3.2).

The GMEL data set consists of 34 polyphonic musical excerpts containing a melody. These melodies of different durations are obtained from 'popular' songs, from which title and performer are listed in Table 3.3. Further, the melody can either be vocal, instrumental, or a combination of both. So, no restrictions are implied upon the instrumentation of the melody. For the ground truth, MIDI files containing the melody track have been manually created so that the MIDI notes match with the melody in the audio excerpts. The MIREX 2005 training set consists of 13 polyphonic musical audio files together with their annotated F0s at a resolution of 10 ms. At the positions where no "melodic tone" is heard, the annotated F0 is 0. In contrast to the GMEL set, the annotation of the MIREX 2005 set is not quantized (to MIDI notes) but is a sequence of frequencies.

The parameters of the melody transcription system are set as follows. The auditory-model-based pitch detection module detects the $n = 4$ most salient pitches every $t = 10$ ms (window size: 40 ms, type: Hann window). We empirically observed that if more than 4 pitches per frame are extracted, the F0-estimation phase yields more errors since low-salience pitches are not always related to the F0.

Further, the F0 is estimated at intervals of 50 cents (a half semitone) and pitch trajectories are created between $F_{min} = 3000$ cents (≈ 92 Hz) and $F_{max} = 9000$ cents (≈ 2960 Hz), a range where melody lines typically occur. The threshold to identify salient peaks in the F0 its PDF is set to $T_p = 0.4$ (see equation 3.14). The total number of subharmonics that are considered, is set to $N = 5$ and the standard deviation of the amplitude of the subharmonics is experimentally set to $H = 2.7$. The period of tracing a temporary trajectory in the near future is set to 5 frames: $Per = 5$ (see equation 3.17 and equation 3.18). The parameters that concern the operation of an agent are chosen so that the penalty can be increased at most 7 times before an agent is deactivated. In the tone segmentation phase, pitch-glides are detected for $\Delta F = 100$ cents and $\Delta t = 40$ ms (4 frames). Further, the maximum frequency difference to segment pitch trajectories is set to $\delta = 150$ cents. Finally, short tones of length smaller than $L_{min} = 100$ ms (10 frames) are removed. Further, the threshold re_{min} is set to 0.05.

Evaluation

The performance of the system is expressed in terms of precision and recall (see equations 2.12 and 2.13, where f_n (false negatives) denotes the number of undetected melodic tones, f_p (false positives) the number of wrongly detected melodic tones, and t_p (true positives) the number of correctly detected melodic tones). Correctly identified empty frames (i.e., frames containing no melodic pitch information in both the transcription and the ground truth) are excluded. At first, an important remark about the difference between annotations of the

ID	composer	title	duration
1	2 Brothers On The 4th Floor	Come Take My Hand	11123
2	2 Unlimited	Tribal Dance (p1)	7359
3	2 Unlimited	Tribal Dance (p2)	14708
4	Aqua	Barbie Girl	7354
5	Harold Faltermeyer	Axel F	6836
6	Darude	Sand Storm	7003
7	Dido	Thank You	5777
8	Dj Sash	Encore Une Fois	7027
9	Dj Taucher	Infinity	6234
10	Elton John	Candle In The Wind	6905
11	Enigma	Callas Went Away (p1)	16683
12	Ennio Moricone	Chimai	16171
13	Enya	Lothlorien	14756
14	Europe	Final Count Down	8494
15	Kabouter Plop	Kabouterdans (p1)	12741
16	Kabouter Plop	Kabouterdans (p2)	8213
17	Kernkraft 400	Zombie Nation	4999
18	La Bouche	Be My Lover	13458
19	Paul Van Dyk	For An Angel	6620
20	Pet Shop Boys	Go West	15493
21	Pet Shop Boys	It's A Sin	10680
22	Playahitti	The Summer Is Magic	14174
23	Prodigy	No Good	6202
24	Robert Armani	Hit Hard	6654
25	Robert Miles	One And One	7099
26	Softcell	Tainted Love (p1)	3220
27	Softcell	Tainted Love (p2)	6654
28	DJ Tiesto	Love Comes Again	8046
29	DJ Tiesto	Traffic	6571
30	Vengaboys	We Like To Party	6999
31	Age Of Love	Age Of Love	6889
32	Enigma	Callas Went Away (p2)	5303
33	Mozart	Piano Concerto 11	7384
34	Francois Van Campenhout	Brabanconne	17381
		total duration (ms)	311210

Table 3.3: The GMEL melody collection set and the duration of each melody in ms.

GMEL set and the MIREX 2005 training set has to be made. Since the ground truth of the GMEL set is derived from MIDI files, the pitch information of a MIDI note will be constant (at least if their is no pitch-bend information available in the MIDI-file) and this can be matched easily with the generated output of our system since a fixed frequency corresponds to a note (e.g., in our tuning `A4` = 440 Hz). In contrast, the ground truth of the MIREX 2005 training set is a sequence of unquantized and continuously altering frequencies. As a matter of consequence, this representation highly differs from the representation used by our system. Because of this, a frequency sequence has to be mapped on the outputted note data. It is obvious that the latter can give rise to quantization errors. To reduce these quantization errors, a ground truth MIDI note with number G_m and a detected tone T (in cents) match if $|100G_m - T| < 50$, thus if their distance is less than a half semitone (50 cents), which is the resolution of our melody transcription system.

Table 3.4 and Table 3.5 list the precision and recall of the MIREX 2005 and GMEL melody detection set, respectively. The 2nd and the 3rd column of both tables list the results for raw pitch detection of the melody line (i.e., octave errors count for an error) while in the 4th and 5th column, the chroma is considered (i.e., octave errors are not taken into consideration). For the MIREX 2005 training set, the (time-weighted) average precision and recall for raw pitch detection are 0.68 and 0.65, respectively. For chroma, the average precision and recall are 0.81 and 0.79, respectively. For the GMEL set, the (time-weighted) average precision and recall are 0.71 and 0.86, and for chroma detection 0.72 and 0.87, respectively. To compare with other algorithms, the best and second best results from the MIREX 2005 melody extraction contest concerning raw pitch accuracy are 69% and 68% respectively, and for chroma 74% and 71%, respectively. The best and second best results for raw pitch detection in the MIREX 2009 contest using the MIREX 2005 data set are 76% and 75%, respectively and for chroma data 81% and 81%, respectively (the pitch accuracy of the MIREX competition is measured in terms of precision).

For both the GMEL and the MIREX 2005 set in our evaluation, we immediately notice that the precision and recall values of the audio tracks where the melody is sung (i.e., a human voice) are generally lower than for the tracks where the melody is played by an instrument (for the GMEL set, these are the tracks with IDs 1, 3, 4, 7, 10, 15, 18, 20, 22, 23, 25, 27, 28, 30, and for the MIREX 2005 set, these are the tracks with IDs 1–9). Consequently, the performance using the MIREX 2005 training set is lower than for the GMEL set since the former mainly contains sung melodies. The observation that the performance of the transcription system using sung melody is lower, can be

	(raw pitch)		(chroma)	
ID	**precision**	**recall**	**precision**	**recall**
1	0.62	0.74	0.71	0.81
2	0.62	0.64	0.76	0.77
3	0.44	0.52	0.64	0.71
4	0.52	0.64	0.71	0.80
5	0.59	0.81	0.71	0.88
6	0.56	0.74	0.66	0.81
7	0.62	0.64	0.76	0.77
8	0.65	0.58	0.76	0.70
9	0.62	0.64	0.76	0.77
10	0.98	0.49	0.99	0.78
11	1.00	0.74	1.00	0.83
12	1.00	0.74	1.00	0.85
13	1.00	0.62	1.00	0.81
weighted average	0.68	0.65	0.81	0.79

Table 3.4: Performance of the presented melody transcription system using the MIREX 2005 set.

explained by a couple of factors.

At first, the pitch of singing is often less stable than the pitch obtained from an instrument, especially near the onset. Further, a sung part may exhibit (slight) fluctuations in the frequency, a phenomenon which is also known as vibrato (which can be either be stylistic effect or caused by 'bad' singing). Consequently, the tone creation process will introduce rounding errors as it averages the frequencies. This process is exemplified in Figure 3.16, which depicts an excerpt from the song "Candle In The Wind" (Elton John). The detected pitches (dots) converge to (i.e., portamento) (a) and fluctuate around (i.e., vibrato) (b) the intended pitch of 4700 cents while the detected tones (bars) represent the average pitch of the obtained segment. Between 0.5 and 1.0 s (a), the detected pitch is erroneously rounded to the average of the obtained segment. The fluctuations of the pitch between 1.0 and 1.65 s (b) cause the creation of small tones which can be erroneously deleted in the tone filtering stage. Also, the quantization to 50 cents generates some errors.

Secondly, the spectrum of the human voice thoroughly differs from many instruments as it contains meaningful frequency components, known as formants, which are essential to distinguish vowels in speech. These formants do not necessarily match with the harmonics of the fundamental pitch during

	(raw pitch)		(chroma)	
ID	**precision**	**recall**	**precision**	**recall**
1	0.69	0.88	0.69	0.88
2	0.98	0.98	0.98	0.98
3	0.55	0.67	0.55	0.67
4	0.74	0.86	0.76	0.92
5	0.83	0.94	0.83	0.94
6	0.96	0.98	0.96	0.98
7	0.91	0.86	0.91	0.86
8	0.58	0.93	0.58	0.93
9	0.94	0.90	0.94	0.90
10	0.32	0.74	0.32	0.74
11	0.85	0.89	0.88	0.92
12	0.77	0.99	0.79	0.99
13	0.53	0.87	0.53	0.87
14	0.60	0.82	0.60	0.82
15	0.63	0.76	0.69	0.81
16	0.62	0.82	0.67	0.84
17	0.67	0.78	0.73	0.87
18	0.51	0.93	0.51	0.93
19	0.88	0.75	0.88	0.75
20	0.88	0.94	0.88	0.94
21	0.87	0.90	0.87	0.90
22	0.61	0.87	0.61	0.87
23	0.74	0.84	0.74	0.84
24	0.67	0.69	0.67	0.69
25	0.81	0.90	0.81	0.90
26	0.77	1.00	0.77	1.00
27	0.57	0.70	0.57	0.70
28	0.74	0.90	0.74	0.90
29	0.92	0.94	0.92	0.94
30	0.96	0.97	0.96	0.97
31	0.97	0.97	0.97	0.97
32	0.67	0.51	0.67	0.51
33	0.70	0.80	0.70	0.80
34	0.37	0.80	0.49	0.87
weighted average	0.71	0.86	0.72	0.87

Table 3.5: Performance of the presented melody transcription system using the GMEL set.

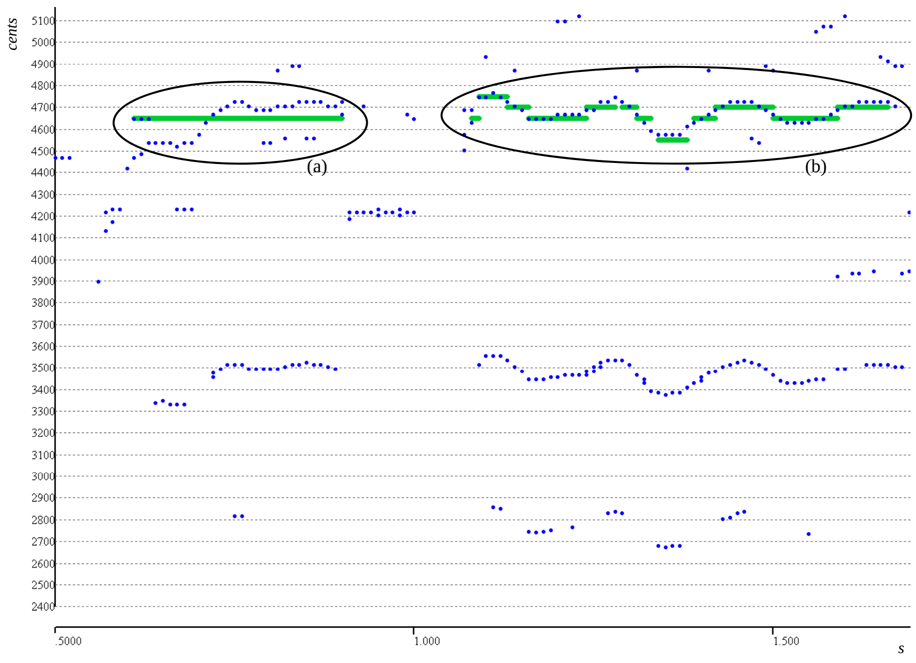


Figure 3.16: Fluctuations in the pitch of the song “Candle in the Wind” (Elton John): (a) portamento and (b) vibrato.

singing. Furthermore, in singing, the pitch is determined by multiple harmonics. If a formant coincides with such a harmonic, this harmonic becomes very salient in the spectrum. Consequently, the corresponding frequency (which differs from the fundamental), is picked up by the transcription system. This process explains the following example. Figure 3.17 depicts the detected pitches (dots) and the detected tone (bar) from an excerpt (between 8.5 and 9 s) of file 3 from the MIREX 2005 training set. The pitch detector finds pitches near 3000, 4200, 5400 and 6600 cents – remark that they are an octave apart and that they are all (integer) multiples of 600 cents. According to the ground truth, the pitch should be 4200 cents. However, the transcription system keeps the pitch track at 6600 cents (2 octaves higher) since the salience is highest at this frequency. The latter is not only due to the presence of the subharmonics, which increase the pitch evidence (cf. the subharmonic summation theory), but also without the evidence adjustment, the most salient pitch occurs at 6600 cents ($\approx$ 740 Hz). Most likely, a formant of the sung vowel coincides at this frequency.

A way to tackle the problems related to sung melody lines, is the detection of vocal and non-vocal regions in the musical signal. Once a segment is identified

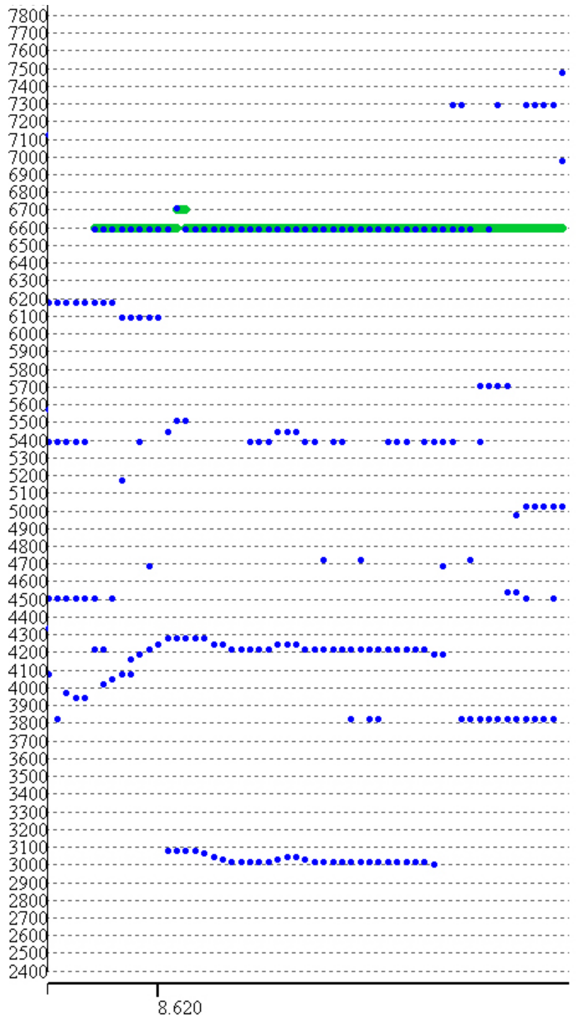


Figure 3.17: An excerpt of the MIREX 2005 training set. The transcription system proposes 6600 cents as F0 since a formant coincides at this frequency while the ground truth indicates 4200 cents as F0.

as vocal, the pitch detection and tone creation phase can deal with the specific characteristics of voices [199]. Voice detection in music is not a widely studied subject, but most of its ideas come from speech recognition topics, which is a well-established research area. Vibrato, harmonicity, timbre, and cepstral coefficients, such as Mel-Frequency Cepstral Coefficients (MFCC) [200], are commonly used to detect voice segments in audio [201–204].

Further, we also noticed that (octave) errors typically occur in excerpts where multiple instruments that play the melody line are involved. As a consequence, the frequencies that correspond to another instrument, which can be more salient in certain time frames, can be retained in the filtering phase. However, abrupt F0 changes are not very common in melodies. This knowledge can help us to scale the range of a detected tone, so that the frequency interval with the average of the previously detected tones must be less than, e.g., an octave. Consequently, by rescaling these intervals, the overall precision and recall increases.

Another melody transcription problem is caused by the fact that the algorithm outputs the most salient tones at each time frame. However, when the instrument that plays the melody line is silent, other salient frequencies are also looked up. Consequently, spectral information of the accompaniment is used for the construction of the F0 hypothesis, which can lead to a false positive (i.e., a redundant tone in the detected melody). We observed that such tones often have a lower salience and a shorter duration than the tones related to the melody line. In this way, many redundant tones can be filtered away. Currently, the salience- and duration-based tone filtering is threshold-based. However, this filtering could be enhanced by a more thorough analysis where the temporal contour of the salience and duration is considered and minima in these contours can be deleted. Ideally, timbre information about the leading instrument (i.e., the instrument that plays the melody line) could help to identify regions where the leading instrument is silent. However, this involves the recognition of instruments in polyphonic contexts and is a complex task due to spectral overlapping between concurrent instruments [205].

During the F0 estimation, certain parameters such as the number of subharmonics, the amplitude of the subharmonics, and the standard deviation of the these amplitudes are set to a constant value. Ideally, these parameters could also be estimated [180].

Further improvements can also be conducted at the front-end, i.e., the pitch detection phase. In the current implementation, the four most salient pitches (that correspond to the four most salient peaks in the summary correlogram) are retained every frame. However, the salience of some pitches is often relatively small compared to the most evident pitch(es), so they can be discarded. Actually, in the tone filtering stage, the resulting low-salience tones are removed to avoid some superfluous computations. Analogously, other possibly important pitch information is not used as only the four most salient pitches are considered and some peaks in the summary correlogram might have similar heights (especially in vocal parts because of the presence of harmonics and formants). A more detailed pitch information can be useful for a better F0 estimation. Further, the peaks are looked up in the summary correlogram what can lead to peak masking. Therefore, the auditory-model-based pitch detector could be improved by carrying out frequency analysis in each channel of the correlogram, especially in noisy frequency regions where peak masking is more likely to occur. Furthermore, lateral inhibition (i.e., the phenomenon that the presence of a salient frequency makes the auditory system less sensitive for frequencies to sounds of about the same frequency) is disregarded in the presented peak-picking algorithm. Finally, introducing high-level musical knowledge such as musical key information can also help to adjust the frequencies of the notes. In the next section, we present a method to obtain the key from musical audio recordings

3.4 Tonality Induction

As the perception of pitch over a short time interval gives rise to the notion of melody, the perception of pitch over a longer interval corresponds to *tonality*. In particular the sense of musical key, also called tonality, has been a topic of many investigations (see [206] for an overview). It is a principal phenomenon of Western music, well-documented over several centuries, and appealing from the viewpoint of cognitive information processing because it seems to involve both bottom-up as well as top-down processing strategies. Knowledge in this area is based on converging evidence from different fields of research, in particular experimental (behavioral) psychology [207], brain research [208], and computer modeling [209].

The pitch that attains the greatest stability in a musical passage is called the *musical key* or *tonal center*. Almost all Western music is built in a way such that certain pitches functionally operate as attractor of other pitches. In music theory, this is expressed by the concept of tonality, and the reference to major and minor scales. The scale is an ordered collection of pitches capable of maintaining a consistent set of hierarchical functional relationships, the most important being related to the first pitch in the scale (called the tonic, or first degree), the fifth pitch (called the dominant, or fifth degree), and the fourth pitch (called the subdominant, or fourth degree). The tonic is the element that tends to assert its dominance and attraction over all others. In Western music theory, the tonic is one in an octave range, within the 12 semitones of the chromatic scale (i.e., `C`, `Cs/Db`, `D`, `Ds/Eb`, `E`, `F`, `Fs/Gb`, `G`, `Gs/Ab`, `A`, `As/Bb`, `B`), where `s` stands for ‘sharp’ (often denoted by $\sharp$) and raises a note by a semitone, a ‘flat’, denoted by `b` (♭), lowers it by a semitone. Remark that in the chromatic scale, a flat and a sharp of two notes at a step (i.e., two semitones) coincide, e.g., `Cs` = `Db`. Consequently, we do not make a distinction between the latter two notations.

Tonality is a harmonic system of 24 keys (i.e., 12 major and 12 minor keys). Both major and minor keys can be placed on a circle of fifths. The circle of fifths is an observation of a mathematical property of the relationship between the number of sharps or flats in a key signature and the starting note of the key. As one moves clockwise, one goes up by fifths from sector to sector. Figure 3.18 depicts the circle of fifths for major keys (outer circle) and their respective relative minor keys (inner circle). Within the remainder of this dissertation, major keys are written in capitals while minor keys are written in non-capitals. In the circle of fifths, we always have the three primary chords next to each other: the tonic or root in the center, the subdominant to the left (counter-clockwise), and the dominant to the right (clockwise). For example, for the tonic `C`, `F` is the subdominant and `G` is the dominant. Major `C` and major `Fs` are both maximally enharmonic and the relationship between a major and its minor is called the parallel, e.g., `C` is the parallel of `c`. The analysis

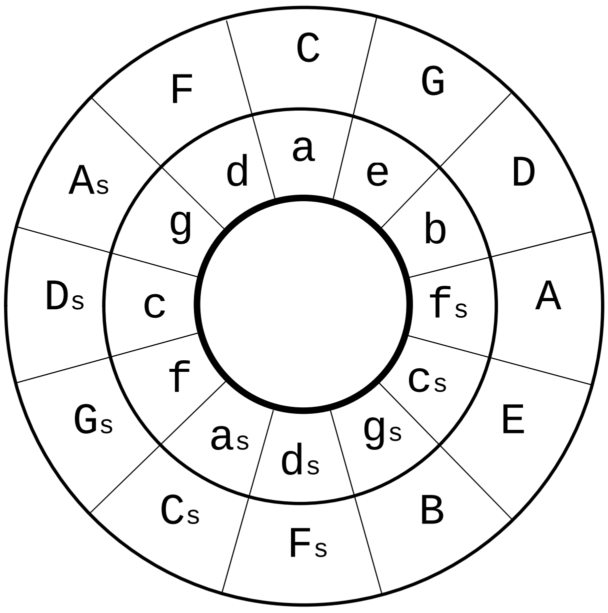


Figure 3.18: The circle of fifths for major keys and their relative minor keys.

of listener responses to an induced tonal key shows that the structure of pitch

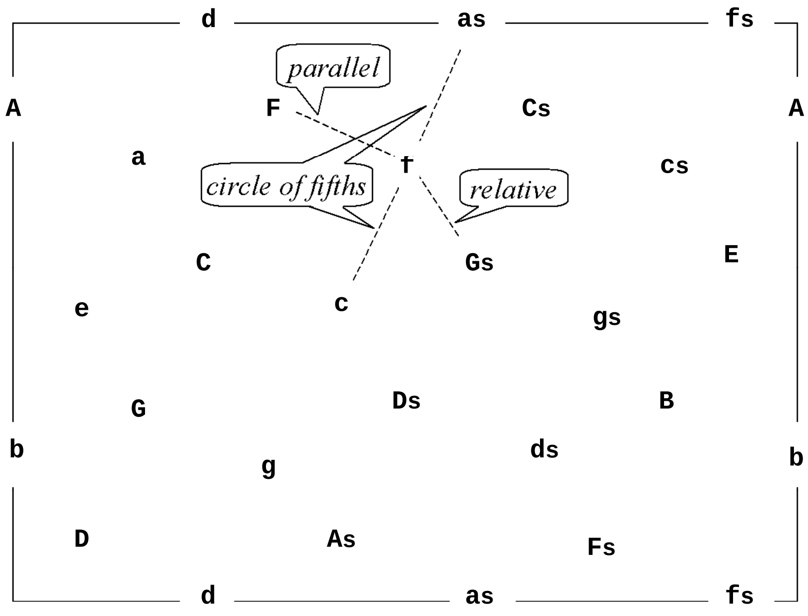


Figure 3.19: Strip of tonal centers derived from their toroidal geometric representation. Left and right borders (respectively upper and lower borders) must be identified.

relationships is low-dimensional [140]. Analysis of the key profiles of all major and minor keys, using multi-dimensional scaling, reveals a 3-dimensional structure which can be represented on a torus, as depicted in Figure 3.19. This torus embeds all relationships between minor keys and major keys as both circles of fifths fit on the torus (each circle wraps around three times before rejoining itself).

The presented research focuses on the assignment of key information from a musical audio signal. Recognition of musical key is an important step in the exploration of methods that allow the development of multi-level music representation schemes for musical content [210], such as chord recognition and also emotion detection since the musical key is deemed to provide a specific emotional connotation [208,211].

3.4.1 Related Work

In the literature on key finding algorithms, a distinction can be made between audio-based methods and score-based methods. Score-based methods start from symbolic representations while in audio-based methods the conversion to symbolic representations is a main task. Score-based methods have been founded on rule-based approaches [212–214] and template correlation or distances [207, 215–217].

Automatically estimating the musical key from an audio signal is still a challenging task. Major difficulties lie in the fact that tonality is a high-level feature and difficult to extract from audio signals based on the complexity of polyphonic musical audio analysis. Audio-based methods have been mainly based on template-related distance approaches and exist of 2 steps: a feature extraction step and a pattern matching step, e.g., [209, 218–220].

Krumhansl proposed a correlation method that compares the spectrum of a musical piece with key profiles that are derived by probe-tone rating. The key that provides the maximum correlation with the musical piece is then considered as the solution [207]. Fujishima proposed a pitch class profile based on spectral information (obtained from a DFT on the audio signal) for use in chord recognition [221]. This pitch class profile is a 12-dimensional vector which represents the intensities of the 12 semitone classes (i.e., the *chroma*) and is similar to the intensity map (a SOM trained with chroma vectors) in Leman's model [209]. Leman's audio-based model also consists of two parts: a first part that does bottom-up tonal center extraction using an auditory model, and a second part that does pattern matching with template patterns obtained by a training procedure based on SOMs. Zhu and Kankanhalli attempted to extract precise pitch profile features for key finding algorithms, considering the interference of percussive noise and pitch mistuning [222]. The latter is accomplished by a pitch tuning determination algorithm to extract the partials from the signal. A consonance filtering technique, which eliminates most peaks from percussive sounds, is then used to pick up the partials that are harmonious and relevant for tonality. Gómez and Herrera propose a harmonic pitch class profile [223] which is based on the pitch class profile of Fujishima. The harmonic pitch class profile is computed using the magnitude of the spectral peaks that are located within a certain frequency band, considered to be the most significant frequencies carrying harmonic information. Bello and Pickens combine a chroma-based representation and a hidden Markov model which is initialized with musical knowledge [224]. The output is a function of beats (instead of time) and represents the sequence of major and minor triads (i.e., three-note chords) that describe the harmonic character of the input signal. Shenoy and Wang apply a 3-step, rule-based method that combines high-level musical knowledge with low-level audio features [225]. Based on a chromagram representation, triads are detected. Based on a rule-based analysis of these chords against the chords present in the major and minor keys, the key of the song is extracted. The final step consists of chord enhancement on the basis of high-level knowledge and the key information. Their conviction is that key and chord information should be simultaneously estimated. Also Catteau et al. start from the opinion that key and chord information should be handled simultaneously [226]. They extend the framework of Bello and Pickens by proposing a probabilistic framework for simultaneous chord and tonal key recognition. By using music theory knowledge, training can be omitted.

In this section, we present a bottom-up approach to obtain the musical key from audio recordings, so no musical knowledge is applied during the decision process. We focus on two types of symbolic representations: one based on key recognition using the classical template distance model and another one based on classification trees. Amongst the variety of well-established non-distance-based supervised learning techniques, our preference has gone to the construction of a classification tree mainly because of its simplicity. Also, classification trees have already been successfully applied in a variety of musical classification tasks. In [227], classification trees have been used to obtain an automatic classification of drum sounds. Both [228] and [229] discuss the use of binary classification trees for the classification of musical instruments. Classification tree techniques have also been applied for musical genre classification [230] or content-based audio classification [231, 232].

3.4.2 Pitch Induction

As a first step, (low-level) features that are correlated to the tonal context of the musical piece need to be extracted from the audio signal. Shmulevich et al. [215, 217] derived tonal information on the basis of a difference of pitch vector (usually derived from a note sequence representation) and on relatedness ratings found by Krumhansl [207]. Here, we follow the approach of Leman [218]. An auditory model, which is an adapted version of the model of Van Immerseel and Martens [175] (the low-pass filter at 300 Hz is removed so that frequencies up to 1250 Hz are supported) is used to obtain a pitch pattern from an audio signal. Next, by integrating this information over an appropriate time interval, tonality-related information is obtained. This approach is called *pitch induction* and proceeds in three steps:

(i) Transformation of a musical signal into a cochleagram (or auditory nerve image).

(ii) Transformation of these auditory nerve patterns into pitch patterns.

(iii) Transformation of pitch patterns into induced pitch patterns.

These steps, as further described below, are implemented using the IPEM Toolbox [233].

Transformation of a Musical Signal into an Auditory Nerve Image

This process is similar to the creation of the cochleagram outlined in Section 3.3.2. Fifteen channels are used, with center frequencies ranging from 141 to 5173 Hz. The filters have a 3 dB bandwidth of one critical band[4], a low-frequency slope of 10 dB per critical band unit (cbu), and a high-frequency slope of about 20 dB per cbu. The firing rate code of the auditory nerve patterns between 80 Hz and 1250 Hz is taken as input. The 80 Hz accounts for the fact that for smaller frequencies, the sensation of pitch becomes more a sensation of textural properties. The higher limit of 1250 Hz is related to the limits of neural synchronization since for higher pitches, the neurons are not able to accurately follow the exact period of the signal.

Transformation of Nerve Patterns into Pitch Patterns

A pitch completion module computes the periodicity pitch image of an auditory nerve image. The neural rate code of the auditory nerve images is taken as input and a periodicity analysis, i.e., a pitch image, is the output. From a conceptual point of view, the pitch completion module performs a transformation from time-code to place-code, i.e., from patterns whose frequency is contained within the temporal characteristics of the pattern to patterns whose frequency is encoded along a spatial array.

For the periodicity analysis, a frame-based autocorrelation analysis is performed on the filtered channels. A frame width w of 40 ms and a step size Δt of 20 ms are chosen. For each frame at time t (t is a multiple of the step size Δt), we perform an autocorrelation analysis based on the following principle:

$$p_c(t) = \sum_{\tau=t}^{t-w} (e_c(\tau)e_c(\tau-\delta)), \tag{3.20}$$

where e_c is the firing pattern in channel c of the auditory nerve image, $\delta = \delta_0 + k\mu$ defines the delay with $\delta \in [0, w]$ and the number of delay samples

[4]For a given frequency, the critical band is the smallest surrounding band of frequencies which activate the same part of the basilar membrane.

is $k = 0, 1, ..., z$. The time lag thus increases in steps that correspond to the sampling rate of the auditory nerve images ($\mu = 0.36$ ms). Each pitch pattern can thus be represented as a vector p at a certain time step t and the components of the vector are indexed by the k value. There are $z + 1$ such values in total which define the dimensionality of the pattern. The periodicity pitch image or completion image at time t is obtained by a summation of the autocorrelation results over all channels:

$$p(t) = \sum_{c=1}^{C} p_c(t). \tag{3.21}$$

The periodicity pitch image is then attenuated according to:

$$a(\delta) = 1 - \left(\frac{\delta - \delta_0}{z} - 0.5\right)^2. \tag{3.22}$$

This attenuation reduces the impact of the short and long periods (respectively the too high and too low frequency regions) in the pitch pattern. Finally, from the attenuated pitch pattern, we select δ between 28.6 ms and $\delta_0 = 4$ ms. This gives a pitch pattern with $\left\lceil \frac{28.6-4}{0.36} \right\rceil = 69$ δ-values.

Transformation of Pitch Patterns into Induced Pitch Patterns

Successive patterns build up a content that captures pitches as they appear in their musical context and hence may induce a particular pitch pattern or key. An echoic memory module takes the pitch image as input and gives the leaky integrated image as output: the images are integrated in such a way that at each time step, the new image is calculated by adding a certain amount of the old image to the new incoming image. At each moment, a new value is obtained by adding the incoming signal to an attenuated version of the previously calculated value. The echo applied to a periodicity pitch image defines the amount of context we take into account. With long half-decay times[5], context is taken into account and the images are called global pitch images (correspond to tonality). With little or almost no context, small half-decay times are used and the images are called local pitch images (e.g., a half-decay time of 0.5 s corresponds to chords). The echo with a half-decay time T is calculated as

[5]The half-decay time specifies the time it takes for an impulse signal to reach half of its original value.

follows:

$$p_T(0) = p(0) \tag{3.23}$$

$$p_T(t) = p(t) + 2^{\frac{-1}{T}} p(t-1). \tag{3.24}$$

To obtain induced key images, we use a half-decay time of 1.5 s.

3.4.3 Creation of Key Templates

Obviously, in order to obtain key templates for all keys, chord sequences have to be generated for all keys (12 major and 12 minor) and subsequently processed. It has been shown by Leman [209] that key templates can be extracted from chord sequences that establish a strong sense of key. These chord sequences are based on the key-inducing pitches from the scale. They are typically based on the first (the tonic), fourth (the subdominant), fifth (the dominant), and again the first scale degree, both in major and minor expressed in roman numeral chords as I-IV-V-I and i-iv-V7-i, respectively. For example, in the key of `C` major the chords are thus made of the following patterns: `C-E-G`, `C-F-A`, `D-G-B` and `C-E-G` and, in the key `c` minor: `C-Ds-G`, `F-Gs-C`, `G-B-D-F`, and `C-Ds-G`. Using the above explained method of pitch induction, a key template is defined by the global pitch image at the end of such a chord sequence.

A set of 24 sequences, each sequence consists of four successive chords as explained above, is created using Shepard tones. A Shepard tone [234] is an octave-related complex sound that is a highly contrived aggregate of individual sinusoidal waves each separated by the interval of an octave. The contour of the spectral envelope of the aggregated sinusoids resembles a bell-shaped bandpass filter and the components with highest amplitude are situated in the 500-1000 Hz frequency domain, the domain where the human ear is the most sensitive for frequency perception. Chords built from Shepard tones are called Shepard chords. The octave position of a Shepard tone being undetermined, Shepard chords are utterly suited for the implementation of algorithmic harmony systems, since the position of the used chords can, in first instance, be completely left out of the system.

Next, a principal component analysis (PCA) has been carried out on the subset consisting of the 24 69-dimensional feature vectors for the Shepard sequences. In this way, the Shepard sequences are assumed to be representative audio samples that can play the role of sufficiently discriminating centers of attraction in the 69-dimensional feature space. Furthermore, it can be foreseen that

C	Cs	D	Ds	E	F	Fs	G
1	2	3	4	5	6	7	8
Gs	A	As	B	c	cs	d	ds
9	10	11	12	13	14	15	16
e	f	fs	g	gs	a	as	b
17	18	19	20	21	22	23	24

Table 3.6: Conversion from induced key notation to integer class labels.

the 24 principal component vectors are situated in a low-dimensional hyperplane. In our setting, the principal components are numbered in increasing order of variability; hence, the 69th principal component points into the direction that accounts for the largest variation. For example, the total variance of the components 66 to 69 (i.e., the four principle components with the highest variance) is 74.6% and the total variance of the five (65 to 69) and six (64 - 69) principle components with the highest variance is 78.9% and 82.5%, respectively. Further, the 24 keys are given a unique label from 1 to 24, where the class labels correspond to the common convention of the induced key, as shown in Table 3.6.

3.4.4 Key Recognition Using the Distance Model

To obtain the key of a musical excerpt, the induced pitch pattern image vectors are derived at a rate of one vector every 20 ms. We then express these vectors in the Shepard key template basis (by means of the linear transformation induced by the principal components associated to the Shepard sequences). The resulting vectors are therefore called the Shepard-normalized feature vectors.

Using the distance model, the Shepard key template which is closest to the Shepard-normalized feature vector, is chosen as the key. Residing on the low-dimensionality, we can use an n-dimensional subset of the 69-dimensional vectors, $n < 69$. For each of the thus obtained n-dimensional vectors, we compute in the n-dimensional subspace the Euclidian distance to the Shepard key templates and determine the nearest of these key templates.

However, we experience that, despite the low-dimensionality, for $n = 4$, 5, or 6 the key assignments using the distance model are different. This finding is demonstrated in the next examples.

In the first application, we take a 7 seconds sample (between 1'53 and 2'00) of

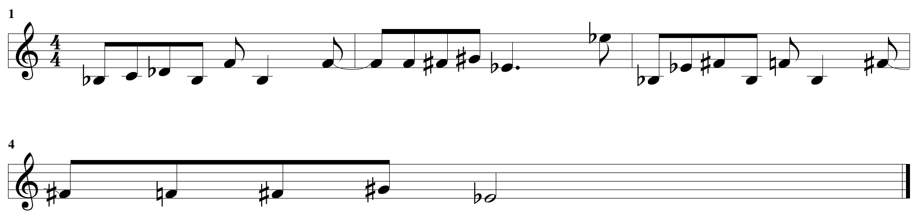


Figure 3.20: Score of the prominent melody line of "Eternally" by Quadran.

the dance song "Eternally" by Quadran and compute the Shepard-normalized feature vectors for every 20 ms. The (prominent) melody line of this excerpt consists of the notes: `As4 C5 Cs5 As4 F5 As4 F5 F5 Fs5 Gs5 Ds5 Ds6 As4 Ds5 Fs5 As4 F5 As4 Fs5 F5 Fs5 Gs5 Ds5` as depicted in Figure 3.20. By considering the key signatures for major and minor scales, the four most probable keys are: `as`, `ds` and their relative major key neighbors `Cs` and `Fs`.

Retaining the components 66 to 69, we obtain a sequence of keys as shown in Figure 3.21. On the vertical axis, the tonal centers are marked by a number while to the right of the figure the corresponding key notations are shown, according to Table 3.6. The key assignment is almost constantly `as`, and only for $1/3$ s equal to `f`. The variation at the beginning of the sequence must be ignored, since we do not want to take into account deviations that could be ascribed to particular starting conditions of the pitch pattern detection process. In Figure 3.22, we show the key recognition for the same musical excerpt, using a 5-dimensional subspace (spanned by components 65 to 69 of the Shepard basis). It is striking that this assignment sequence is completely different from the one established in the 4-dimensional subspace, as one notices that now the assignment varies in time from `Ds` over `f` to `ds`.

This apparent instability in key recognition when the dimension of the projection subspace is augmented (or more generally, altered), is again confirmed by comparing the results obtained in the 6-dimensional subspace with the previous ones, as shown in Figure 3.23.

At first sight, one may be inclined to conclude that the Euclidean distance method is unreliable when used in combination with the dimensionality reduction. Nevertheless, psycho-acoustic experiments by Krumhansl and Kessler [140], as well as computational methods using neural networks [209] and brain imaging experiments [208] show that the geometric representation of the inter-key relations is low-dimensional [207,217]. The geometrical representation as

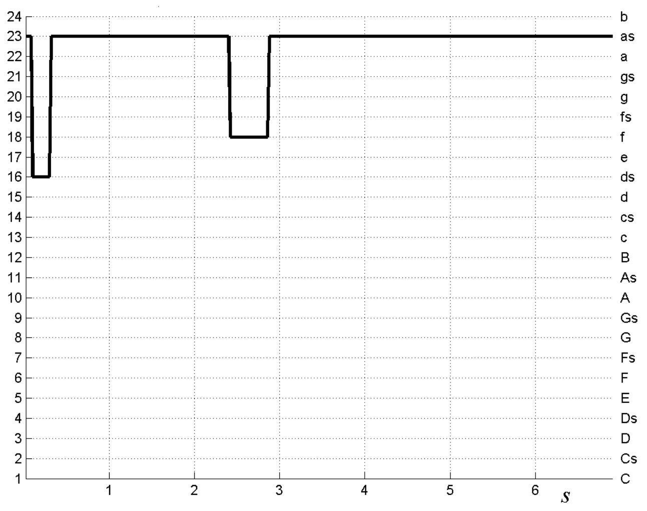


Figure 3.21: Sequence of nearest Shepard key templates in the 4-dimensional Euclidean subspace (components labeled 66 to 69 in the Shepard basis) for the excerpt of "Eternally" by Quadran.

depicted in Figure 3.19 suggests, for instance, that the variation between the keys `as` and `f`, observed in Figures 3.21 and 3.23, may be somehow situated in the middle of key `as` and `f`, being close neighbors in this representation. The same geometrical argument can be invoked for the key assignment variation in Figure 3.22 because `ds` and `Ds` are also close neighbors, i.e., parallel neighbors. Hence, in this example, tonal variation can be explained as a positioning between neighboring tonal centers, although the distance-based assignment rule does not offer the possibility to select the second nearest tonal center for key assignment in order to produce more stable key assignment sequences.

The assignment sequences in the 4- and 6-dimensional spaces show a great resemblance, whereas the assignment sequence in the 5-dimensional space is somewhat different. One possibility to reduce these differences in the key assignments, is to agglomerate the keys in larger groups of keys or key-clusters. Since theoretical acoustical considerations do not provide a straightforward way of agglomerating tonal centers in larger groups, we need to devise such a clustering by means of the systematic comparison of the assignment sequences for a representative set of musical extracts, respectively generated in 4, 5, 6, or higher dimensions.

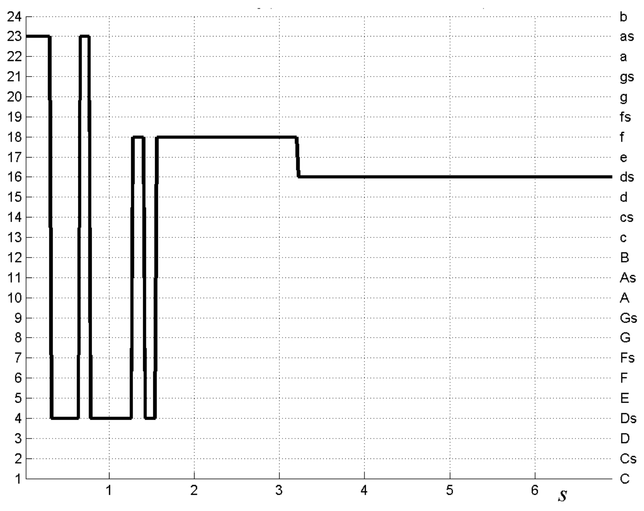


Figure 3.22: Key assignment sequence based on Euclidean distance in the 5-dimensional subspace for the same excerpt as in Figure 3.21.

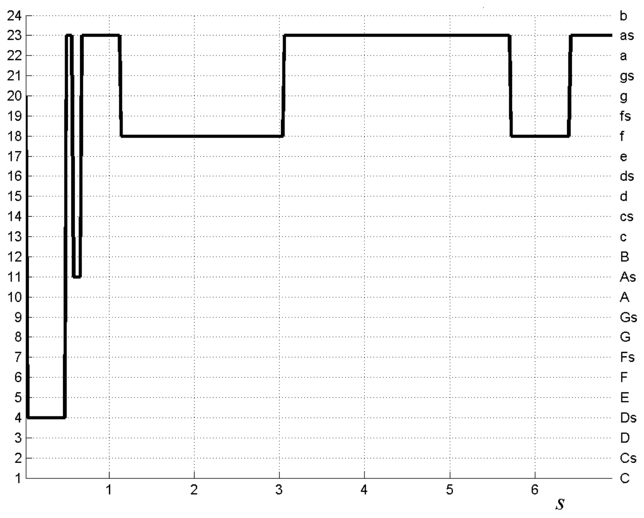


Figure 3.23: Key assignment sequence based on Euclidean distance in the 6-dimensional subspace for the same excerpt as in Figure 3.21.

3.4.5 Classification Tree for Key Recognition

As a way to overcome the drawbacks described in the previous section, we have developed a new method of key recognition. There is an abundance of classification methods that fall in the category of supervised learning techniques, such as different types of neural networks, instance-based learning algorithms, and tree-based approaches. However, tree classification techniques (see Appendix E) have a number of advantages. First, the interpretation of the results summarized in a tree is very simple. This is not only useful for purposes of rapid classification of new observations, but it can also yield a much simpler model for explaining why observations are classified or predicted in a particular manner. Secondly, tree-based methods are non-parametric and non-linear. There is no implicit assumption that the underlying relationships between the predictor variables and the dependent variable are linear, follow some specific non-linear function, or are monotonic in nature. Thus, tree-based methods are particularly well suited for data mining tasks where little a priori knowledge is available.

Classification trees are constructed top-down, beginning at the top node with the most informative feature, i.e., the one that maximally reduces entropy. Branches are then created for all values of the descriptor of this node. The training examples are sorted to the appropriate descendant node and the process is repeated recursively. Each node is connected to a possible set of answers and each branch carries a particular test results subset to another node. The eventual rules for classification can be summarized in a series of logical if-then conditions and each terminal node is associated with a single class. The building process of a tree is a recursive procedure, but it is still faster than the training of a neural network.

Besides the 24 Shepard sequences, we have generated 10 more sets of 24 chords sequences, each consisting again of 4 chords and representing one of the 24 major and minor keys. These 240 additional sequences were sampled from professional synthesizers[6] and cover a wide spectrum of instrumental sounds, such as brass sound, piano, strings, acoustic guitar, etc. We then converted the feature vectors of the training sequences into key templates, represented as normalized feature vectors divided over 24 classes (10 new vectors per class plus one vector associated to the Shepard sequence) respectively labeled from 1 to 24 according to the conversion rules of Table 3.6. This whole set of 264 key templates is used as training set for growing a classification tree. Once established, the classification tree can be used to predict membership of

[6]Yamaha © QS300 and Waldorf © Micro Q

newly induced pitch patterns in one of the 24 classes [235].

The construction of a classification tree has been done by means of the CART 4.0 software package [236]. In CART there are a lot of parameter settings and option selections, and for most of them we have chosen the recommended default values. In particular, priors have been taken proportional to the class sizes and misclassification costs have been taken equal for every class. The right-sized tree has been selected on account of the 10-fold cross-validation technique. Furthermore, we have chosen the discriminant-based univariate splits option since linear combination splits are clearly irrelevant for the normalized data. Indeed, the PCA-normalization pre-processing of the data implies that optimal linear combinations of all vector components have already been taken into account. Finally, we have done experiments with the two most frequently used splitting rules: the Twoing rule and the Gini rule [235] (also see Appendix E). Both methods yield classification trees of comparable quality (where the used quality criterion, called the relative cost, takes into account both the misclassification score and the complexity of the obtained tree). Here, we will mention only the results obtained using the Twoing criterion, as it yields a tree that is much more equally balanced than the tree obtained with the Gini criterion. This observation can be attributed to the fact that the Twoing criterion seeks for splits that are roughly equal in size and Twoing is better for multi-class dependent variables than Gini.

The smallest subset of components giving rise to a perfect classification tree, in the sense that none of the learning examples is misclassified, is the set of the four components label ed 66 to 69. In Figure 3.24, the obtained classification tree is plotted. Notice that this tree contains 30 leaf nodes (in general, a perfect tree can possess more leaf nodes than classes). It should be mentioned that a perfect and well-balanced classification tree with exactly 24 leaf nodes and with univariate splits on the components 66-69 can be constructed by solely using the Shepard key-templates. However, it appears that, due to the restricted training set consisting of only Shepard tones, the applicability of the obtained classification tree is quite limited since most musical instruments profoundly differ from Shepard tones [14]. On the other hand, one should not focus too much on considering only classification trees based on a minimal number of components (or dimensions) on which to perform splits.

We found, for example, a perfect classification tree with 30 leaf nodes and with univariate splits on the components 65-69 (5-dimensional space) that is topologically equivalent to the classification tree of Figure 3.24, and a perfect classification tree with 28 leaf nodes with univariate splits on the components 64-69 (6-dimensional space) that is topologically almost equivalent to the clas-

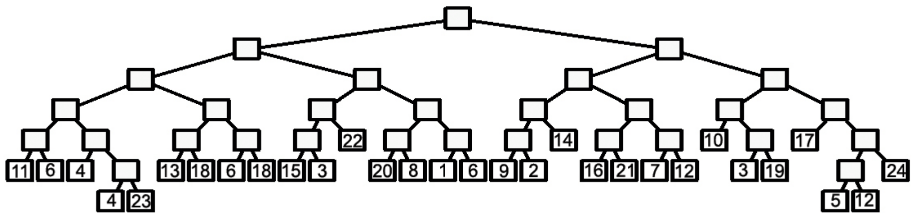


Figure 3.24: A well-balanced perfect classification tree for the classification of Shepard-normalized feature vectors into 24 classes, with univariate splits on the components 66-69.

sification tree of Figure 3.24. It must be emphasized that topological equivalence or nearly topological equivalence not necessarily implies that the classification is the same or almost the same. Indeed, classification does not just depend on the topology of the classification tree, but more dominantly on the splitting conditions which for topologically equivalent trees may be completely different. We did not directly compare the splitting conditions at corresponding nodes in the classification trees for 4, 5, and 6 dimensions, respectively, but preferred to compare the classification results or key assignments on a variety of musical data.

In testing this method, we reconsidered the musical excerpt "Eternally" of Quadran. Each of the three classification trees accounts for a key recognition method that, in contrast to distance-based recognition, will be called tree-based recognition. Note however that the tree-based recognition is expected to be more reliable than the distance-based recognition, because the latter only uses information from the subset of Shepard sequences.

Figure 3.25 shows key recognition with the classification tree of Figure 3.24 using splits on 4 components. As in the case of distance-based recognition, the key does not remain constant over the whole excerpt. The key varies between the keys `as` and `Ds`, which are not direct neighbors in the toroidal geometric representation, but are also not separated too far from each other.

Tree-recognition using splits on 5 components results in Figure 3.26. Again, the recognition varies between the same two keys, i.e., `as` and `Ds`, but the time of transition clearly does not match.

Finally, using splits on the last 6 components, the result of Figure 3.27 is obtained. Again a variation between the same two keys is observed. The alternation between the keys `as` and `Ds` is conform to most probable keys using key signatures. This alternating between two keys is significant as this behavior

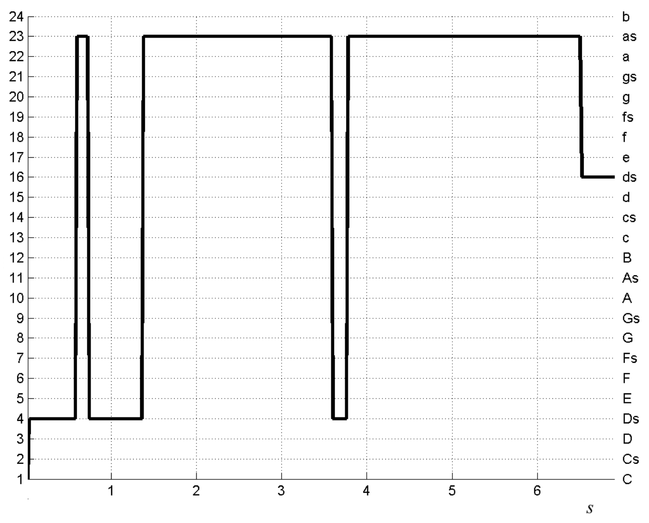


Figure 3.25: Tree-based key assignment sequence with splits on 4 components for the same musical excerpt as in Figure 3.21.

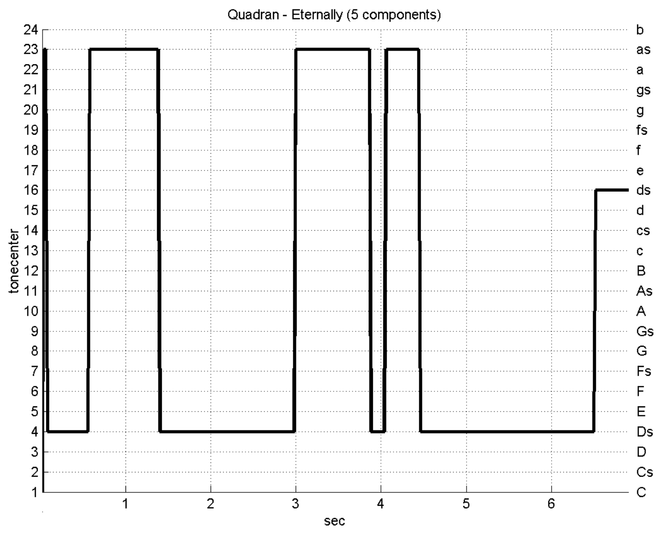


Figure 3.26: Tree-based key assignment sequence with splits on 5 components for the same musical excerpt as in Figure 3.21

occurs in more experiments. Therefore, we can already postulate the preliminary conclusion that for a fixed dimension, the key assignment with a tree-based method is more stable than the key assignment with the distance-based method. The tree-based recognition, however, produces results that seem to be more robust and invariant with respect to the dimension of the tree (i.e., the number of components on which splits are performed). This also implies that the classification trees in 4, 5 or 6 dimensions, respectively, are not only topologically similar, but that they also possess splitting conditions that do not significantly differ from each other in corresponding nodes. Thus, the final classification is almost the same with each of these trees.

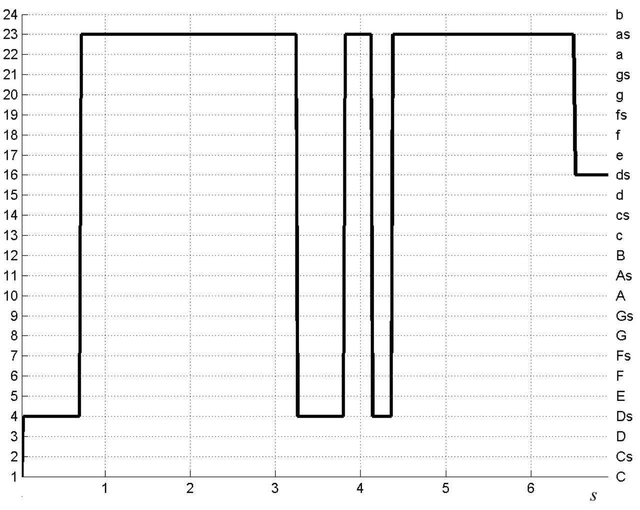


Figure 3.27: Tree-based key assignment sequence with splits on 6 components for the same musical excerpt as in Figure 3.21.

3.4.6 Clustering Keys into Key Classes

The property of independence with respect to the number of components withheld motivates us to proceed with the tree-based key recognition methods. In particular, we shall consider the lowest-dimensional classification tree of Figure 3.24 with splits on 4 components. The question arises whether we could reduce the variability in key assignment at least for short musical excerpts of only a few seconds long. Usually, such a task is carried out by filtering techniques that take into account tendencies over longer periods and try to use that knowledge to eliminate short term variations as much as possible. However, the classification tree structure on its own provides a way of agglomerating classes into larger groups and it can be expected that a coarser grain classification will automatically lead to fewer variations. Especially the well-balanced trees that are obtained by means of the Twoing splitting rule may be considered for the formation of clusters that more or less group together a same number of classes. Taking into account the binary structure of the classification tree, it is clear that a reduction is possible from 24 classes into 16, 8, 4, or 2 clusters of classes. It should also be mentioned that the agglomeration of the 24 keys into 16 (or less) key groups or key clusters, appears to be the same when either the 4-dimensional, the 5-dimensional, or the 6-dimensional classification tree is used. It does not mean, however, that the different trees will necessarily classify a given feature vector in the same cluster. With the help of the conversion rules in Table 3.6 one can verify from Figure 3.24 that one obtains, for instance, the 8 clusters of musical keys as depicted in Table 3.7.

$C_1 = \{as, As, Ds, F(1)\}$	$C_2 = \{c, f, F(8)\}$
$C_3 = \{a, d, D(9)\}$	$C_4 = \{C, g, G, F(2)\}$
$C_5 = \{cs, Cs, Gs\}$	$C_6 = \{ds, gs, B(10), Fs\}$
$C_7 = \{fs, A, D(2)\}$	$C_8 = \{b, e, E, B(1)\}$

Table 3.7: 8 clusters of musical keys obtained from the classification tree of Figure 3.24.

The numbers in parenthesis in Table 3.7 indicate how many of the 11 training sequences associated to each key are present in that cluster. For example, of the 11 examples of key `D`, 9 belong to cluster C_3 while 2 belong to cluster C_7. If we trace these clusters in the torus (see Figure 3.19), we readily see that each of them is an assembly of keys that are in geometrical sense close to each other. The numbering of these clusters has been attributed by a left-to-right scanning of the classification tree and this implies that clusters with consecutive numbers are usually also close to each other in Figure 3.19. For example, `D` is close to `a` and `d`, as well as to `fs` and `A`, which corresponds to the clusters C_3 and C_7 over which the examples of key `D` have been distributed.

Let us reconsider the example of "Eternally" by Quadran. The cluster assignment sequences obtained by means of the classification trees in 4, 5, or 6 dimensions, respectively, are identical and yield the fixed assignment of cluster C_1 to that musical excerpt, as shown in Figure 3.28. The property that the three classification trees produce the same cluster sequences has been ob-

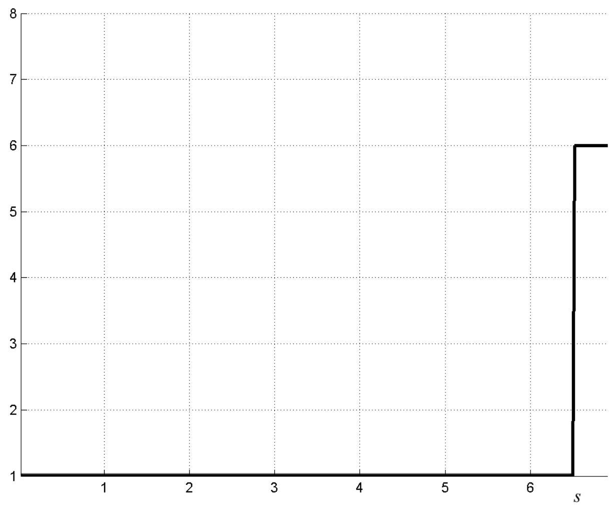


Figure 3.28: Cluster assignment sequence obtained with any one of the classification trees discussed for the same musical excerpt as in Figure 3.21.

served in many other examples. Therefore, in the remainder, we will only be concerned with results generated by the lowest-dimensional classification tree (i.e., with splits on 4 components).

For example, by considering an excerpt of J.S. Bach's "Inventions Nr. 1 in C Major", a piece of (classical) music written in C major, we see that the tree-based key recognition is mainly characterized by the assignment to the key `C` (with some small fluctuation to `F`, a neighbor on the circle of fifths) as depicted in Figure 3.29. The assignment of key clusters also reveals a constant assignment to cluster C_4 as shown in Figure 3.30. It is worthwhile mentioning that only two of the 11 training sequences associated with key `F` fall in cluster C_4. Nonetheless, the `F`-key assignment for this piece of music is precisely of this rather exceptional type, so that one cluster assignment is found.

To illustrate that it is not a rule that the cluster number is constant over a longer piece of music, we present the results obtained for an excerpt of the song "Blowing in the wind" by Bob Dylan. It must be emphasized that our classification tree has not been trained on human voices, and therefore we do not expect in advance a stable key assignment, nor a stable cluster assignment. Indeed, from Figures 3.31 and Figures 3.32 we see that the cluster assignment varies between the three clusters C_4, C_7 and C_8.

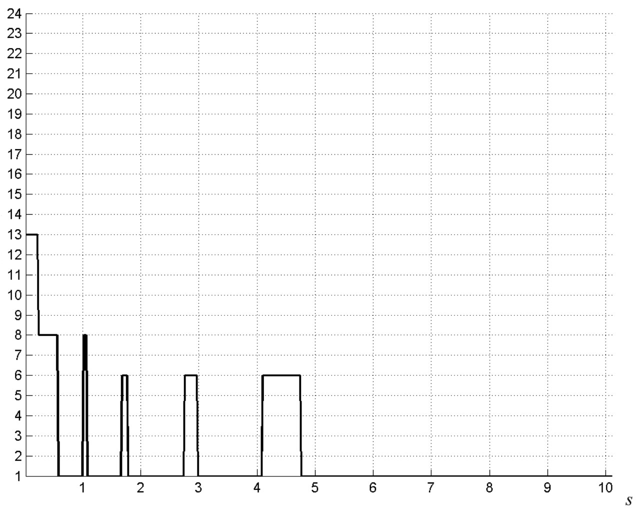


Figure 3.29: Key assignment sequence for J.S. Bachs “Inventions Nr. 1 in C Major”obtained from the classification tree of Figure 3.24.

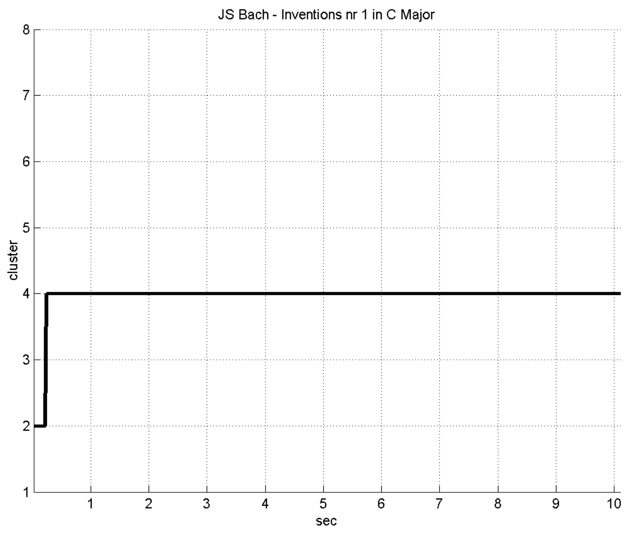


Figure 3.30: Cluster assignment sequence for J.S. Bach’s “Inventions Nr. 1 in C Major”.

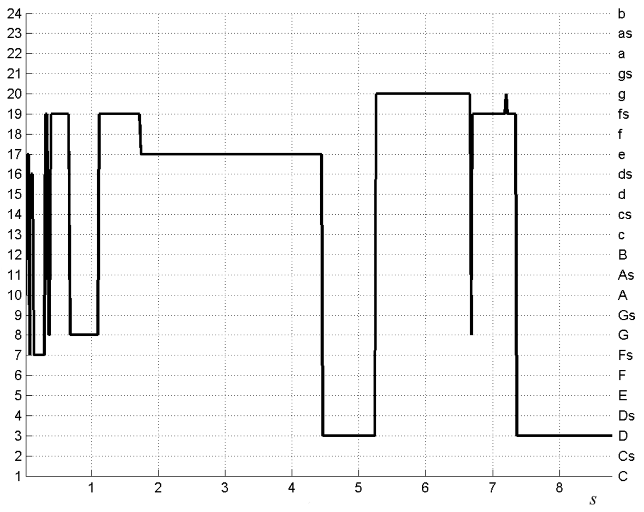


Figure 3.31: Key assignment sequence for Bob Dylan's "Blowing in the Wind".

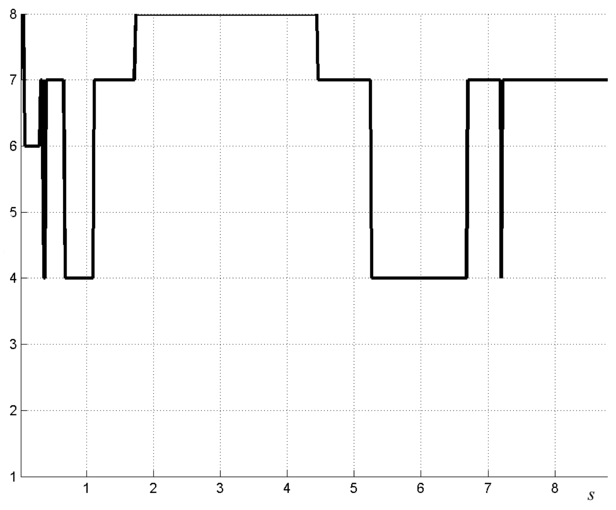


Figure 3.32: Cluster assignment sequence for Bob Dylan's "Blowing in theWind".

3.5 Conclusions and Future Work

In this chapter, we have dealt with the extraction of pitch-related information from musical audio signals. Two types of pitch information are considered: melody and tonality.

At first, we presented a transcription system to automatically extract the melody from both monophonic and polyphonic musical audio recordings. Unlike many melody extraction approaches, we aimed to distinguish individual musical notes characterized by specific temporal boundaries and a frequency resolution of a half semitone. Our melody transcription system consists of four stages. In the first stage, we used the auditory model of Slaney and Lyon to obtain the necessary pitch information for the later processing stages. In the second stage we estimated the F0 in individual time frames, and we created pitch trajectories over adjacent time frames. In the third stage, the obtained trajectories are transformed into tones so that a note representation can be easily deduced. Finally, in the fourth and last stage, a post-processing which removes low-salience tones or too short tones, is applied to obtain the melody. We evaluated this system using two melody sets: (i) the GMEL set, a self-created set of 34 excerpts from popular music with their respective MIDI representation and (ii) the MIREX training set consisting of 13 F0-annotated musical audio files. For the GMEL set, the average precision and recall for raw pitch detection are 0.71 and 0.86 while for chroma detection,0.72 and 0.87, respectively. For the MIREX 2005 training set, the average precision and recall for raw pitch detection are 0.68 and 0.65, respectively while for detection the chroma information the average precision and recall are 0.81 and 0.79, respectively. These results are very promising. However, there is still room for improvement. Especially the quantization of frequencies related to sung melodies could be improved by detecting vocal and non-vocal regions in the musical signal and analyzing the fragment accordingly. The pitch fluctuation of a sung melody (and other parts with vibrato) give rise to segmentation errors as we aim to obtain a note representation. On the other hand, one might ask if a note representation is a good representation (both as ground truth and transcription) to be used for sung musical fragments since the produced frequencies actually deviate from the intended pitch. Next, a basic assumption of the presented system is that the melody line is salient in polyphonic mixes. In songs where the melody is not so salient, other instruments and percussive sounds may cause peak masking, which results in a more difficult estimation of the F0. Psychoacoustic findings about the perception of melody (see Section 3.2.3) can therefore be helpful in the creation of pitch trajectories. Further, in the post-processing phase, information about the musical key can be embedded so that in the tone creation phase, the most likely notes, given a determined key, can be favored. Consequently, quantization errors can be reduced.

Next to the extraction of melodic information, we also considered the extraction and classification of tonality information. Like melody, tonality also entails pitch information, but it manifests at a relatively larger time interval than melody. The presented approach is based on pitch induction as proposed by Leman. Tonality is low-dimensional which can be deduced from the fact that by applying a PCA on a 69-dimensional space, the 4 biggest principal components account for 75% of the total variance. However, when this low-dimensionality is exploited, the widely used metric-based approach is unstable as different tonal centers are returned in a 4-, 5-, and 6-dimensional space. The presented tree-based approach overcomes this shortcoming. Furthermore, the tree-based method offers a way of regrouping keys into larger key clusters by ascending one (or more) level(s) in the tree. An additional advantage of the tree-based method is that the set of training data can always be enlarged with more specific sounds so that specialized trees can be grown for different types of music whereas the metric-based approach is based on a fixed data set (preferably Shepard tones) from which the tonality information is extracted. However, the latter can also be seen as a shortcoming of the presented approach since it requires the creation of annotated training data for the supervised classification of it. This information can be exploited in order to significantly reduce the musical search space when a query is launched in a content-based music retrieval system. Also, we think that it must be possible to find correlations between these sequences and other musical features at a higher conceptual level (e.g., emotional connotations). As to the results, the fact that the labeling of the tonal sequences is absolute, is a fundamental weakness of the application domain rather than the weakness of the method. Indeed, a musical piece played at a higher pitch may yield a completely different tonal assignment. A straightforward extension, therefore, is to consider pitch interval patterns instead of (absolute) pitch patterns or to estimate the pitch mistuning as later performed by Zhu and Kankanhalli [222].

The work that was presented in this chapter is published in the journal Soft Computing:

(i) G. Martens, H. De Meyer, B. De Baets, M. Leman, M. Lesaffre, and J.P. Martens. Tree-based versus distance-based key recognition in musical audio. *Soft Computing*, 9(8):565–574, 2005

Additionally, work in this domain can be found in the following publications:

(i) G. Martens, H. De Meyer, B. De Baets, M. Leman, J.P. Martens, L. Clarisse, and M. Lesaffre. A tonality-oriented symbolic representation of musical audio generated by classiffication trees. In *The EUROFUSE Workshop on Information Systems*, 2002

(ii) M. Leman, L. Clarisse, B. De Baets, H. De Meyer, M. Lesaffre, G. Martens, J.P. Martens, and D. Van Steelant. Tendencies, perspectives, and opportunities of musical audio-mining. In A Calvo-Manzano, A Perez-Lopez, and J Salvador Santiago, editors, *Revista de Acustica*, volume 33, 2002

(iii) M. Lesaffre, D. Moelants, M. Leman, B. De Baets, H. De Meyer, G. Martens, and J.P. Martens. User behavior in the spontaneous reproduction of musical pieces by vocal query. In *Proceedings 5th Triennial ESCOM Conference*, pages 208–211, 2003

(iv) M. Lesaffre, K. Tanghe, G. Martens, D. Moelants, M. Leman, B. De Baets, H. De Meyer, and J.P. Martens. The MAMI Query-By-Voice Experiment: Collecting and annotating vocal queries for music information retrieval. In *Proceedings of the International Conference on Music Information Retrieval*, pages 65–71, 2003

Chapter 4

Content Management and Semantic Metadata

Talk to a man about himself and he will listen for hours.
– Benjamin Disraeli (1804 – 1881)

4.1 Introduction

In the last decades, the digitization of content has experienced an enormous growth in both the number of users and the range of applications. Furthermore, the amount of digital information is still drastically increasing, a trend that is going to continue in the next years according to the IDC [1]. This increase over the last decades is caused by the fact that end-users now are also content creators besides content consumers. Indeed, the current technology makes creating multimedia content, such as videos or images, an easy and everyday task. This growth in digital technology has created the desire amongst users to manage and exchange their content in a variety of ways. Extra hardware (such as external hard disks, media players, and photo frames) and on-line file servers are used for storage, but posting content on the World Wide Web (e.g., publishing content on social networks) is also extremely popular. As a consequence, one's personal content is stored on many devices, and so it becomes more difficult to efficiently manage these data. Furthermore, this huge amount of information which is generated daily, is mostly stored in unstructured repositories, much of which can be accessed through the Internet. As one wants that as much information as possible about the content is provided, the need for appending and managing metadata, i.e., data about the data, is even bigger. Thus, metadata are not only necessary to describe the content itself, but also the management of this vast amount of content requires metadata.

Concerning the description of the content, the metadata can either be added manually or automatically be extracted from the content itself. When the metadata are manually supplied, we talk about *annotations*. Annotation is a process in which a human (expert) generates metadata for some content. Generically, an annotation can cover any type of metadata, such as the title and performer of a song, the date a file was created, the name of a person in a photograph, etc. Metadata that are automatically obtained from the content concern the extraction of *features*, possibly followed by some decision process. Examples of automatically obtained metadata are texture features, color histograms, an automatic labeling of an image region, a melody transcription, etc.

Despite its great value, multimedia metadata have yet to find its way into standard use [237]. In fact, many people use sites, such as Flickr[1] or YouTube[2], to publish their content, and in order to manage their collections to some extent, they typically assign keywords to their content. However, the annotation of content using keywords implies some issues. Since a keyword is separated from its context, it loses a lot of its intended meaning. For example, the keyword "java" can refer to a programming language, but also to an island of Indonesia, or even to the coffee produced on that island. For the human user, the context in which a keyword is used, explains its semantic meaning. However, for a machine, the context is often difficult to obtain. Furthermore, other difficulties show up: different words are used to refer to the same concept, spelling errors occur, words can have multiple meanings, slang is used, etc. In other words, relying on keywords for describing the semantics is not trivial. To tackle this problem, the use of metadata models is favored as they define fields and values for specific metadata types. A metadata model describes how data are represented and formally defines the elements and relationships between the elements of the model.

In order to efficiently manage the content and the metadata, a content management system (CMS) is needed. Besides a set of procedures to manage the work flow, a CMS typically involves a metadata model. Within the context of a single CMS, the semantics of different fields of the metadata model are well-defined. However, there is a major discrepancy when it comes to the interoperability between different metadata models. Especially in networked systems where many different parties need to cooperate and share information

[1]http://www.flickr.com
[2]http://www.youtube.com

resources, semantic interoperability is a crucial problem. In a general context, there are a set of independent data sources of a common domain and we need to share and exchange information among them. Each data source can be represented by its proper metadata model that uses a certain vocabulary with specific semantics. Thus, an appropriate mediation system is needed for allowing the interoperability of different data sources. This mediation system must provide a means to overcome the semantic heterogeneity between all data sources and also a means to access the data with transparency as much as possible.

The work in this chapter represents our activities in the IBBT – PeCMan (Personal Content Management) project[3] and our collaboration in the W3C Multimedia Semantics Incubator Group[4]. We describe a CMS for the management and disclosure of personal content in a community context. Here, a community context indicates that there are different end-users of the system. These end-users can, if desired, have access to both the content and the metadata of other end-users. Further, we deal with the interoperability issues between different metadata standards. This interoperability problem was the main topic of the W3C Multimedia Semantics Incubator Group in which we have actively participated within the photo use case[5]. The photo use case deals with the problems that arise for retrieving personal photos that are stored without any metadata. Accordingly, the photo use case will be used in this chapter as we discuss the creation of a content-descriptive metadata model. In traditional CMSs, data are typically stored at one central location, but due to the increasing number of (high-capacity) storage devices and file servers on the Internet, it becomes more difficult to keep track of one's personal content. Therefore, the proposed CMS will provide a solution for the management of this kind of data. At first, we describe in Section 4.2 the requirements of the metadata model and the CMS by use cases. Next in Section 4.3, we give an overview of related work about metadata modeling languages, image metadata standards, and open-source CMSs. Then in Section 4.4, we describe our developed CMS and elaborate on its metadata model which relies on Semantic Web technologies. Furthermore, the presented CMS metadata model embeds a model for image annotation. As such, we provide a solution for the interoperability problem between (image) metadata standards. Finally, in Section 4.5 conclusions are drawn.

[3]http://www.ibbt.be/en/project/pecman

[4]http://www.w3.org/2005/Incubator/mmsem

[5]http://www.w3.org/2005/Incubator/mmsem/XGR-image-annotation-20070814/

4.2 System Requirements

In order to reveal the requirements on the metadata model and the CMS, multiple use cases were set up. These use cases covered a range of different types of content, terminals, and services. The following use cases were considered:

(i) *content disclosure*,

(ii) *content relocation*,

(iii) *annotation of content*,

(iv) *content browsing*,

(v) *security*,

(vi) *interoperability*.

4.2.1 Content Disclosure

The actual file (here further referred to as *document*, such as an image, audio, video, or text file) can be stored on several possible devices that can be connected to the World Wide Web (e.g., a PC, a cell phone, or a storage provider). The disclosure of a document denotes its registration by the system so that it is also available for the end-users. Furthermore, the metadata of each document should also be managed so that the metadata are always available, even if a document is unavailable. For example, a document that is stored on a mobile device can become unavailable when the device's batteries run flat. However, this unavailability is only temporal and meanwhile queries must be able to rely on the document's metadata (e.g., to retrieve all documents of a given end-user or to impose restrictions upon the content).

4.2.2 Content Relocation

Since a document can be stored on a random device, its storage location should not impose a constraint upon its disclosure. So, it must be possible to move a document to an arbitrary device while it remains available for the system and possibly other end-users. Therefore, three storage modes are possible: (i) "off-line", (ii) "on-line", and (iii) "always on-line" storage. Off-line storage means that the document is not accessible once the device is switched off (e.g., a document stored on a laptop will be automatically set to "off-line" as soon as the laptop is disconnected from the World Wide Web and it will be set "on-line" when the laptop is reconnected). To make a document accessible at any time (i.e., "always on-line"), even when the storage device is switched off, the system can hold a copy of the document in its cache so that it is always available.

4.2.3 Content Annotation

The purpose of content annotation is to provide a description of the content. This information is relevant for classification and retrieval operations. Examples of queries are: "find all pictures containing flowers" and "find all songs with a similar melody". The description of the content can thus be an annotation, a set of features, or a combination of both. Consequently, metadata fields are necessary to allow annotation. Both a predefined set of keywords from a controlled vocabulary (e.g., a taxonomy) and the possibility to freely add keywords are necessary. For the digital photo use case, basic image parameters (such as file name, dimension, and coding format), creational and content-descriptive metadata parameters are mandatory. Further, the following content-descriptive concepts have been identified in the PeCMan project for photo annotation:

(i) depicted object, person, group, and event,

(ii) the position within the image of the latter,

(iii) the time and location,

(iv) the role of the participants in an event.

4.2.4 Content Browsing

As a personal collection of documents permanently grows, it will be harder for the end-user to find specific content as time passes. Therefore, different strategies can be employed to retrieve documents. To query a collection, three broad strategies can be distinguished:

(i) *Search by association.* The intention of the user is to browse through a large collection without a specific aim. Search by association tries to find "interesting" documents. This search method is already implemented in some state-of-the-art CBIR systems, such as PicSOM [115], Photobook [238], and Blobworld [239], to find images based on visual similarity.

(ii) *Target search.* The purpose of this search mode is to find a specific document. For images, the search may be for a precise copy of the image in mind (e.g., when searching in an art catalogue) or for the same object the user has already an image of (e.g., stamps, paintings, and catalogues in general).

(iii) *Category search.* This search mode aims at retrieving an arbitrary document for a specified class. Categories may be derived from annotations or may emerge from the data set (e.g., "find all outdoor images" or "find all photos depicting one or more family members").

4.2.5 Security and Interoperability

Security rules are necessary to limit the access to both the content and metadata. It might be desirable (because of copyright reasons) to grant access to the annotations, but to hide the content itself. For example, a musical album can not be shared because of copyright issues, but its metadata can be shared.

Further, there should be interoperability with metadata standards so that exchanging information using different formats is possible.

4.3 Related Work

At first, we explain the use of Semantic Web Languages in Section 4.3.1 for metadata modeling. Then, we give an overview of image metadata standards in Section 4.3.2 and open-source CMSs in Section 4.3.3.

4.3.1 Metadata Modeling

For the modeling of metadata standards, the Extensible Markup Language (XML) has been widely used over the last decade [3]. XML is a tag-based metalanguage (i.e., a language to define languages) used to define domain-specific grammars because it allows the generation of tags that specify the structure and syntax of metadata. As such, the metadata are machine-parsable and -readable. Though, the use of XML neither guarantees a correct semantic interpretation. This is because XML is rather a structuring language and it does not allow to explicitly define the semantics of the concepts that are described. XML tag names are simply character strings that may be meaningful to a human reader but are meaningless for software. For example, the

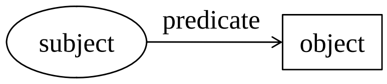


Figure 4.1: A graphical representation of an RDF statement.

tags `<father>` and `<child>` may be ascribed with semantics by a human reader, but not by software. Therefore, the meaning of tags must b e agreed upon a priori to support interoperability. Traditional metadata standards that rely on XML, supply an (often voluminous) textual description of the meaning of the different XML tags. This knowledge has then to be incorporated into software in order to correctly interpret this XML.

A way to overcome these problems is the use of Semantic Web technology. Within the context of this dissertation, when Semantic Web technology is used to create a metadata model, this model is called an *ontology* (i.e., domain knowledge). Ontologies enable the machine interpretation of and reasoning about semantics. A first metalanguage to create ontologies is the Resource Description Framework (RDF) which is an approved W3C recommendation for asserting values of properties associated with web resources [240]. Rather than displaying information, RDF is intended to be used by applications to process information. RDF uses the XML standard for its syntax and is a language for representing information about *resources*. In this context, a resource is anything that can be associated with a Uniform Resource Identifier reference (URIref) [240]. So, resources include both physical (e.g., a book or a menu) and virtual (e.g., web page, images, or services) objects. A special type of resource is a *property* that serves to describe attributes of resources and binary relationships between resources. Besides resources, the RDF data model includes *literals* and *statements*. As such, literals are actually text strings with optional language and data identifiers. Literals are possible property values in a statement. Statements are attribute-value pairs formed by properties with associated values. RDF statements consist of a subject resource (i.e., the resource that is being described), a predicate (i.e., the property), and an object (i.e., a literal or a resource value) as graphically represented in Figure 4.1.

Since RDF lacks the description of more complex semantic relationships (e.g., classes) nor provides mechanisms for describing properties and concepts for enumeration and custom data types, RDF Schema (RDFS) was introduced [241]. RDFS defines classes and properties that may be used to describe classes, properties, and other resources. RDFS builds on the RDF foundation to provide additional descriptive features and a language for describing the expanded vocabulary. However, RDFS still does not provide sufficient expressiveness to create meaningful ontologies (e.g., RDFS places no restrictions on property cardinality, nor methods to express that two properties are each others inverse).

The Web Ontology Language (OWL) was then developed to satisfy these requirements [4]. OWL supports more expressive descriptions of semantic relationships than RDFS. The OWL language, which builds on RDF(S) constructs, consists of three species which support various levels of expressiveness:

(i) *OWL Full* is the complete set of OWL language constructs and places no restrictions.

(ii) *OWL Description Logic* (OWL DL) uses the same constructs as OWL Full but applies some restrictions.

(iii) *OWL Lite* uses a subset of OWL DL constructs some of which are restricted.

The restrictions on OWL DL are the same restrictions that make reasoning systems decidable (i.e., computations will finish in a finite amount of time) and are thus intended to support reasoning system requirements.

Since October 2009, OWL 2 (i.e., the successor of OWL) is the new W3C recommendation [242]. OWL 2 adds some new functionality and is backwards compatible with OWL. New constructs include, e.g., qualified cardinality restrictions (OWL does not have a means to restrain the range of instances to be counted) and disjoint properties (OWL only allows assertions that classes are disjoint, but not for properties). For a complete list of new constructs, we refer to [243]. In this dissertation, the OWL language is considered because it was at the time of modeling the W3C recommendation for Semantic Web languages and OWL 2 was still under development. However, as we will discuss the problems that arose during ontology modeling using OWL, we will clarify whether OWL 2 is able to solve the identified deficiency or not. Since we focus in this chapter on the development of an ontology to describe photos in a CMS, we give an overview of image metadata standards and their semantic representations in the next section.

4.3.2 Image Metadata Standards

According to the semantics they describe, (image) metadata can roughly be subdivided into five classes: (i) low-level, (ii) creational, (iii) content-descriptive, (iv) history, and (v) intellectual property rights (IPR) metadata. In this context, low-level metadata are the metadata that are extracted from the image content, e.g., color histograms, automatically obtained image regions, detected faces, etc. Creational metadata are relevant to the creation of the digital image data, e.g., camera or scanner information and technical information about the capturing condition as well as the software or firmware to create the image. Content-descriptive information describes information about the depicted scene and is relevant for classification and retrieval of images. History metadata hold partial information about how the image got to the present state. These metadata contain a summary of image editing operations that have already been applied to the image and previous versions of the image metadata. Finally, IPR metadata are designed to protect the contents of an image file from misuse and must preserve both moral rights and copyrights.

Probably, the best-known image metadata standard is the Exchangeable Image File Format (EXIF) as it is the specification used by (almost) any digital camera nowadays [244]. The metadata tags provided by the EXIF standard cover metadata related to the capturing process of the image. Recently, there have been efforts to represent the EXIF metadata tags in an RDF Schema ontology [245, 246]. As these are mainly technical metadata, other standards are required to describe supplementary categories of metadata.

RDF is also used for describing (image) metadata by the standards Dublin Core (DC [247]) and PhotoRDF [248]. DC consists of a rudimentary set of 15 elements describing common properties of resources, such as title and creator. For DC, there is an OWL DL version available at the website of Protégé[6] [249]. PhotoRDF is an attempt to standardize a set of categories and labels for personal photo collections using the DC schema as well as an additional schema for technical and content data. The content schema contains 10 fixed keywords to be used in the subject property of the DC schema, such as “Portrait” and “Baby”.

MPEG-7 is developed by the Moving Picture Experts Group (MPEG) and offers a comprehensive set of audiovisual description tools [250]. The metadata elements, their structure and relationships are defined by the standard in the form of *Descriptors* and *Description Schemes*. The latter specify the structure and semantics of the relationships while the former define the syntax and the semantics of each metadata element. MPEG-7 provides rich and general purpose multimedia content description capabilities, including both low-level features and high-level semantic description constructs. However, the lack of

[6]http://protege.stanford.edu/plugins/owl/dc/dublincore.owl

formal semantics in MPEG-7 makes the interpretation of high-level descriptions difficult to cope with. Low-level features are common, as they can be easily extracted from the content, but there is a deficiency of high-level descriptions. For example, different MPEG-7 metadata constructs can be used to denote the same semantic concept [251]. For MPEG-7, there is no commonly agreed mapping to RDF/OWL. There are some existing approaches translating (parts of) MPEG-7 into RDF/OWL. The first one was created by Hunter [252]. This MPEG-7 ontology was firstly developed in RDFS, then converted into DAML+OIL [253], and is now available in OWL Full. The ontology covers the upper part of the Multimedia Description Scheme part of the MPEG-7 standard. Starting from the ontology developed by Hunter, the MPEG-7 OWL DL ontology by Tsinaraki covers the full Multimedia Description Scheme part of the MPEG-7 standard [254]. The MPEG-7 ontology by the Rhizomik initiative[7] has been produced fully automatically using a generic mapping tool (XSD2OWL) and is an OWL Full ontology [255]. The Core Ontology of MultiMedia (COMM) has been created by re-engineering MPEG-7 according to the intended semantics of the standard [256]. It is an OWL DL ontology that covers the low-level audio and visual features represented in MPEG-7 that are used for describing the structure and content of multimedia documents.

The International Press Telecommunications Council (IPTC) Core Schema is a metadata set primarily for photographer's use and aligns with the IPTC Headers [257]. These categorize the metadata fields into four groups regarding to the semantics they describe: (a) *administrative*, (b) *content-descriptive*, (c) *rights*, and (d) *technical* metadata [258]. The *administrative* metadata are data about the content that can not be inferred from the content, such as a free text field for describing the event at which the photo was taken, creation time, and any number of instructions from the provider to the receiver of the photo. The *content-descriptive* metadata consist of 12 fields, including a field "keywords" to express the subject of the content in free text and two fields to describe the most prominent subjects of the photo by one or more codes. However, the use of such free text fields decreases the disclosure of data collections. This is because a free text field contains unstructured data from which the semantics are difficult to obtain. The *rights* and *technical* metadata describe the rights (e.g., a copyright notice) and metadata about the hardware and creational parameters, respectively.

The Visual Resource Association (VRA) is an organization consisting of many American universities, galleries, and art institutes. For maintaining and describing large collections of (annotated) slides, images, and other representa-

[7]http://rhizomik.net/ontologies/mpeg7ontos

tions of works of art, the VRA Core has been defined [259]. VRA Core is a set of metadata elements used to describe works of visual culture as well as the images that represent them. In the context of VRA Core, a work is a physical entity. Similarly to DC, VRA Core defines a set targeted especially at visual resources. There currently exists no commonly accepted mapping from VRA Core to RDF/OWL. At least two conversions have been proposed: RDF/OWL representation by M. van Assem[8] and the ontology by SIMILE[9].

Most standards explained above are mainly focused on one metadata type (e.g., EXIF mainly describes technical metadata), allow full text annotations, or provide a fixed set of descriptive keywords. Furthermore, MPEG-7 lacks high-level semantics. The DIG35 standard, a specification of the International Imaging Association (I3A[10]), defines a set of public metadata for digital images and covers a broad spectrum of metadata fields [260]. The DIG35 metadata definition consists of five logical blocks (i.e., *Basic Image Parameters*, *Image Creation*, *Content Description*, *Image History*, and *IPR*) with a separate common definition, i.e., the fundamental metadata types and fields, that is referred to by the other blocks. While each block is logically partitioned, they may be linked to each other to form additional semantics. The DIG35 is provided in XML accompanied with plain text to describe the semantics.

Table 4.3.2 lists a comparison between these image metadata standards and the different metadata types they cover. As can be seen, no image metadata standard covers the whole range of metadata types. For example, PhotoRDF and VRA Core describe the IPR metadata by only one (free text) field. Further, MPEG-7 and PhotoRDF offer only very limited support for creational metadata, such as the name of the creator, the lens type or camera brand but lack to support technical metadata (such as, aperture and focal length). On the other hand, DIG35 covers the whole range of technical metadata of EXIF, but DIG35 cannot describe low-level features. Furthermore, DIG35 is only available as an XML Schema. Therefore, in the context of our participation in the W3C Multimedia Semantics Incubator Group, we have created an OWL DL ontology for the DIG35 standard [261]

4.3.3 Content Management Systems

Nowadays, a plethora of CMSs are available. In this section, we give an overview of some popular open-source CMSs. Zope [262] is a CMS which

[8]http://www.w3.org/2001/sw/BestPractices/MM/vra-conversion.html
[9]http://simile.mit.edu/2003/10/ontologies/vraCore3
[10]http://www.i3a.org

	PhotoRDF	**Exif**	**VRA Core**	**IPTC Photo**	**MPEG-7**	**DIG35**
Low level	-	-	-	-	✓	-
Creational	limited	✓	limited	✓	limited	✓
Content	*keywords*	*title*	✓	✓	✓	✓
History	-	-	-	-	✓	✓
IPR	*copyright*	*copyright* & *holder*	*description* & *rights*	✓	✓	✓

Table 4.1: Image metadata standards and their covered metadata types.

makes it possible to publish content to the web. Zope comes with an object database back-end, known as the Zope Object Database (ZODB) [263]. ZODB is an object-oriented database for transparently storing Python [264] objects. The Zope framework, centered around the Python scripting language, uses DC as the foundation for cataloguing and syndicating its content. Furthermore, Zope objects support inheritance. Plone [265], a CMS that works hand-in-hand and sits on top of Zope, comes with a work-flow engine, pre-configured security and roles, a set of content types, and multi-lingual support. On the other hand, the open source CMS Drupal [266] interfaces with a relational database through a lightweight database abstraction layer which handles the processing of SQL queries. Content types are derived from a single base type referred to as a node. A node's related metadata is then internally stored in one or different tables from the database. Although Drupal is based on the PHP scripting language [267], it does not allow inheritance. Also Joomla and Exponent CMS reside on the PHP language and a relational database [268, 269]. For these CMSs, support for some image metadata standards can be obtained by installing additional modules or packages to the system, e.g., [270–272]. However, one of the major drawbacks of these systems is that semantics are not formally defined, and consequently, they do not support reasoning about the data. Due to the increasing availability of web-based data sources, the need for integrating heterogeneous data and machine understanding of metadata has grown in importance. As a result, the use of Semantic Web technologies for exchanging and managing digital (multimedia) collections is favored as they permit reasoning about and relating custom defined concepts.

The Flexible Extensible Digital Object Repository Architecture (Fedora) [273] is a framework for managing digital information. In a Fedora repository, all content is managed as data objects that contain either the content or metadata about it. Relationships between digital objects are created using RDF. Bricks (Building Resources for Integrated Cultural Knowledge Services) is a European project, funded by the IST priority in FP6[11] [274]. The Bricks open source framework aims at managing digital assets and consists of decentralized nodes, called *Bnodes*. A Bnode can be seen as a set of services that can become part of the Bricks network so it can communicate through a peer-to-peer mechanism with other Bnodes. Each Bnode is characterized by a web-based interface giving access to administration, cataloguing, consultation, annotation, and personalization of content. Furthermore, Bricks uses the OWL DL language to model the underlying metadata schemas so that decidability of reasoning algorithms is guaranteed.

4.4 A Model for a CMS

In our CMS, documents are indexed in a separate module (i.e., the indexer), such that the storage capacities and network traffic can be optimized [275]. As a result, system-related metadata will be needed to keep track of the storage information. Further, the metadata related to each document are managed by another system (i.e., the metadata service) that is used by the indexer to decide upon the location of the content. Similarly, a security service maintains the policy rules which are based on specific security-related metadata fields. Finally, to allow personalized management, user-centric metadata are introduced. The user-centric metadata should deal with IPR and should allow both annotations and extracted features.

During our participation in the IBBT-PeCMan project, we created a metadata service for a CMS. This metadata service is built using the Bricks framework and implemented in a single Bnode. The structure of the metadata service is shown in Figure 4.2. The access to such a Bnode is mediated through a web service interface. The *Schema management* module stores arbitrary metadata schemas which are modeled in OWL DL. These schemas describe the actual metadata, denoted as *Metadata records*. The *Metadata management* module manages these actual metadata about available content, imports them into the system, and makes it available to querying. The query module uses Lucene[12] for simple text-search querying. Moreover, advanced ontology queries are ex-

[11] http://cordis.europa.eu/ist

[12] http://lucene.apache.org/java/docs

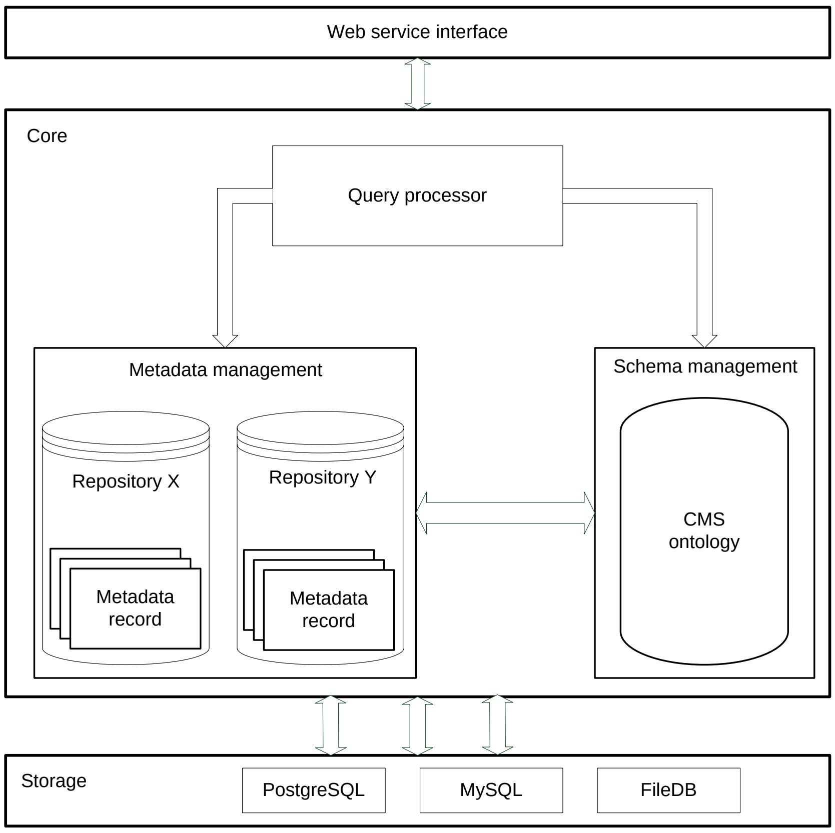


Figure 4.2: Metadata architecture of a Bnode of the Bricks framework.

pressed in SPARQL (SPARQL Protocol And RDF Query Language [276]) and executed by an underlying Jena[13] RDF query processor. An external content manager, such as MySQL[14], is used to store and manage object identifiers that refer to the actual multimedia content.

A constraint for using the Bricks framework is that the metadata model should be OWL DL. However, creating an OWL DL ontology which captures all the semantics of a metadata standard is not so straightforward as explained in Section 4.4.1. Then in Section 4.4.2, we describe the metadata model of our CMS.

[13]http://jena.sourceforge.net/
[14]http://www.mysql.com

```
<owl:Restriction>
   <owl:onProperty rdf:resource="#fNumber"/>
   <owl:allValuesFrom>
      <owl:DataRange>
         <owl11:onDataRange rdf:resource="&xsd;double"/>
         <owl11:minInclusive rdf:datatype="&xsd;double"> 0
         </owl11:minInclusive>
      </owl:DataRange>
   </owl:allValuesFrom>
</owl:Restriction>
```

Fragment 4.1: Restricting the range of a simple data type using OWL 2 constructs.

4.4.1 Modeling an OWL DL Ontology

In this section, we give an overview of some common difficulties that arise when creating an OWL DL ontology by, e.g., converting an XML Schema. Furthermore, some solutions are proposed in order to create an OWL DL ontology. As will be shown, OWL (DL) is insufficient to tackle many common problems for converting an XML Schema with their accompanying text into an ontology without losing formal semantics.

One of the major problems of OWL is its well-known inability to customize the standard data types[15] to deal with data ranges. According to, e.g., the DIG35 specification, many properties have a range of a non-negative double (such as the subject distance, the focal number, color temperature, and the iso saturation). Due to this incapability, we are obliged to use the whole range, i.e., *xsd:double*. In OWL 2, this issue can be tackled by "complex" data ranges that can be constructed from the simpler ones using the *dataOneOf*, the *dataComplementOf*, or the *datatypeRestriction* constructor. These constructs can be combined with one of the available facets to express a restriction (e.g., *minInclusive*, *maxExclusive*). Fragment 4.1 exemplifies the application of OWL 2 data ranges to model the property *fNumber* that has a range of a non-negative double. In contrast, using OWL the range can only be defined by *xsd:double* (i.e., both negative and positive values are included).
Many concepts can have an associated identifier. Expressing the concept of an identifier (cf. a key in a relational database system) in OWL, can be accomplished using the *owl:InverseFunctionalProperty*. However, there is a restriction: the latter property type is only applicable for object properties. Conse-

[15]http://www.w3.org/TR/owl-guide/#term_datatype

quently, if one wants to use a literal value as identifier (ID), an additional ID class should be created. An ID class holds a data type property *uid* to refer to the value of the ID. A drawback of this approach is that some extra complexity is added since one ends up with the definition of 2 properties and an extra class for modeling one single concept. In OWL 2, this is tackled by the *HasKey* construct which allows to define keys for a given class. In OWL 2, the *HasKey* property is not required to be a functional property, but it is always possible to separately state that it is functional.

Sometimes, it is desirable to describe the order of appearance in a sequence. For example, the history metadata section of DIG35 specifies a list of the operations that were applied when editing an image. Consequently, the order in which the operations are listed, is of great importance. RDF provides two container mechanisms to encapsulate data in a user-defined order: the *rdf:Seq* class, which is used for representing ordered lists of literals or resources, and *rdf:List*, which is a class for representing a closed list of items. However, those predefined classes are not OWL DL compatible. To formalize the order in which an object or literal appears in a list, there are two approaches. A first method is to create a (double) linked list of items as depicted in Figure 4.3(a). The linked list is navigated by the two properties *next* and *previous* while the object is referred by the property value. As a new item is inserted, the corresponding properties of the previous, next, and current item need to be adjusted. The latter, however, can not be formalized in the ontology, and consequently, this should be carried out at a higher level (e.g., by the software). A second approach is to create an item class with two properties representing the order and the value. The first property, a data type property (*order*), defines the order of the object and the second (*value*) refers to the object itself as depicted in Figure 4.3(b). The item class is to be referred to by an *owl:ObjectProperty* with no cardinality restrictions. However, the constraint that the value of the *order* property should be unique within the scope of the list, cannot be expressed in OWL. So, in OWL DL only a partial solution for this problem can be elaborated. Also, OWL 2 does not provide a solution for this issue.

Another issue is caused by the fact that many existing ontologies are still OWL Full which make them difficult to reuse and import in existing ontologies [277]. Examples of some popular OWL Full ontologies are Friend Of A Friend (FOAF [278]) and Simple Knowledge Organization Schema (SKOS [279]). In most cases, ontologies are rather OWL Full due to syntactic errors or accidental misuse of the vocabulary, such as the use of RDF-properties instead of OWL-properties. On the other hand for SKOS, the restrictions on OWL DL prevent treating SKOS concepts as OWL classes. Since a SKOS concept is defined as

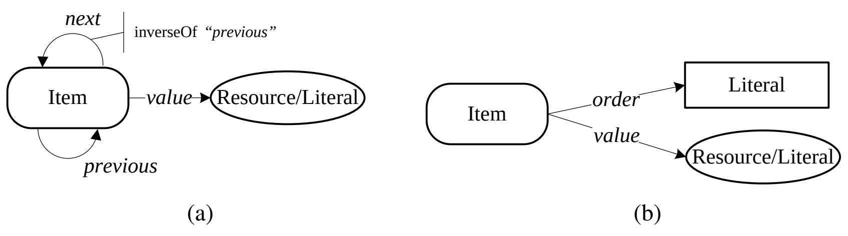


Figure 4.3: Expressing the order of appearance in OWL: (a) a double linked list, (b) a list item.

an OWL class, an instance of a concept should also be an OWL instance, but according to SKOS, an instance must be treated as a class. The latter statement is only supported by OWL Full. Meta-modeling (i.e., the treatment of classes, properties and other entities as individuals) is partially allowed by OWL 2.0: a name can be used for any or all of an individual, a class, or a property. However, a DC property is modeled using an *owl:AnnotationProperty*. The range of the latter must be an individual, literal, or URI, which makes it impossible to refine an existing OWL DL ontology using the OWL DL version of DC. For example, it is not possible to express that a data type property is an equivalent property or sub-property of, e.g., *dc:format* or that the range of *dc:creator* is, e.g., the custom class *Person*. As a result of the OWL Full-ness of many ontologies, we did not reuse any existing ontologies during the modeling process in order to keep our ontologies OWL DL.

4.4.2 CMS Datamodel

As explained before, with every document three types of metadata are related: (i) system-, (ii) security-, and (iii) user-centric metadata. A schematic overview of the structure of the metadata model is given in Figure 4.4. The presented CMS ontology is modularized into three main modules which describe the above three metadata types. Each module then relies on multiple modules that are specialized into the specific media types. Aside design-time advantages, the modularization also minimizes execution overhead when processing data in an RDF store or when using some reasoning services.

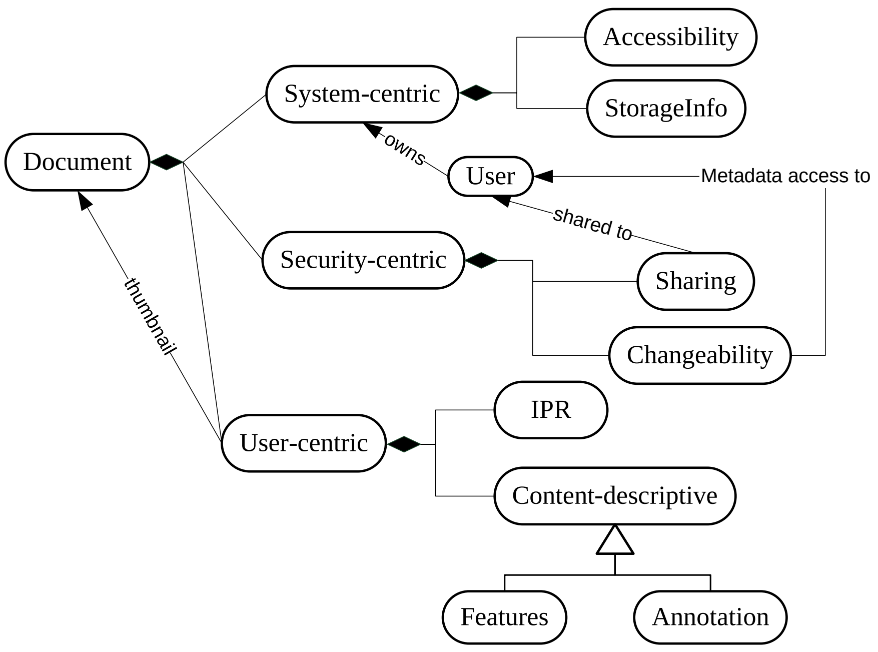


Figure 4.4: A schematic overview of the metadata model in our content management system.

System-centric metadata

The system-centric metadata hold the system-related metadata of a document. Every document is related to its unique owner (who is an end-user of the CMS). The *StorageInfo* class holds data about its current location whereas the *Accessibility* class denotes whether the resource is stored "off-line", "on-line" , or "always on-line". The *StorageInfo* class has, e.g., properties to describe the ID of the remote storage service, the ID of the document on the remote service, the type of the storage service, time stamps, etc.

Security-centric metadata

Security-centric metadata describe the rights that end-users have concerning a document and its metadata. Other users might have some access to the document and can add or change some metadata. Access to the content relates to the *Sharing* class, while the access to the document's metadata is managed by the *Changeability* class. If a document is shared, rights can be set to different users or groups. Typical rights are, e.g., “view” and “hide”. Typical permission settings concerning the accessibility of the metadata are “see”, “add”, “hide”, and “change”. It is often desirable that some documents are inaccessible for other users (e.g., because of copyright issues), but to make its metadata accessible. For example, an audio file may often not be shared because of copyright issues, but by making the metadata accessible, an end-user can inform (s)he owns the content and (s)he can share some metadata (e.g., an opinion about or rating of the content).

User-centric metadata

The user-centric metadata hold the metadata which are assigned to the document by the end-users. The user-centric metadata are content-aware since different types of media (e.g., image, video, music, or text) require specific metadata properties. Within the context of the proposed CMS, the user-centric metadata consist, at a generic level, of the following metadata: (i) a thumbnail, (ii) IPR metadata, and (iii) content-descriptive metadata.

At the most generic level, a thumbnail can be interpreted as some summarization of the content. Therefore, a thumbnail does not need to be of the same type as the document (e.g., a thumbnail of some music file could be a small audio excerpt or an image that represents the cover of the corresponding album). Therefore, in our CMS any document can act as a thumbnail for another one.

The IPR metadata define the exploitation rights of the content. This exploitation can define metadata to impose restrictions upon the use of the content, a mechanism (e.g., watermark or registration), or metadata to specify obligations resulting from the use (e.g., a fee for watching a movie). This IPR mechanism is in line with the IPR systems described in MPEG-21 Rights Expression Language and Rights Data Dictionary [280]. These define standardized language constructs that can be used to create a licensing system that allows a user to define the rights that other persons have upon the content [281]. In the proposed CMS, the IPR-related metadata concern the claimers of the rights (*IPRClaimer*) and the exploitation rights (*IPRExploitation*). An IPR claimer can be a person (*Person*), a group (*Group*), or an organization (*Organization*). The *IPRExploitation* class defines metadata to identify IPR mechanisms, e.g., a watermark, registration, specific restrictions imposed by the right holder, or obligations resulting from the use of the document.

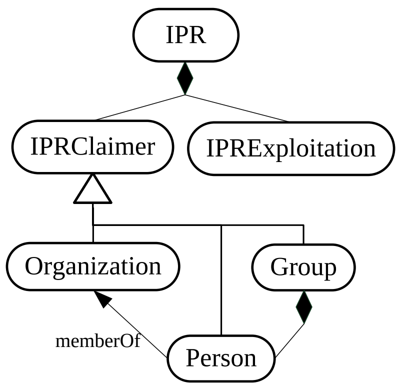


Figure 4.5: The modeling of IPR metadata.

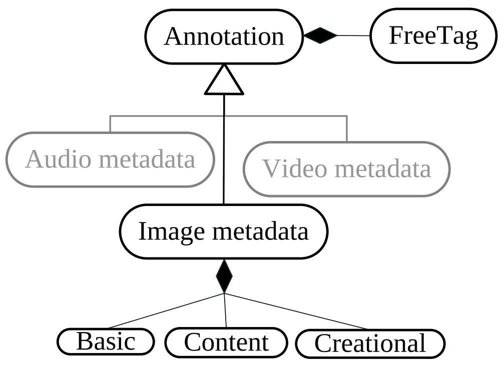


Figure 4.6: Annotations can consist of multiple keywords (*FreeTag*) and can be specialized according to the media type, such as image, audio, and video metadata.

User-centric metadata also concern the data that are related to describe the content. The *Features* class relates to the data that are automatically extracted from the content itself. The resulting (meta)data therefore depend strongly on the algorithms that are used. This class keeps also track of the specific information about the used feature extraction algorithm. The *Annotation* class takes care of the metadata added by a human. For different types of media (such as video, audio, or images), different specializations of the *Annotation* class are created as shown in Figure 4.6.

Any document in the CMS can be freely annotated with keywords (i.e., "free tagging"). Because in the free tagging scenario a tag is divorced from its context, it loses a lot of its intended meaning: different keywords are used to refer to the same concept, spelling errors occur, vernacular or different languages are used, or a word can have multiple meanings. Despite all these disadvantages, the popularity of free tagging is still enormous (e.g., Flickr, Youtube, and Facebook). Therefore, a structuring is introduced by adding an extra context field to refer, on a high level, to the intended semantics of the keyword. The context field can take one of the following values: “who”, “what”, “where”, “when”, or “misc” to refer to a person, a subject, a place, the time, or miscellaneous, respectively. To obtain a more detailed content description, an ontology corresponding the identified requirements (see Section 4.2) has been created to annotate digital images. This ontology consists of three parts: (i) basic image metadata, (ii) creational metadata, and (iii) content-descriptive metadata as shown in Figure 4.6.

Basic image parameters enclose the file name, the image dimension in pixels, the bit size, and the coding format (e.g., “jpeg”, or “tiff”).

Creational metadata describe how the image was created and encompass both general and detailed creation information. The creation information has similar properties as those defined by EXIF. The *image source* property relates to the device that created the image, e.g., a digital camera or software and the settings that were used during the creation process.

The content-descriptive image metadata are the most detailed part of the ontology and allow end-users to create a detailed and queryable description of their images. As depicted in Figure 4.7, the whole image or a region within, as defined by the *Position* class, can be described. By explicitly including this information, we can make statements about specific image regions. As depicted in Figure 4.8, a position can be defined by a textual description (*Comment*), a point (*Point)*, a bounding box (*BoundingBox*), or by a region (*Region*) as a set of closed splines, i.e., Bézier curves (*Spline).*

Additionally, a rating class (defined by a value and the minimum and maximum allowed values) is introduced. In this way, users can rate and comment the whole photo or regions within. Events can be annotated using the *depictedEvent* property. The *depictedItem* property describes all tangible things (represented by the *Tangible thing*[16] class) that are depicted. The latter can be an object or living thing but also a person, a group of people, or an organization. Depicted things can participate in an event of a specified type (e.g., “holiday‘” or “wedding”) and they can be assigned a role. Furthermore, it is possible to relate events with each other by the *relatedEvent* property.

[16]Remark that *owl:Thing* is a predefined class which is the root class (superclass) for all classes

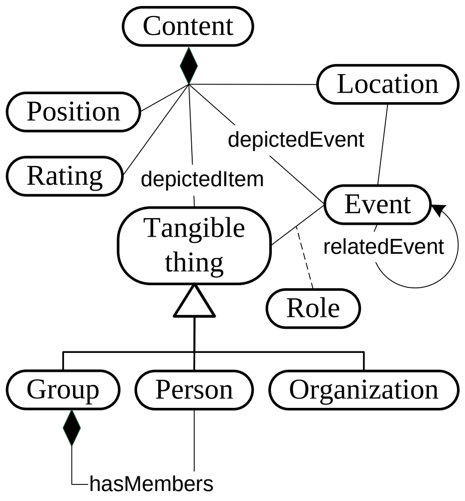


Figure 4.7: Concepts that can be related to (a position within) an image.

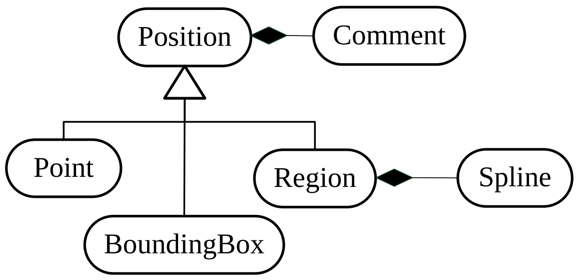


Figure 4.8: Position class as a generalization of region descriptions.

To manage the automatically extracted data, the *Feature* class is introduced. These data strongly depend on the algorithms that are used. Therefore, information about the used algorithm is stored by the association class *Algorithm-Info*. The specific structure of the extracted data (such as the dimensionality of the feature vectors and the type of data) is managed by the *Data* class and can then be modeled by using, e.g., MPEG-7 descriptors. In most CBIR systems, a decision process is used to relate the extracted features with high-level data. For example, a face detection algorithm can detect the faces in a photograph and these facial data can then be used to recognize the depicted person. Therefore, the association class *AlgorithmInfo* is used to relate the *Features* class with the *Content* class as is depicted in Figure 4.9. Analogously, this class

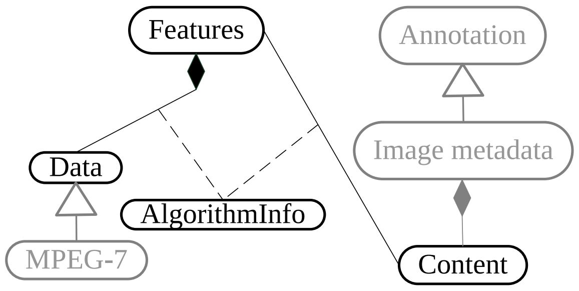


Figure 4.9: Modeling of automatically extracted data in the CMS.

can also be used to store, e.g., the index of the document on a remote system. This index can, e.g., be based on a similarity metric so that documents with a similar content can be retrieved relying on this index. This way, the content can be retrieved relying on both annotations and a similarity metric.

Appendix F lists some OWL schemas of this ontology. Because of the vast size of this ontology, not all schemas are included.

4.4.3 Taxonomy

The annotation of digital resources involves the use of domain-specific metadata. For digital photos, most of these metadata correspond to the concepts humans know from the real world. It has been argued that the human mind naturally organizes its knowledge of the world in a hierarchical structure of a taxonomy [282]. For the purpose of annotation, taxonomies have the advantage of a fixed vocabulary and a hierarchical structure. Within the context of our CMS, a simple but effective taxonomy has been defined. This taxonomy extends the content-descriptive part of our metadata model and allows including a plethora of concepts. As shown in Figure 4.10, the *Tangible thing* class is extended with some additional specializations (such as *Animal*, *Plant*, and *Structure*). These specializations can be on their turn generalizations of more detailed concepts, e.g., *Bike* is a specialization of *Vehicle*, *Cat* and *Dog* are specializations of *Animal*, etc. Using RDF/OWL, semantic properties allow us to model the typical hierarchical structure of a taxonomy. The latter can be easily done using the well-known *rdfs:subClassOf* property as exemplified in Fragment 4.2. Additionally, the application of Semantic Web technologies permits the taxonomy to be extensible, easy to remodel, and enables semantic reasoning.

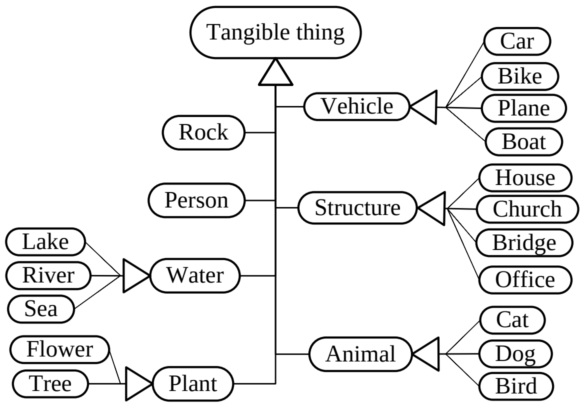


Figure 4.10: An excerpt of the taxonomy used to annotate digital photos.

```
<owl:Class rdf:ID="Vehicle">
 <rdfs:label>Vehicle</rdfs:label>
 <rdfs:subClassOf rdf:resource="#TangibleThing"/>
</owl:Class>
<owl:Class rdf:ID="Car">
 <rdfs:label>Car</rdfs:label>
 <rdfs:subClassOf rdf:resource="#Vehicle/">
</owl:Class>
```

Fragment 4.2: Example of an OWL-implementation of the taxonomy.

To further enrich (photo) annotations, domain-specific properties (e.g., name and birth date for the *Person* class) are created and relationships are outlined between the previously defined concepts. Analogously to the previously created taxonomy of tangible things, many properties can be structured in a taxonomy as well. Figure 4.11 depicts an excerpt of the taxonomy of domain-specific properties for the *Person* class. As can be seen, the property *isFamily* for the *Person* class is hierarchically split up into subproperties which further specify the relationship, such as *isParent* and its inverse *isChild*. Those subproperties are on their turn generalizations of more specific properties, e.g., the properties *isFather* and *isMother* are specializations of the *isParent* property. In OWL/RDF, these constructs can be modeled using the *rdfs:subPropertyOf* property as shown in Fragment 4.3.

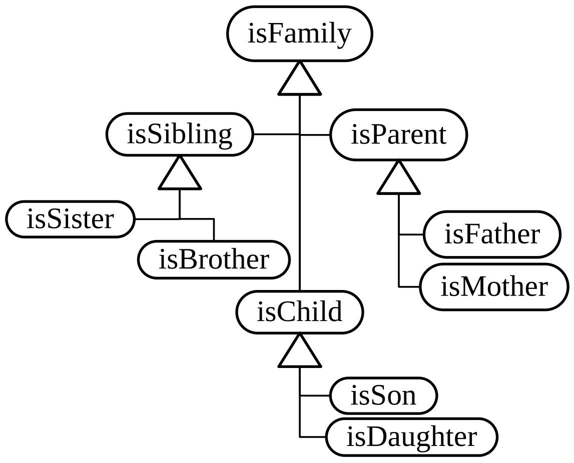


Figure 4.11: A taxonomy of properties for the *Person* class.

```
<owl:ObjectProperty rdf:ID="isFamily">
 <rdf:type rdf:resource="&owl;SymmetricProperty">
 <rdfs:domain rdf:resource="#Person"/>
 <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isParent">
 <rdfs:subPropertyOf rdf:resource="#isFamily"/>
 <owl:inverseOf rdf:resource="#isChild">
 <rdfs:domain rdf:resource="#Person"/>
 <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>
```

Fragment 4.3: Example of a specialization of a domain-specific property for the *Person* class.

4.4.4 Interoperability

Obtaining semantic interoperability using XML is a hard task. This is because XML is actually a structuring language and it does not formally define the semantics. Metadata formats typically consist of an XML Schema accompanied with plain text. The text often describes the semantical meaning of the XML tags specified in the XML Schema. For instance, taking into account different metadata standards, the same tags can have a different meaning, different

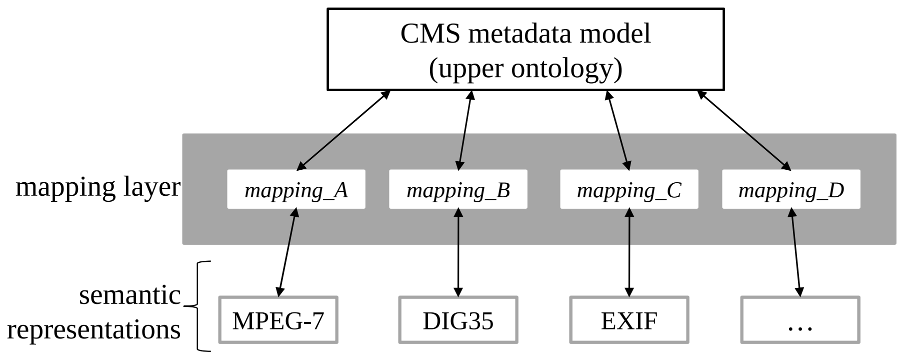


Figure 4.12: Mapping of metadata standards to the CMS.

XML-constructs can be created to model the same concepts, tags with the same meaning can occur in different structures, etc. As a result, mappings between different XML Schemas are difficult to create and are not generic.

Due to the Semantic Web, a lot of research has been recently devoted to ontologies and ontology language standard proposals [283]. Ontologies are generally recognized as an essential tool for allowing communication and knowledge sharing among users and applications by providing a semantically rich description and a common domain of interest. In this context, an additional advantage of Semantic Web technologies is the ability to solve interoperability problems between different metadata standards [284]. This issue was also tackled by the W3C Multimedia Semantics Incubator Group in which we have actively participated. The W3C Multimedia Semantics Incubator Group favours to use semantic representations of metadata standards. These semantic representations can be used to relate to concepts of different ontologies. Also, the W3C Media Annotations Working Group[17] aims to provide an ontology designed to facilitate cross-community data integration of information related to media objects in the Web, such as video, audio, and images. Following this approach, an upper ontology can operate as an intermediate matching step between different ontologies. This upper ontology, which defines a set of common concepts, is then used to match concepts from the semantic representations of metadata standards. This way, the ontology of the CMS acts as an upper ontology as schematically shown in Figure 4.12.

A mapping between ontologies typically consists of basic OWL and RDFS

[17]http://www.w3.org/2008/WebVideo/Annotations/

```
<owl:Class rdf:about="DIG35/Event.owl#Event">
 <owl:equivalentClass rdf:resource="CMS/Event.owl#Event"/>
</owl:Class>
```

Fragment 4.4: The DIG35 class *Event* and the class *Event* of our metadata model are both equivalent.

constructs to relate the respective concepts (i.e., classes, properties, and individuals) of the ontologies. OWL provides properties to relate classes, properties, and individuals with each other as further explained below.

Relating Classes

Basically, equivalence, disjunction, and specialization/generalization relations between classes allow us to describe a hierarchical structure of classes in an ontology. We can also apply these class relations to establish mappings between classes from different ontologies. One class of an ontology may be considered as a subclass of another class of another ontology (*rdfs:subClassOf*) or they can also be equivalent (*owl:equivalentClass*). On the other hand, two classes which have no individual in common may be explicitly declared disjoint (*owl:disjointWith*) with each other. Fragment 4.4 expresses the equivalence between the DIG35 class *Event* and the class *Event* of our metadata model.

OWL also provides expressions to construct a concept that represents a class of individuals which satisfy some common conditions. A complex class can be formed by classical set operations like union, intersection, and complement. Complex class constructions and restrictions allow the description of even more complicated classes. In OWL, a restriction on a certain property can be specified according to its associated value with *owl:hasValue*, its range of values (*owl:someValuesFrom* or *owl:allValuesFrom*), and its cardinality (*owl:min/maxCardinality* or *owl:cardinality*). Fragment 4.5 exemplifies a class mapping using conditions between the concept *Tree* from the above taxonomy and the *Thing* class of DIG35. At first, the *Tree* is declared to be a subclass of *Thing* (line 2) and secondly, a restriction is imposed upon the value of the DIG35 property *name* (lines 3–8).

```
<owl:Class rdf:about="CMS/taxonomy.owl#Tree">
 <rdfs:subClassOf rdf:about="DIG35/Thing.owl#TangibleThing"
     />
 <owl:equivalentClass>
  <owl:Restriction>
   <owl:onProperty rdf:resource="DIG35/Thing.owl#name"/>
    <owl:hasValue rdf:resource="tree"/>
   </owl:Restriction>
 </owl:equivalentClass>
</owl:Class>
```

Fragment 4.5: The concept *Tree* is equivalent with the DIG35 class *TangibleThing* if and only if the DIG35 class its property *name* has the value *"tree"*.

```
<owl:DatatypeProperty rdf:about="CMS/Event.owl#id">
    <owl:equivalentProperty rdf:resource="DIG35/Event.owl#
        eventId"/>
</owl:DatatypeProperty>
```

Fragment 4.6: The two properties *id* and *eventId* are equivalent.

Relating Properties

Analogously to the mapping of classes, properties of different ontologies can also be mapped. Properties can be generalized/specialized (*rdfs:subPropertyOf*), described as equivalent (*owl:equivalentProperty*), or as the inverse of each other (*owl:inverseOf*). Fragment 4.6 exemplifies that the two properties *id* and *eventId* are equivalent .

Relating Individuals

OWL also provides properties for relating individuals with each other. These properties identify equivalence and diversity, and differentiate groups of individuals. The *owl:sameAs* is used to specify that two URIrefs refer to the same individual. Therefore, it can be used to link individuals from different ontologies. The *owl:differentFrom* property has the opposite meaning of *owl:sameAs* and is used to state that two individuals are different. Remark that OWL does not have a unique name assumption. Consequently, individuals must be explic-

```
<owl:AllDifferent>
 <owl:distinctMembers rdf:parseType="Collection">
  <Color rdf:about="#Red"/>
  <Color rdf:about="#Green"/>
  <Color rdf:about="#Blue"/>
  <Color rdf:about="#Cyan"/>
  <Color rdf:about="#Magenta"/>
  <Color rdf:about="#Yellow"/>
  <Color rdf:about="#Black"/>
  <Color rdf:about="#White"/>
 </owl:distinctMembers>
</owl:AllDifferent>
```

Fragment 4.7: This example states that the eight URIrefs are different colors.

itly declared to be different. Also groups of individuals can be differentiated all at once using the predefined *owl:AllDifferent* class. The *owl:distinctMembers* is then used within the class to associate the list of individuals that must be disjoint. Fragment 4.7 exemplifies the use of these properties to declare that the colors (*Red*, *Green*, etc) are distinct.

```
dig35:Rectangle(?r1) ∧ pecman:Retangle(?r2)
∧ dig35:doubleWidth(?r1,?w1) ∧ pecman:doubleWidth(?r2,?w2)
∧ dig35:doubleHeight(?r1,?h1)
∧ pecman:doubleHeight(?r2,?h2)
∧ dig35:topLeftPoint(?r1,?p1)
∧ pecman:topLeftPoint(?r2,?p2)
∧ dig35:x(?p1,?x1) ∧ pecman:x(?p2,?x2)
∧ dig35:y(?p1,?y1) ∧ pecman:y(?p2,?y2)
∧ (?x1=?x2) ∧ (?y1=?y2) ∧ (?w1=?w2) ∧ (?h1=?h2)

⇒ owl:sameAs(?r1,?r2)
```

Fragment 4.8: Rule for mapping two rectangles described by the CMS metadata model and the DIG35 ontology.

Rules

However, those properties to relate classes, properties, and individuals can, e.g., not express that instances of equivalent classes with identical values for (some) properties should actually be the same. So, rules that can be expressed in terms of OWL concepts are needed to provide more powerful deductive reasoning capabilities. To express such a condition, the Semantic Web Rule Language (SWRL) can relieve [285]. SWRL rules are of the form of an implication between an antecedent and consequent: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. The latter is exemplified by Fragment 4.8 which contains the rules to express that two rectangles *r1* and *r2* are equal if they have the same width and height, and if their upper left corner is the same.

4.5 Conclusions

Nowadays, the amount of personal content is continuously growing. Furthermore, this content is typically stored on multiple devices, e.g., on a mobile phone, a desktop computer, or on the WWW. As a result, it becomes more difficult to manage these data. In this chapter, we have presented a solution for this problem by a CMS to manage one's personal content. The metadata model of the proposed CMS relies on an ontology that is created using Semantic Web technologies. This ontology is an OWL DL ontology as reasoning needs to be supported. However, the creation of an OWL DL ontology is not trivial at all. Therefore, we have outlined how some common difficulties, with respect to ontology modeling, can be tackled and what the deficiencies of the OWL DL language are. The presented ontology for the CMS consists of three major parts that describe system-, security-, and user-centric metadata. The system-centric metadata concern the storage location and how these data can be accessed. The security-centric metadata describe the rights that end-users have concerning a document and its metadata. The user-centric metadata hold the metadata which are assigned to a document by the end-users. Within the context of the proposed CMS, the user-centric metadata also consist of metadata that describe IPR metadata and the content. The IPR metadata define the exploitation rights of the content. This exploitation defines metadata to impose restrictions upon the use of the content, a mechanism, or metadata to specify obligations resulting from the use. The content-descriptive metadata concern both the management of extracted features and manually added descriptions (i.e., annotations). These metadata are content-aware since different types of media (such as image, video, music, or text) require specific properties. Two types of annotations are supported: keywords and a content-specific model. In this chapter, we have elaborated on image metadata and we have presented an image ontology that is enclosed by the metadata model of the CMS. The image ontology is a rich ontology that supports many concepts for detailed image annotation and encapsulates basic, creational, and content-descriptive metadata. The image ontology is further extended with a taxonomy that refines additional concepts and properties for annotation. An advantage of the application of Semantic Web technologies is to tackle interoperability issues between different metadata standards. We have explained that this issue can be solved by creating semantic representations of metadata standards and a mapping between the resulting ontologies. From this perspective, an image metadata ontology of the DIG35 standard was created and was the input for the photo use case of the W3C Multimedia Semantics Incubator Group. Furthermore, the presented image ontology of the CMS can be used as an upper-ontology to relate concepts from other image metadata standards and is able to solve the interoperability issues. The latter was the starting point of the W3C's Multimedia Semantics Incubator Group and its successor the Multimedia Annotations Working Group. This way, the proposed CMS also supports the use of image metadata standards for image annotation. Our CMS has native support for (image) metadata standards, in contrast with existing CMSs on which extra modules need to be installed. Our CMS supports reasoning since we rely on Semantic Web technologies. Furthermore, the embedded image ontology covers more metadata types than the state-of-the-art metadata standards.

Our work in this area can be found in following publications:

(i) C. Poppe, G. Martens, E. Mannens, and R. Van de Walle. Personal Content Management System a Semantic Approach. *Visual Communication and Image Representation*, 47:131 – 144, February 2009

(ii) M. Leman, J. Dierickx, and G. Martens. The IPEM-archive conservation and digitalization project. *Journal Of New Music Research*, 30(4):389–393, 2001

(iii) C. Poppe, G. Martens, E. Mannens, and R. Van de Walle. *Data Management in the Semantic Web*, chapter Creating Personal Content Management Systems Using Semantic Web Technologies. Distributed, Cluster and Grid Computing. Nova, 2011

(iv) D. Van Deursen, C. Poppe, G. Martens, E. Mannens, and R. Van de Walle. XML to RDF conversion : a generic approach. In P. Nesi,

K. Ng, and J. Delgado, editors, *Fourth International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, pages 138–144. IEEE Computer Society, 2008

(v) G. Martens, C. Poppe, and R. Van de Walle. Lifting a metadata model to the semantic multimedia world. In *The 2010 International Workshop on Advanced Future Multimedia Services*, 2010. Accepted

Chapter 5

Conclusions

The imagination has made more discoveries than the eye.
– Joseph Joubert (1754 – 1824)

In this dissertation, we have focused on three research domains that concern the extraction and representation of semantic information from digital media. In the first domain, we have dealt with the modeling of textures from digital images and their application for segmentation, classification, and automatic scene analysis. The second domain concerns the extraction of melody and tonality information from polyphonic music recordings. Finally, our third domain deals with the representation of semantic information in content management systems. In the next three sections, we present our conclusions of our research in each domain.

5.1 Texture Analysis

In the computer vision domain, we have focused on texture analysis. Texture is used in many image applications and can be modeled in a variety of ways. Since texture can be related with human perception, we have advocated to rely on features that are related to the human visual system. In Chapter 2, we have presented a novel texture feature relying on the outputs of simple and complex cells from the human visual cortex. These features consist of enhanced grating cell responses combined with Gaussian smoothed Gabor responses. Enhanced grating cell responses give a stronger response to the salient texture-specific orientations and periodicities than the original grating cell features. Although the use of (enhanced) grating cell responses is not sufficient to obtain a successful texture characterization for classification purposes, their combination with Gaussian smoothed Gabor responses produces a texture feature that has more distinguishing characteristics than state-of-the-art texture measures. Furthermore, the behavior of grating cells is an added value for a texture operator since non-texture features, such as isolated pixels or edges, can be distinguished from texture patterns.

Since a priori information, such as the number of textures or the texture types that exist in an image, is (often) unknown for many practical applications, we have to rely on unsupervised machine learning techniques. More specifically, we have chosen to use Self-Organizing Maps (SOMs) since they are both a clustering method and a projection method (from a high-dimensional feature space to a low-dimensional plane). Furthermore, SOMs need no a priori information. Using SOMs for the unsupervised segmentation of textured images, we have shown that the proposed features clearly obtain the best segmentation results compared to other state-of-the-art texture measures.

For the classification of texture information, we employed a semi-supervised approach. Therefore, we have used hierarchical SOMs which are created by retraining the SOM nodes that contain training data from different classes. Experiments conducted on different texture libraries indicate that the presented texture features have excellent discriminating capabilities, and can compete with state-of-the-art texture measures.

However, for most practical applications, the presence of noise in images is a problem in texture characterization. In our experiments, we have shown that our approach is highly robust to various levels of uniform, speckle, and Gaussian noise. In contrast, the classification rate of other well-established texture analysis methods steadily drops as the variance of the noise increases. Next to noise, image compression artifacts can also have a negative influence on texture characterizations, particularly since the number of artifacts increases for high compression ratios. By applying JPEG image compression, we have shown that the presented texture features are highly robust and can deal better with the impact of the compression technique on the high spatial frequencies of the textures than the state-of-the-art texture analysis methods. For example, for the highest tested compression rate (IJG quality level 15), the classification rate was at least 50% higher than using the state-of-the-art features.

Classifying materials from a single image obtained under unknown viewpoint and illumination conditions is a very challenging task. It is similar to texture classification since they both involve the classification based on the visual appearance of a surface. Therefore, we have grouped texture images that correspond to the same material in the same class. This way, we have obtained classes with high intra-variation in terms of orientation and scale. Once again, our proposed texture features achieved the highest classification rate. This way, we have actually proposed a technique to relate semantic information with texture information, and thus to bridge the semantic gap.

Finally, we have examined the application of texture information for outdoor scene analysis. We have restricted our experiments to outdoor photos containing natural textures. Our methodology applies both a bottom-up and a top-down approach. A bottom-up approach is used to compute relevant information from the low-level image data. First, texture features are computed and these are then used to segment the image into regions. Second, the obtained image regions are labeled using a previously trained classifier (a hierarchical SOM). Third, a top-down strategy is deployed by relying on domain knowledge (an ontology) in order to merge image regions and to cope with possible misclassifications. Following this methodology, we achieved an accuracy of 91.9% of the images' pixels that were assigned the correct label.

Regarding our work about the application of texture for automatic material detection, the altering of the scale and the high intra-class variation of the visual appearance of materials cause difficulties. Future work in this domain could include a scale-robust texture measure and the capability to generalize across different instances of the same material. Therefore, we are convinced that more sophisticated machine learning techniques and class-specific feature selection methods are necessary to tackle these problems.

For automatic scene description, solely relying on texture information is not sufficient. Although we have shown that textures can be used to describe some image data, it is, e.g., not applicable for object detection. Also, the altering of the texture scale caused by changes in the perspective, is an issue. It is a fact that recognition by the HVS is yet not fully understood. Moreover, the visual cortex consists of many cell types from which it is believed that their functionality still has to be discovered yet. To achieve a performance that approximates the HVS, we believe one has to rely on both domain knowledge (to understand the relationships between the depicted concepts) and models of certain functions of the HVS that must somehow be integrated into a complete cortical architecture.

Finally, a general issue related to texture analysis methods is the computation time of the feature extraction. To achieve real-time processing using Gabor filter banks, a hardware implementation might be inevitable, especially if one wants to cope the enormous amounts of visual data in limited time. As the computational power of recent graphic processing units (GPU) exceeds recent main processors, GPUs could bring relief to increase the computational efficiency for computing HVS-based features.

5.2 Musical Audio Mining

In Chapter 3, we tackled two issues from the musical audio mining domain, i.e., melody transcription and tonality extraction.

Melody Transcription

At first, we presented a melody transcription system for polyphonic and monophonic music. We aimed to distinguish individual musical notes which are characterized by temporal boundaries and a fixed frequency. The proposed melody transcription system consists of four parts.

The first part is an auditory-model-based multi-pitch extractor which finds for each time frame, the four most salient pitches. Therefore, we have applied the auditory model of Slaney and Lyon which mimics the operation of the cochlea of the inner ear and outputs a cochleagram. The cochleagram contains the auditory nerve patterns for each frequency channel. By applying an autocorrelation-based periodicity analysis in each frequency channel of the cochleagram, we obtain a correlogram. Next, we sum up the channels of the correlogram to obtain a summary correlogram which represents the global periodicity of the signal. At last, we select the highest four peaks from the summary correlogram. These four peaks correspond to the four most salient pitches. At the same, we adjust the evidence of each obtained pitch according to the subharmonic summation theory.

The second part of our melody transcription system is the estimation of the fundamental frequency and the tracing of these frequencies over multiple frames. Therefore, we assumed that the pitch distribution can be obtained from a mixture of Gaussian harmonic tone models. Each harmonic tone model has a fundamental frequency and models the spread of its subharmonics. The weight of each tone model is estimated using the Expectation-Maximization algorithm which is an iterative method for finding maximum likelihood estimates of parameters. The fundamental frequency of a time frame is then characterized by the fundamental frequency of the tone model with the highest weight. However, over adjacent time frames, the output of the latter is not stable because peaks corresponding to the fundamental frequency of the several simultaneous instruments sometimes compete in the probability density functions. So, in order to obtain a note representation, it is necessary to consider the global temporal continuity of these peaks. Hence, we introduce an agent-based peak tracing algorithm. This algorithm sequentially traces for peak trajectories of the weights of the tone models over adjacent time frames in order to select the most dominant and stable frequency trajectories. Salient peaks in close (temporal) proximity of each other are traced by agents.

Since these trajectories still can contain small fluctuations, we need a method to remove these fluctuations in order to obtain a note representation. As a result, the third part of our architecture is a tone creation algorithm that takes as input the obtained trajectories and creates a set of tones and pitch-glides.

Finally, in the fourth and last part of our melody transcription system, we remove redundant tones (such as very short sounding tones) by applying some post-processing.

For the evaluation of our melody transcription system, we have used two polyphonic melody collections that accompanied with the ground truth, i.e., the MIREX 2005 training set and the GMEL set, a self-created set of melodies acquired from popular songs. For the MIREX 2005 training set, the precision and recall for raw pitch detection are 0.68 and 0.65, respectively, and for chroma detection, the precision and recall are 0.81 and 0.79, respectively. For the GMEL set, the precision and recall for raw pitch detection are 0.71 and 0.86, and for chroma detection 0.72 and 0.87, respectively. We have observed that the obtained accuracy of sung melodies is generally lower than the accuracy obtained of non-vocal melodies. This observation can be explained by a couple of factors: (i) the pitch of singing is often less stable than the pitch obtained from an instrument, especially near the onset (portamento), (ii) vibrato, and (iii) formants of the human voice. For the first two cases, our segmentation step can make wrong decisions about the intended frequency. In the third case, certain frequency components, which may differ from the fundamental frequency, can get a higher evidence which results in a wrong estimation of the fundamental frequency. Our system has, however, a major limitation: when the instrument that plays the melody line is silent, the fundamental frequencies of other instruments (i.e., the accompaniment) are detected. Consequently, spectral information of the accompaniment is used for the construction of the tone models which can lead to a false positive, i.e., a redundant tone in the detected melody.

Future work can be done in several sub-domains. At first, the auditory-model-based pitch detector could be improved by carrying out frequency analysis in each channel of the correlogram to avoid peak masking. Further, the automatic segmentation of the musical signal into voiced and unvoiced parts would improve the melody transcription. Also, the detection and tracing of instruments in polyphonic contexts would facilitate the melody transcription process. The latter could be achieved by using timbre information, but this is a complex task due to spectral overlapping between concurrent instruments. Further, information about the musical key and musical domain knowledge could be used to adjust the melody transcription process.

Tonality Extraction

In the second part of Chapter 3, we have examined the extraction and classification of tonality information. Like melody, tonality also entails pitch information, but it manifests at a relatively larger time interval than melody. We have discussed two key-recognition methods for musical audio, the (classical) metric-based method and a novel method based on classification trees. In order to extract the tonality information from a musical audio signal, we have employed pitch induction which is computed in three successive steps as proposed by Leman. In the first step, we compute a cochleagram from the musical signal. Therefore, we have used an auditory model which is an adapted version of the model of Van Immerseel and Martens. In the second step, the cochleagram is transformed into pitch patterns by applying a frame-based autocorrelation, a summation over all channels, and an attenuation to reduce the impact of the too low and too high frequency regions. In the third step, induced pitch patterns are obtained by a leaky integration with a half-decay time of 1.5 s on the pitch images. As was already suggested by Leman, tonality information is low-dimensional. The latter can be deduced from the fact that by applying a PCA on the high-dimensional data, the 4 biggest principal components account for 75% of the total variance.

To obtain the tonality of a musical audio signal, a frequently used method is the metric-based approach. This approach searches for the closest induced pitch pattern which is computed from a probe sound, i.e., a sequence of Shepard tones, that characterizes the tonality. However, when low-dimensionality is exploited, the metric-based approach is unstable as different tonal centers are returned in a 4-, 5-, and 6-dimensional space. To overcome the latter, we have presented a tree-based method which is based on classification trees, a supervised machine learning technique. To train a classification tree, we calculated pitch induction images from audio signals that are recorded from various instruments. Our tree-based approach overcomes the shortcomings of the metric-based approach. An additional advantage of the tree-based method is that the set of training data can always be enlarged with more specific sounds so that specialized trees can be grown for different types of music whereas the metric-based approach is based on a fixed data set (preferably Shepard tones) from which the tonality information is extracted. This information can be exploited in order to significantly reduce the musical search space when a query is launched in a content-based music retrieval system. Also, we think that it must be possible to find correlations between these sequences and other musical features at a higher conceptual level (e.g., emotional connotations).

Future work in this domain resides in the fact that the labeling of the tonal sequences is absolute. A musical piece played at a higher pitch may yield a completely different tonal assignment. Therefore, an extension would be pitch interval patterns instead of (absolute) pitch patterns, or to estimate the pitch mistuning.

5.3 Content Management and Semantic Metadata

In Chapter 4, we have described a system for the management, annotation, and disclosure of personal content that can be stored on different devices. Additionally, we have discussed the creation of a content-descriptive metadata model which embeds an image metadata model for annotating digital photos. The system we have developed, gives a solution for the problems described above. The latter is achieved by relying on Semantic Web technologies for creating the metadata model and by relying on a distributed architecture (which contains services for the indexing, security, and metadata storage). In this dissertation, we have elaborated on the metadata service which has been implemented in a Bnode of the Bricks framework.

At first, we have outlined the requirements of the CMS, and next, we have discussed related work on CMSs and image metadata standards. We have observed that many existing image metadata standards cover different metadata types, but no standard covers them all. Further, the discussed CMSs only support certain metadata standards. This support can be attained by installing additional modules on the CMS. Furthermore, the major drawback of those systems is that the semantics are not formally defined, and as a consequence, reasoning is not supported.

The metadata model of the presented CMS relies on Semantic Web technologies and is an OWL DL ontology as reasoning needs to be supported. Since the modeling of OWL DL ontologies often implies some problems which are due to OWL DL language constraints, we have outlined how some common difficulties can be tackled and what the deficiencies of the OWL DL language are. Next, we have described in detail our metadata model. Our ontology for the CMS consists of three major parts that describe system-, security-, and user-centric metadata. The system-centric metadata concern the storage location and how these data can be accessed. The security-centric metadata describe the rights that end-users have concerning a document and its metadata. Besides IPR metadata, the user-centric metadata consist of content-descriptive metadata. The content-descriptive metadata concern both the management of extracted features and manually added descriptions. These metadata are content-aware since different types of media (such as image, video, music, or text) require specific properties. In this context, we have created a fine-grained image ontology that supports many concepts for detailed image annotation. The image ontology is further extended with a taxonomy that refines additional concepts and properties for annotation. Finally, we have described how interoperability issues are tackled between different metadata standards and are solved by creating semantic representations (ontologies) of metadata standards and by creating an upper ontology to map the respective semantic representations of the metadata standards. The proposed image ontology is used as an upper-ontology to relate concepts from other image metadata standards. From this perspective, an image metadata ontology of the DIG35 standard was created and was the input for the photo use case of the W3C Multimedia Semantics Incubator Group.

Future work focuses on metadata facilities, such as methods to extract information from collaborative filtering and ranking sites which try to provide information that is already present on the web or in a community. Since ontologies evolve over time, versioning is an important feature to ensure the consistency of different versions of ontologies. How to extend and update existing ontologies is an important issue. Further, one of the main purposes of ontologies is to enable knowledge sharing and re-use. However, it can be observed that many ontologies already cover the same domain (e.g., the different ontologies to describe images and photos). As a result, mapping ontologies will be needed to relate the different ontologies that share the same domain knowledge, as well as language constructs to formally relate these concepts.

To conclude this dissertation, much research will have to be devoted to fully bridge the semantic gap. Understanding how humans process certain information is of utmost importance, as well as how and which domain knowledge or personal experiences steer this process. Once we understand these processes, representative models and machine learning techniques can be developed and optimized. We hope that we have convinced the reader that this dissertation, although limited in its scope, contributed to bridge the semantic gap, and that we have aroused the curiosity of the reader about this research domain.

Appendix A

Human Visual System

Functionally, the (human) eye can be compared with a camera that focuses an image that is correctly exposed and sharp. After passing through the pupil of the eye, the light goes on through the lens that projects the light rays onto the retina. In contrast to a photographic film, the retina is far more complex. The retina in the eye contains photoreceptors (rods and cones for black/white and color perception, respectively), which are the only neurons that are directly sensitive to light and convert the image formed by the light rays into nerve impulses. The retina also contains *ganglion cells* that receive their input from many rods and cones. Ganglion cells are neurons with receptive fields of the type *center-on* or *center-off*. As shown in Figure A.1, for a ganglion cell with center-on receptive field, light on the center of the field increases the frequency of the cells firing rate while light on the surround suppresses the firing. Cells with center-off receptive fields have the opposite behaviour.

The optic nerve collects all the axons of the ganglion cells. As shown in Figure A.2, optic nerve fibers run from the retina through the *optic chiasm* and continue via the *optic tract* to the *Lateral Geniculate Nucleus* (LGN) and then via the *optic radiation* to the first stage of the visual cortex, i.e., the *primary visual cortex*. The LGN is situated in the thalamus in the middle of the brain and is the primary processing center of visual information. The LGN consists of 2 parts: one part lies in the left hemisphere and the other lies in the right hemisphere of the thalamus. In the optic chiasm, fibers from the nasal side of each retina cross sides so that the left LGN receives information of the right visual field, and vice versa. The full functionality of the LGN is still unknown but, experiments using functional Magnetic Resonance Imaging (fMRI) in humans have found that both spatial attention and fast eye movements can modulate

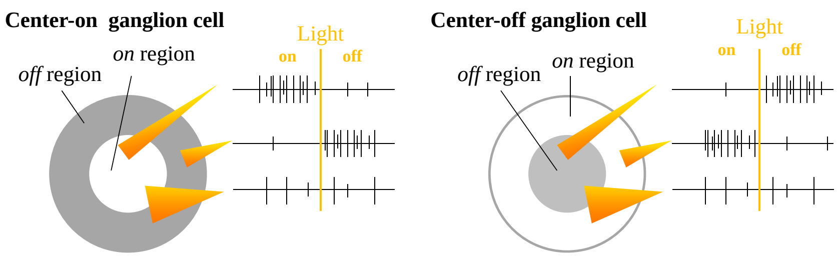


Figure A.1: Representation of center-on and center-off ganglion cells.

activity in the LGN. Additionally, the LGN receives many strong feedback connections from the primary visual cortex. The primary visual cortex thus appears to exert a significant feedback effect on the LGN. In other words, the LGN's main target (i.e., the primary visual cortex) may in turn modify the LGN's own visual responses. The LGN likely helps the visual system focus its attention on the most important information. Neurons of the LGN send their output via the optic radiation directly to the primary visual cortex (also known as *the striate cortex* or *V1*). In primates, nearly all visual information enters the cortex via the primary visual cortex. The V1 area consists of cells having elongated receptive fields. Consequently, they respond best to elongated stimuli such as bars and edges. Hubel and Wiesel categorized these cells as *simple* and *complex* cells [286]. If the response of the cell depends on the stimulus in an approximately linear fashion, the cell is termed simple, otherwise, complex. Complex cells are the most numerous in the primary visual cortex, possibly making up to three-quarters of the cells in this area. There exist many types of complex cells, each with a specific functionality and responding to different input stimuli. Furthermore, it is believed that the functionality of many complex cell types still has to be discovered.

The primary visual cortex sends a large proportion of its connections to the *secondary visual cortex* or *V2*. Though most of the neurons in the V2 area have properties similar to those of the neurons in V1, many others have the distinctive trait of responding to far more complex shapes. The analysis of visual stimuli that begins in V1 and V2 continues through two major cortical systems for processing visual information: the *dorsal* and the *ventral* pathway as depicted in Figure A.3. The dorsal pathway runs from the medial temporal area (MT) to the parietal lobe (which is positioned above the occipital lobe and behind the frontal lobe) and appears to be essential for locating objects. The ventral pathway is running from the primary visual cortex over extrastriate visual areas V2 and V4 to the inferotemporal cortex (IT). The ventral pathway is

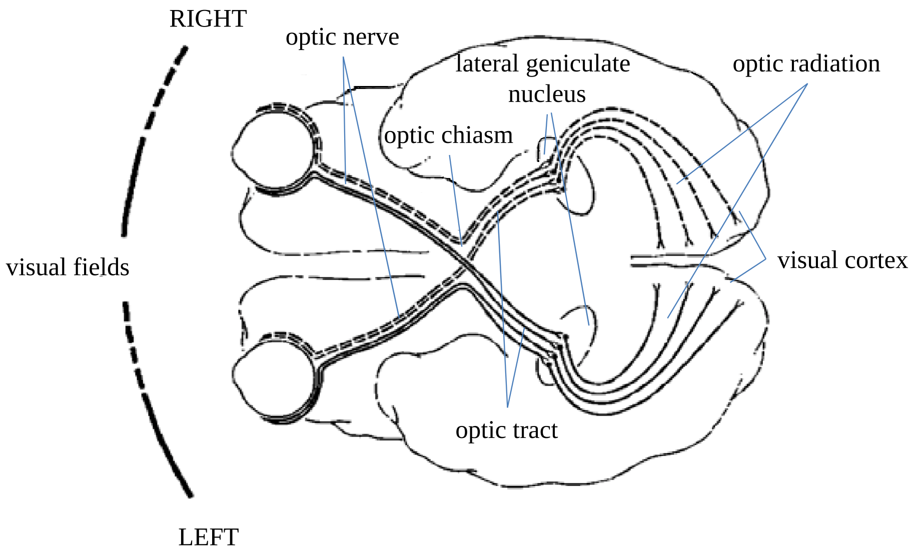


Figure A.2: The visual pathway in the human brain.

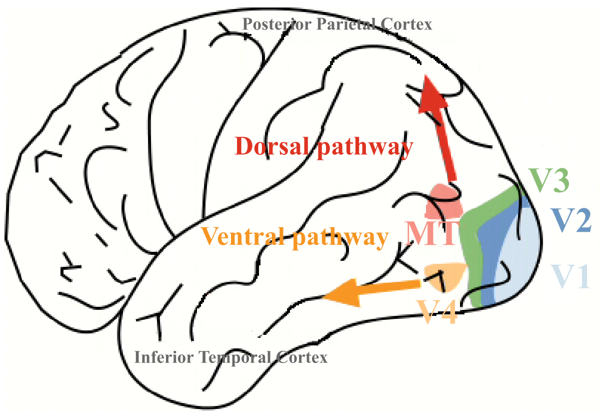


Figure A.3: The dorsal and the ventral pathway.

thought to be involved in recognizing objects. Based on physiological experiments in monkeys, the IT has been postulated to play a central role in object recognition. The IT in turn is a major source of input to prefrontal cortex, i.e., "the center of cognitive control" which is involved in linking perception to memory. Furthermore, physiologic evidence points out that the processing of the ventral visual pathway is mainly feedforward for "immediate recognition tasks" [287].

Appendix B

Color perception

Since color is the primary visual stimulus, the choice of a color system is of great importance for the purpose of proper image retrieval. Color can be modeled and interpreted in many different ways and color systems have been developed for various purposes, such as RGB and CMYK for displaying and printing, YIQ & YUV, for television and video transmission efficiency, XYZ for color standardization, etc.

B.1 HSI

The first geometrical model of color perception was created in the 17th century by Isaac Newton. He epitomized his experiments in light and pigment mixing by ingeniously overlapping the red and violet ends of the spectrum to create a hue circle. This circle shows the spectrum as a continuous gradation of color from red to violet, and from violet to red via the mixed colors carmine, magenta and purple. This circular representation of color is also used in the HSI (or HSL) space, where HSI stands for hue, saturation, and intensity (or lightness) (Gevers, 2001). HSI defines a color space in terms of three constituent components:

(i) *Hue*: the color type (such as red, blue, or yellow),

(ii) *Saturation*: the "vibrancy" of the color; the lower the saturation of a color, the more "grayness" is present and the more faded the color will appear, thus useful to define desaturation as the qualitative inverse of saturation,

(iii) *Intensity* or *Lightness*: the brightness of the color.

Converting RGB color space to HSI color space is obtained by first normalizing RGB values:

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B}.$$

The normalized H, S and I components are then obtained by;

$$h = \cos^{-1}\left(\frac{0.5[(r-g)+(r-b)]}{\sqrt{(r-g)^2+(r-b)(g-b)}}\right) h \in [0,\pi] \text{ for } b \leq g$$
$$h = 2\pi - \cos^{-1}\left(\frac{0.5[(r-g)+(r-b)]}{\sqrt{(r-g)^2+(r-b)(g-b)}}\right) h \in [h, 2\pi] \text{ for } b > g$$
$$s = 1 - 3 \times min(r,g,b) \ s \in [0,1]$$
$$i = \frac{R+G+B}{765} \ i \in [0,1]$$

B.2 Color Opponent Process

In the HVS color vision is mediated by specialized nerve cells in the retina, called cones. The ability to discern different wavelengths of light (i.e. colors) gives us more information for detecting and identifying objects than would be provided solely by black and white vision. The human retina has three types of cones which makes color detection possible: red, green and blue cones. By appropriately mixing these three primary colors it is possible to match all of the colors in the visible spectrum. The latter observation is known as the trichromatic theory (von Helmholtz, 1867). However, the fact that some colors cannot be perceived in combination, e.g. “reddish green” or “bluish yellow”, cannot be explained by the trichromatic theory. This proved to Edwald Hering that the visual substances were organized as opponent processes [288]. In summary, Hering proposed there are six fundamental color processes arranged as three visual contrasts including two opponent processes:

(i) black versus white,

(ii) red - green opponent process,

(iii) blue - yellow opponent process.

By the middle of the 20th century it was proven that both theories are necessary to explain the physiological processes of color perception. So, color vision is a dual process: the trichromatic theory is correct at photo-pigment level by conical photoreceptors in the retina and the opponent theory is correct at the neural level by opponent cells found in the LGN.

Digital images are mainly stored in RGB and can thus easily be transformed into color opponent values (COV) using the following transformation:

$$\begin{bmatrix} RG \\ BY \\ KW \end{bmatrix} = \begin{bmatrix} 1/2 & -1/2 & 0 \\ -1/4 & -1/4 & 1/2 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where RG, BY and KW represent the red-green, blue-yellow and black-white channel, respectively. For R ,G and B values between 0 and 255, the values for RG and BY range between -127.5 and 127.5, while KW ranges between 0 and 127. Remark that the transformation from RGB to HSI is computationally more expensive than the transformation into COV.

Appendix C

The Self-Organizing Map

The Self-Organizing Map (SOM), also called the Kohonen Map, is a single layer artificial neural network that simulates the process of self-organization with a numerical algorithm. The SOM is both a clustering method and projection method. There exists a lot of neurophysiological evidence to support the idea that the SOM captures some of the fundamental processing principles of the human (both visual and auditory) cortex. A SOM converts the non-linear statistical relationships between high-dimensional data into simple, geometric relationships of their image points on a low-dimensional array, also called the display. This low-dimensional display is usually a regular two-dimensional grid of nodes. A schematic representation of a SOM is depicted in Figure C.1. Similar vectors, x and y, are mapped in the vicinity of each other while dissimilar vectors are mapped far away from each other, x and z.

The SOM can be formally described as a non-linear, ordered, smooth mapping of high-dimensional input data onto the elements of a regular, low-dimensional array. Each element in the array, a neuron or node, is associated with a parametric real vector $m_i = \{\mu_{i,1}, \mu_{i,2}...\mu_{i,n}\}$ called the reference or model vector.

Let $x = \{\xi_1, ...\xi_n\}$ be a stochastic data vector. Further we assume a general distance measure (metric) between m_i and x, denoted $d(x, m_i)$. The image of an input vector is defined as the element m_c that matches best with x, i.e. the distance between the input vector and the reference vector is the minimum:

$$c = \arg\left(\min_i\{d(x, m_i\}\right).$$

The element m_c is also called the best matching unit (BMU), or the winning node.

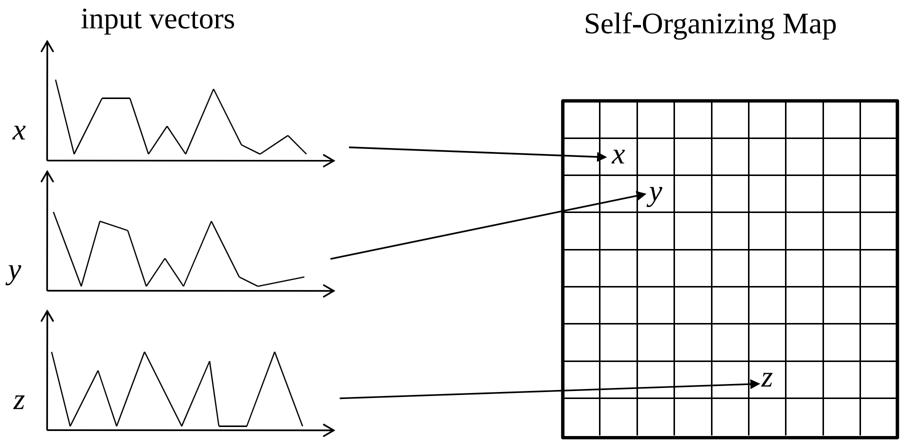


Figure C.1: Schematic representation of the Self-Organizing Map.

Before recursive processing can begin, the m_i must be initialized. The values of the m_i components can be selected at random: starting from an arbitrary initial state, the m_i will finally attain low-dimensional ordered values. This is the basic effect of self-organization. A more careful selection of the initial reference vectors can make the process converge much faster.

During learning the nodes that are topologically close in the array up to a certain geometric distance will activate each other to learn something from the input x. This will result in a local relaxation or smoothing effect on the weight vectors of the neurons in this neighborhood. In continued learning this leads to global ordering.

The learning process can be defined as follows:

- $m_i(t+1) = m_i(t) + h_{c,i}(t)(x(t) - m_i(t))$ for $t = 0, 1, 2, ...$
- $h_{c,i}$ is the neighborhood-function. For convergence, it is necessary that $h_{c,i} \to 0$ when $t \to \infty$.
- usually, $h_{c,i} = h(d(r_c, r_i), i)$ where d is some metric and r_c, r_i are the location vectors of respectively the node c and the node i in the array.
- with increasing $d(r_c, r_i)$, $h_{c,i} \to 0$.

In the literature a neighborhood set of array points around a certain node c is frequently used. Let their index set at time t be denoted by $N_c(t)$ where

$h_{c,i} = \alpha(t)$ if $i \in N_c(t)$ and $h_{c,i} = 0$ if $i \notin N_c(t)$. The value of $0 < \alpha(t) < 1$ is denoted as the learning-rate factor.

Both $\alpha(t)$ and the radius of $N_c(t)$ are usually decreasing monotonically in time during the ordering process. If the neighborhood is too small to start with, the map will not be ordered globally. Instead, locally ordered clusters are seen between which the ordering direction changes discontinuously. This can be avoided by starting with a wide $N_c(0)$ and letting it shrink with time. This ensures that the global order is obtained already at the beginning, whereas towards the end, as the radius gets smaller, the local corrections of the model vectors in the map will be more specific.

During the initial phase, an accurate time function is not important: $\alpha(t)$ can be linear, inversely or exponential proportional to t, e.g., $\alpha(t) = 0.9(1 - t/1001)$ may be a reasonable choice for the first 1000 steps. During this initial period, the m_i get ordered. After the initial ordering phase, over a long period $\alpha(t)$ should attain small values. It is not crucial whether $\alpha(t)$ decreases exponentially or linearly during the final phase.

Since learning is a stochastic process, the final statistical accuracy of the mapping depends on the number of steps in the final convergence phase. This phase must be reasonably long enough.

According to Kohonen, the number of steps should be at least 500 times the number of neurons [59]. A method of evaluating the quality of the resulting map is to calculate the average quantization error over the input samples. The quantization error, often called distortion measure, is the distance between each data vector and its BMU.

After training, for each input sample vector the BMU in the map is searched for, and the average of the respective quantization errors is returned. Another often used quality criterium measures the topology preservation, and is called the topographic error which is the proportion of all data vectors for which first and second BMUs are not adjacent units. However, both measures give the best results when the map has overfitted the data. This may happen when the number of map units is as large or larger than the number of training samples.

Appendix D

Human Auditory System

The processing of sound comprises many layers of analysis and processing in the human auditory system. Furthermore, music listening is affected by many variables, such as sound production, listening conditions, personal memory, or personal experiences which make it rather complex. Since music is mediated as sound, it is important to understand how this information is processed by the first stages of the human auditory system as it precedes the remaining processing of the musical information in the brain.

The human ear can be divided into three distinct parts: (i) the external, (ii) the middle, and (iii) the inner ear.

(i) The *external* ear just functions as an acoustic antenna: it defracts and focusses the sound waves and further acts as a resonator. The final part of the external ear is the eardrum.

(ii) The *middle* ear transmits acoustic energy from the eardrum to the inner ear by allowing adjustment of the difference in impedance between an air environment and a fluid environment. Furthermore, the middle ear also acts as a pressure amplifier so that it is able to capture the available acoustic energy in the air and augment the amplitude of the mechanical-acoustic stimuli in the inner ear.

(iii) The *inner* ear houses two sensory organs: the *vestibule* which is the balance organ and the *cochlea* which is the auditory portion of the inner ear. The cochlea is responsible for the decoding of the sound and the generation of neural responses towards the brain.

Figure D.1 schematically depicts these major components of the human ear.

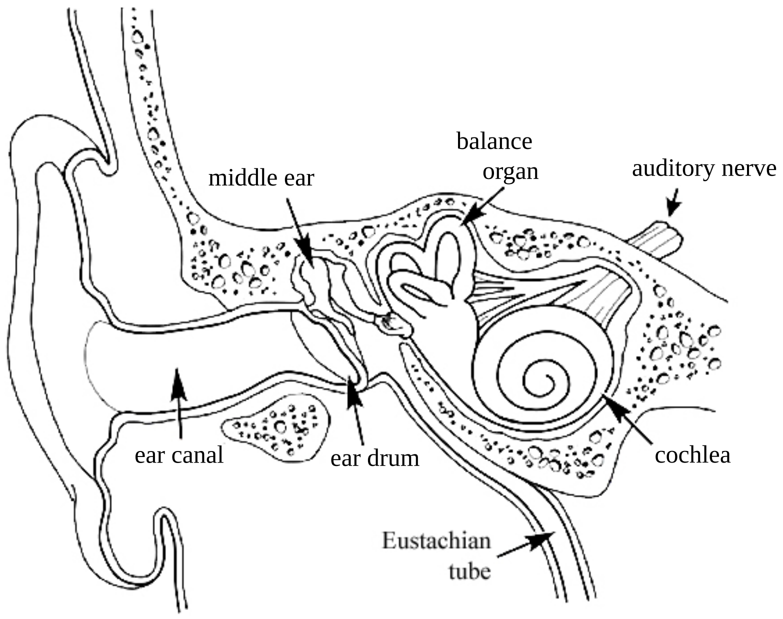


Figure D.1: The various parts of the external, middle, and inner ear.

Sound waves are collected by the external ear and travel through the ear canal where they bump up against the eardrum. The eardrum vibrates in sympathy with these sound waves and, consequently, it moves a series of tiny bones in the middle ear, which carry the vibrations to the cochlea. The cochlea is a spiralled, hollow, and conical chamber filled with fluids which move as a response to the vibrations evoked by a sound coming from the middle ear through the oval window. Structurally, the cochlea is divided by two membranes: *Reissner's membrane*, which keeps the fluids separated, and the *basilar membrane* which is the base of the *organ of Corti* which houses the hair cells, i.e., the sensory cells of hearing. An important function of the basilar membrane is that its movement is responsible for the electric impulses transmitted by the hair cells. A vibration of a certain frequency that reaches the inner ear generates a pressure in the fluids which distorts the basilar membrane. On its turn, this distortion bends the hair cells and causes the firing of electric impulses that are sent to nerve cells. Hair cells located along the inside of the cochlear coil are referred to as *inner* hair cells while hair cells along the outside of the curve are called the *outer* hair cells. The inner hair cells are responsible for the neural output of the cochlea while the outer hair cells operate as an amplifier. In the human cochlea, there are only about 12000 outer hair cells and

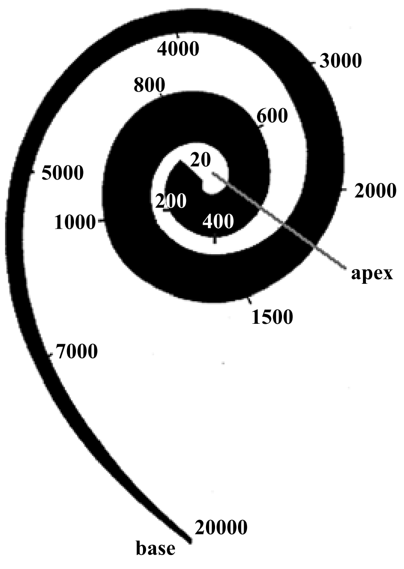


Figure D.2: Spatial arrangement of frequencies (in Hz) along the basilar membrane.

3500 inner hair cells. This number is very low when compared to the millions of photoreceptors in the retina. Furthermore, the final number of hair cells is reached about 10 weeks after conception since they have the inability to proliferate. The inner hair cells are tuned to specific frequencies according to their spatial arrangement. This behaviour, which is also referred to as *tonotopy*, is illustrated in Figure D.2. Low frequencies are situated near the apex of the cochlea, the higher frequencies are situated near the intersection of the middle ear with the inner ear (i.e., close to the oval window). The cochlea acts thus as a frequency analyzer. Experiments indicate that the frequency range of human hearing ranges approximately from 20 Hz to 20000 Hz [289].

There are two theories to explain pitch perception: the *frequency theory* and the *place theory*. The frequency theory states that the pitch is encoded by the frequency of discharge in the primary auditory fiber. Actually, auditory nerve fibers can signal frequencies up to about 4000 Hz in their discharge. However, human pitch perception reaches up 20000 Hz so the frequency theory falls short to fully explain pitch perception. On the other hand, the place theory states that pitch perception is a matter of which fibers are active, determined by the hair cells where they are connected with. This is in agreement with the basilar membrane which is widest and most flexible at the apex of the cochlea, and narrowest and least flexible at the base. Because the base is stiffer than the rest of the membrane, traveling waves always begin at the base and progress toward the apex regardless of how they are initiated. The lowest frequencies propagate thus along the complete route of the cochea. A given frequency displaces obviously more than a single point along the basilar membrane. The amount of displacement leads to different excitation of the hair cells according to their place, being maximum for the cells that match the stimulation frequency and spreading in the direction of the base. In other words, each cell acts as a bandpass filter. Furthermore, the stiffness of the basilar membrane decreases nearly exponentially towards the apex. However, the place theory cannot explain how sound intensity is encoded. This is because the traveling waves excite more than one point. For example, a tone of 100 Hz sets up a traveling wave with a maximum excursion near the apex. At a slightly greater intensity, the maximum excursion is greater, but still occurs at the same point. A point adjacent to the 100 Hz location that, e.g., corresponds to 150 Hz, is also caused to vibrate by the more intense tone. The problem is now how to distinguish the louder 100 Hz tone from a (softer) 150 Hz tone which both cause to vibrate the same point by an equal intensity. It is probable that a combination of both the place and frequency theory is carried out by the auditory system. The frequency theory seems to dominate the lower frequencies up to 4000 Hz, whereas higher frequencies are probably handled according to the place theory [290].

Auditory messages are conveyed to the brain via two pathways: the *primary auditory pathway* which exclusively carries messages from the cochlea and the *reticular sensory pathway* which carries all types of sensory messages. The primary auditory pathway is a relatively short pathway, consisting of four intermediate stations as shown in Figure D.3. Primary auditory fibers enter the brainstem and immediately make connections with secondary neurons in the *cochlear nucleus*. At the level of the cochlear nucleus, the input from the two ears mainly remains separated. Just as the inner hair cells are arranged according to the best frequency, so is the cochlear nucleus. The cochlear nucleus receives input from each spiral ganglion, but also receives input from other parts of the brain (mainly for sound localization). Information about the sound is carried over the *lateral lemniscus* (a tract of axons in the brainstem) to various nuclei in the brainstem (such as the *superior olivary complex* where the first major binaural interactions occur) and ends in the *inferior colliculus* which is located below the visual processing centers. Inferior colliculi possibly integrate information about sound localization before sending it to the cortex and the *medial geniculate nucleus* in the thalamus. Similarly to the lateral geniculate nucleus (see Appendix A), the medial geniculate nucleus acts as a

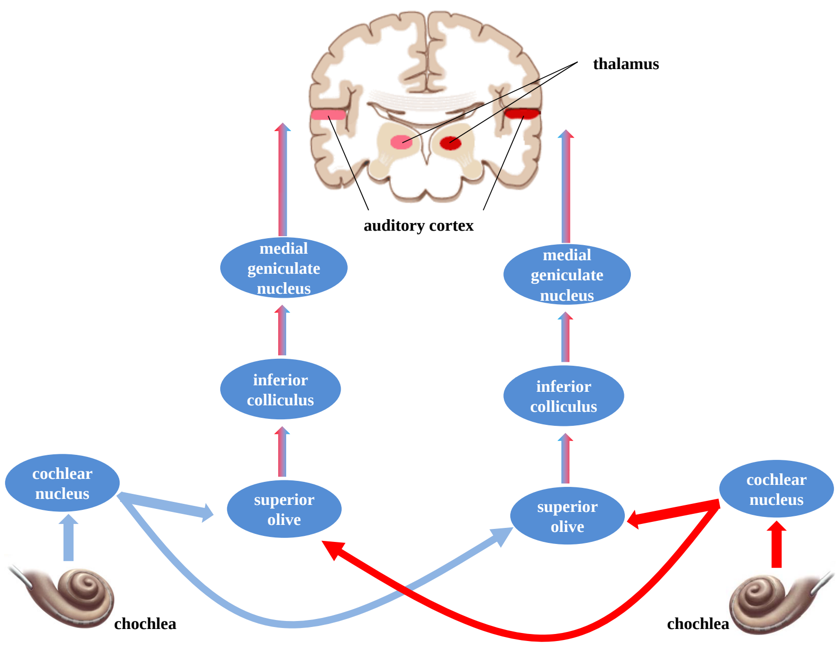


Figure D.3: The primary auditory pathway.

key auditory relay between the inferior colliculus and the primary auditory cortex. The medial geniculate nucleus influences the direction and maintenance of attention. When auditory impulses reach this area, the sound is heard but not fully comprehended. Understanding requires the participation of the auditory association area, the auditory cortex. Figure D.3 depicts a schematic overview of the primary auditory pathway. For a more detailed overview of the auditory pathways, we refer to [291, 292]. The primary auditory pathway ends in the *auditory cortex*.

The auditory cortex is located on the temporal lobe as depicted in Figure D.4 and consists of three parts: the *primary*, the *secondary* and the *tertiary* auditory cortex. The primary auditory cortex is situated in the temporal lobe of the human brain and is responsible for processing sound information. The neurons in this brain region are organized according to the frequency of sound to which they respond best, and thus reflects the tonotopy of the basilar membrane in the cochlea. Individual cells consistently get excited by sounds at specific frequencies or multiples of that frequency. Human brain scans have indicated that only a minor part of this brain area is active when trying to identify musical pitch. Further, it appears that some areas of the primary auditory cortex are special-

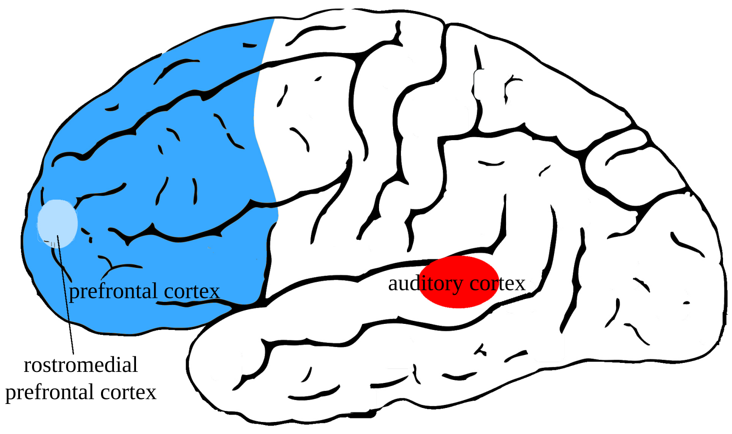


Figure D.4: Localization of the auditory, prefrontal and rostromedial prefrontal cortex.

ized for processing combinations of frequencies while others are specialized for processing modulations of amplitude or frequency. The secondary auditory cortex is likely specialized in processing temporally complex acoustical signals, such as harmonic, melodic and rhythmic patterns, and in processing elementary speech sounds. The tertiary auditory cortex supposedly integrates everything into the overall experience of music [293].

However, pitch is perceived in more places than just the auditory cortex. Janata et al. showed that areas of the *rostromedial prefrontal cortex*, which is thought to aid in the inhibition of emotions, are active during tonality processing [208]. Thus, it is likely that tonal contexts are maintained in cortical regions that mediate interactions between sensory, cognitive, and affective information.

In addition to the pathway that propagates from the cochlea to the cortex (which is also called the ascending pathway), there is also a descending pathway propagating from the cortex to the cochlea. Many of these descending fibers end up synapsing to the outer hair cells as well as to afferent fibers from the inner hair cells. Little is known about this pathway except for the fact that it aids in the detection of sounds in a noisy background.

Appendix E

Classification Tree

Classification trees are used to predict membership of cases or objects in the classes of a categorical dependent variable from their measurements on one or more predictor variables. The goal of classification trees is to predict or explain responses on a categorical dependent variable, or otherwise stated, obtain the most accurate prediction possible. The available techniques have much in common with the techniques used in the more traditional methods of discriminant analysis, cluster analysis, non-parametric statistics, and non-linear estimation.

Tree classification techniques have a number of advantages over other supervised learning techniques. At first, the interpretation of the results summarized in a tree is very simple. This simplicity is not only useful for purposes of rapid classification of new observations, but it can also often yield a much simpler model for explaining why observations are classified or predicted in a particular manner. Secondly, tree methods are non-parametric and non-linear. There is no implicit assumption that the underlying relationships between the predictor variables and the dependent variable are linear, follow some specific non-linear link function or that they are monotonic in nature. Thus, tree methods are particularly well suited for data mining tasks where there is often little a priori knowledge.

Classification trees are constructed top-down, beginning at the top node with the most informative feature, that is the one that maximally reduces entropy. Branches are then created for all values of the descriptor of this node. The training examples are sorted to the appropriate descendant node and the process is repeated recursively. Each node is connected to a possible set of answers and each branch carries a particular test results subset to another node. The final results of using tree methods for classification can be summarized in a series of logical if-then conditions and each terminal node is associated with a single class. Despite the building process of a tree is a recursive procedure, it is still faster than the training of a neural network.

E.1 Specifying the criteria for predictive accuracy

The most accurate prediction is operationally defined as the prediction with the minimum costs. The need for minimizing costs, rather than just the proportion of misclassified cases, arises when some predictions that fail are more catastrophic than others, or when some predictions that fail occur more frequently than others. In many typical applications, costs simply correspond to the proportion of misclassified cases. The notion of costs was developed as a way to generalize, to a broader range of prediction situations, the idea that the best prediction has the lowest misclassification rate.

E.1.1 Priors

Priors, or a priori probabilities, specify how likely it is, without using any prior knowledge of the values for the predictor variables in the model, that a case or object will fall into one of the classes. Minimizing costs, however, does correspond to minimizing the proportion of misclassified cases when priors are taken to be proportional to the class sizes and when misclassification costs are taken to be equal for every class. The a priori probabilities used in minimizing costs can greatly affect the classification of cases or objects. The general point is that the relative size of the priors assigned to each class can be used to adjust the importance of misclassifications for each class. Minimizing costs corresponds to minimizing the overall proportion of misclassified cases when priors are taken to be proportional to the class sizes (and misclassification costs are taken to be equal for every class), because prediction should be better in larger classes to produce an overall lower misclassification rate.

E.1.2 Misclassification Costs

Sometimes more accurate classification is desired for some classes than others for reasons unrelated to relative class sizes. Higher misclassification costs can be specified for misclassifying observations with a certain property then for other observations. But to restate, minimizing costs corresponds to minimizing the proportion of misclassified cases when priors are taken to be proportional to the class sizes and when misclassification costs are taken to be equal for every class.

E.1.3 Case Weights

The application of case weights on a weighting variable as case multipliers for aggregated data sets is also related to the issue of minimizing costs. Interestingly, as an alternative to using case weights for aggregated data sets, one could specify appropriate priors and/or misclassification costs and produce the same results while avoiding the additional processing required to analyze multiple cases with the same values for all variables.

Priors, misclassification costs, and case weights illustrate the wide variety of prediction situations that can be handled using the concept of minimizing costs, as compared to the rather limited prediction situations that can be handled using the narrower idea of minimizing misclassification rates.

E.2 Splitting Rules

Splitting rules attempts to divide a N-dimensional attribute space into homogeneous regions, i.e. regions that contain examples from just one category. The goal of adding new nodes to a tree is to split up the sample space so as to minimize the impurity of the training set. Some algorithms measure goodness instead of impurity, the difference being that goodness values should be maximized while impurity should be minimized.

There exist different splitting criteria to build a classification tree that are both accurate and can reveal important data structure.

The two most frequently used splitting rules are:

(i) *Gini*: excellent for two class dependent variables

(ii) *Twoing*: better for multi-class dependent variables

Both methods yield classification trees of comparable quality (where the relative cost, the used quality criterion, takes into account both the misclassification score and the complexity of the obtained tree). They use different formulas but pick the same splitters in the two class problem. The main difference is that the Gini method attempts to isolate one class and on the other hand the Twoing criterion seeks for splits which are roughly equal in size.

Assume that the set of examples T at the node about to be split contains $n > 0$ instances that belong to one of k categories (Initially T is the entire training set). A hyperplane H divides T into two non-overlapping subsets T_L and T_R. L_j and R_j are the number of instances of category j in T_L and T_R, respectively.

E.2.1 Gini

The Gini index measures the probability of misclassification of a set of instances, rather than the impurity of a split.

$$\begin{aligned} Gini_L &= 1.0 - \sum_{i=1}^{k} \left(\frac{L_i}{|T_L|}^2 \right), \\ Gini_R &= 1.0 - \sum_{i=1}^{k} \left(\frac{R_i}{|T_R|}^2 \right), \\ Impurity_T &= \frac{|T_L| Gini_L + |T_R| Gini_R}{n}, \end{aligned}$$

where $Gini_L$ denotes the Gini index on the left side of the hyperplane and $Gini_R$ denotes the Gini index on the right side of the hyperplane.

E.2.2 Twoing

The value to be computed with the Twoing rules is defined as:

$$Twoing = \frac{T_L}{n} \frac{T_R}{n} \left(\sum_{j=1}^{k} |L_j/|T_L| - R_j/|T_R|| \right)^2,$$

where $|T_R|$ is the number of examples on the right of a split at node T, n is the number of examples at node T, and R_i is the number of examples in category i on the right of the split. The Twoing value is actually a goodness measure rather than an impurity measure.

E.3 Stop Splitting

Splitting could continue until all cases are perfectly classified or predicted. However, this wouldn't make much sense since one would likely end up with a tree structure that is as complex as the original data. So reasonable stopping rules are required. The two most commonly used rules are called *Minimum n* and *Fraction Of Objects*.

E.3.1 Minimum n

One way to control splitting is to allow splitting to continue until all terminal nodes are pure or contain no more than a specified minimum number of cases or objects.

E.3.2 Fraction Of Objects

Another way to control splitting is to allow splitting to continue until all terminal nodes are pure or contain no more cases than a specified minimum fraction of the sizes of one or more classes (in the case of classification problems, or all cases in regression problems). If the priors used in the analysis are equal and class sizes are equal as well, then splitting will stop when all terminal nodes containing more than one class have no more cases than the specified fraction of the class sizes for one or more classes. Alternatively, if the priors used in the analysis are not equal, splitting will stop when all terminal nodes containing more than one class have no more cases than the specified fraction for one or more classes

E.4 Pruning

A major issue that arises when applying classification trees to real data with much random error noise concerns the decision when to stop splitting. If not stopped, the tree algorithm will ultimately extract all information from the data, including information that is not and cannot be predicted in the population with the current set of predictors (most training samples will appear in a separate leaf node). This phenomenon is called overfitting or overlearning. The general approach to addressing this issue is first to stop generating new split nodes when subsequent spits only result in very little overall improvement. For example, if one can predict 9011 splits, then it makes little sense to add the 11th split to the tree. Once the tree building algorithm has stopped, it is also useful to further evaluate the quality of the prediction of the tree with samples of observations that did not participate in the original computations. These methods are used to prune back the tree, i.e. to select a simpler tree than the one obtained when the tree building algorithm stopped, but one that is equally as accurate for predicting or classifying new observations.

E.4.1 Crossvalidation

One approach is to apply the tree computed from one set of observations (learning samples) to another completely independent set of observations (testing samples). If most or all of the splits determined by the analysis of the learning sample are essentially based on random noise, then the prediction for the testing sample will be very poor. Hence one can infer that the selected tree is not very good (useful), and not of the right size.

E.4.2 V-fold crossvalidation

V-fold crossvalidation repeats the crossvalidation analysis many times over with different randomly drawn samples from the data, for every tree size starting at the root of the tree, and applying it to the prediction of observations from randomly selected testing samples. Then uses the tree that shows the best average accuracy for crossvalidated predicted classifications or predicted values. In most cases, this tree will not be the one with the most terminal nodes, i.e. the most complex tree. This method for pruning a tree, and for selecting a smaller tree from a sequence of trees, can be very powerful and is particularly useful for smaller data sets. It is an essential step for generating useful (for prediction) tree models.

E.4.3 Tree selection after pruning

The pruning often results in a sequence of optimally pruned trees. So the next task is to use an appropriate decisive factor to select the right-sized tree from this set of optimal trees. A natural criterion would be the cross-validation costs. There is nothing wrong with choosing the tree with the minimum cross-validation costs as the right-sized tree, but often there will be several trees with crossvalidation costs close to the minimum. But one could also choose as the right-sized tree the smallest sized tree whose cross-validation costs do not differ significantly from the minimum cross-validation costs. Or one could choose the smallest-sized tree whose cross-validation costs do not exceed the minimum cross-validation costs plus the standard error of the cross-validation costs for the minimum cross-validation costs tree.

Appendix F

PeCMan OWL schema

In this appendix, we present the ontology of our proposed CMS in scope of the PeCMan project. The presented ontology consists of a number of OWL schemas. Due to the vast size of the ontology, only some schemas are incorporated in this appendix.

At first, the schema describes a document in the CMS as shown in Fragment F.1. With a document, user-, system-, and security-centric metadata are related. Fragment F.2 shows the OWL schema of the user-centric metadata, and Fragment F.3 outlines the Annotation class. Fragment F.4 shows the modeling of the image metadata and Fragment F.5 describes the image creation metadata.

Fragment F.1: OWL schema describing a document in the CMS.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF[
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
<!ENTITY user "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/UserCentric.owl#">
<!ENTITY system "http://multimedialab.elis.ugent.be/users/gmartens
    /Ontologies/Pecman/v2.0/SystemCentric.owl#">
<!ENTITY security "http://multimedialab.elis.ugent.be/users/
    gmartens/Ontologies/Pecman/v2.0/SecurityCentric.owl#">
 ]>

<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="&xsd;"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://multimedialab.elis.ugent.be/users/gmartens/
        Ontologies/Pecman/v2.0/PDocument.owl#"
    xml:base = "http://multimedialab.elis.ugent.be/users/gmartens/
        Ontologies/Pecman/v2.0/PDocument.owl"
```

```
     xmlns:user="&user;"
     xmlns:system="&system;"
     xmlns:security="&security;"
     >

<owl:Ontology rdf:about="">
  <rdfs:comment xml:lang="en">
    Pecman Ontology for describing PDocument
  </rdfs:comment>
  <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#
      string">version 2.0</owl:versionInfo>
  <owl:imports rdf:resource="&user;"/>
  <owl:imports rdf:resource="&system;"/>
  <owl:imports rdf:resource="&security;"/>
</owl:Ontology>
<!-- ========================================= -->
<!-- Class that describes a document in the CMS -->
 <owl:Class rdf:ID="PecmanDocument">
     <rdfs:label>Real Pecman Document</rdfs:label>
      <rdfs:subClassOf>
         <owl:Restriction>
             <owl:onProperty rdf:resource="#userCentric"/>
             <owl:maxCardinality rdf:datatype="http://www.w3.org
                 /2001/XMLSchema#nonNegativeInteger">1</
                 owl:maxCardinality>
         </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <owl:Restriction>
             <owl:onProperty rdf:resource="#systemCentric"/>
             <owl:maxCardinality rdf:datatype="http://www.w3.org
                 /2001/XMLSchema#nonNegativeInteger">1</
                 owl:maxCardinality>
         </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <owl:Restriction>
             <owl:onProperty rdf:resource="#securityCentric"/>
             <owl:maxCardinality rdf:datatype="http://www.w3.org
                 /2001/XMLSchema#nonNegativeInteger">1</
                 owl:maxCardinality>
         </owl:Restriction>
     </rdfs:subClassOf>
    </owl:Class>

<!-- ============================================= -->
<!-- property to refer to the user-centric metadata -->
<owl:ObjectProperty rdf:ID="userCentric">
     <rdfs:domain rdf:resource="#PecmanDocument"/>
     <rdfs:range rdf:resource="&user;UserCentric"/>
 </owl:ObjectProperty>

<!-- ============================================= -->
<!-- property to refer to the system-centric metadata -->
 <owl:ObjectProperty rdf:ID="systemCentric">
     <rdfs:domain rdf:resource="#PecmanDocument"/>
     <rdfs:range rdf:resource="&system;SystemCentric"/>
 </owl:ObjectProperty>

<!-- ================================================== -->
<!-- property to refer to the security-centric metadata -->
 <owl:ObjectProperty rdf:ID="securityCentric">
     <rdfs:domain rdf:resource="#PecmanDocument"/>
     <rdfs:range rdf:resource="&security;SecurityCentric"/>
 </owl:ObjectProperty>


 </rdf:RDF>
```

Fragment F.2: OWL schema describing user-centric metadata.

```
<?xml version="1.0" encoding="UTF-8"?>
 <!DOCTYPE rdf:RDF[
 <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
 <!ENTITY ipr "http://multimedialab.elis.ugent.be/users/gmartens/
     Ontologies/Pecman/v2.0/IPR.owl#">
 <!ENTITY annot "http://multimedialab.elis.ugent.be/users/gmartens/
     Ontologies/Pecman/v2.0/Annotation.owl#">
 ]>
 <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="&xsd;"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns="http://multimedialab.elis.ugent.be/users/gmartens/
         Ontologies/Pecman/v2.0/UserCentric.owl#"
     xml:base = "http://multimedialab.elis.ugent.be/users/gmartens/
         Ontologies/Pecman/v2.0/UserCentric.owl"
     xmlns:annot="&annot;"
     xmlns:ipr="&ipr;"  >

     <owl:Ontology rdf:about="">
         <rdfs:comment xml:lang="en">
             Pecman Ontology for describing user centric metadata
         </rdfs:comment>
         <owl:versionInfo rdf:datatype="http://www.w3.org/2001/
             XMLSchema#string">version 2.0</owl:versionInfo>
         <owl:imports rdf:resource="&annot;"/>
       </owl:Ontology>
 <!-- ========================================= -->
 <!-- Class that describes user-centric metadata -->
 <owl:Class rdf:ID="UserCentric">
     <rdfs:label>User centric</rdfs:label>
     <rdfs:subClassOf>
         <owl:Restriction>
             <owl:onProperty rdf:resource="#manualAnnotation"/>
             <owl:maxCardinality rdf:datatype="http://www.w3.org
                 /2001/XMLSchema#nonNegativeInteger">1</
                 owl:maxCardinality>
         </owl:Restriction>
     </rdfs:subClassOf>
 </owl:Class>
```

```
<!-- ========================================== -->
<!-- Properties to relate to a thumbnail, manual, and automatic
    annotations -->

<owl:DatatypeProperty rdf:ID="thumbnail">
    <rdfs:domain rdf:resource="#UserCentric"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="manualAnnotation">
    <rdfs:domain rdf:resource="#UserCentric"/>
    <rdfs:range rdf:resource="&annot;ManualAnnotation"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="automaticAnnotation">
    <rdfs:domain rdf:resource="#UserCentric"/>
    <rdfs:range rdf:resource="&annot;AutomaticAnnotation"/>
</owl:ObjectProperty>

<!-- ====================== -->
<!-- Relates to the IPR data-->
<owl:DatatypeProperty rdf:ID="iprData">
    <rdfs:domain rdf:resource="#UserCentric"/>
    <rdfs:range rdf:resource="&ipr;IPR"/>
</owl:DatatypeProperty>

</rdf:RDF>
```

Fragment F.3: OWL schema describing an annotation.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF[
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
<!ENTITY owl "http://www.w3.org/2002/07/owl#">
<!ENTITY log "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/Log.owl#">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:xsd="&
        xsd;" xmlns:owl="&owl;"
    xmlns="http://multimedialab.elis.ugent.be/users/gmartens/
        Ontologies/Pecman/v2.0/Annotation.owl#"
    xml:base="http://multimedialab.elis.ugent.be/users/gmartens/
        Ontologies/Pecman/v2.0/Annotation.owl"
    xmlns:log = "&log;">

    <owl:Ontology rdf:about="">
        <rdfs:comment xml:lang="en"> Ontology for describing an
            annotation </rdfs:comment>
        <owl:versionInfo rdf:datatype="&xsd;string">version 0.1</
            owl:versionInfo>
    <owl:imports rdf:resource="&log;"/>
    </owl:Ontology>
    <!-- ================================= -->
    <!-- Class that describes an annotation -->
```

```
<owl:Class rdf:ID="Annotation">
    <rdfs:label>Annotation</rdfs:label>
</owl:Class>

<!-- ======================= -->
<!--    Automatic annotation -->
<owl:Class rdf:ID="AutomaticAnnotation">
    <rdfs:label>Automatic annotation</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Annotation"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#algorithmInfo"/>
            <owl:maxCardinality rdf:datatype="&xsd;
                nonNegativeInteger">1</owl:maxCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#timeStamp"/>
            <owl:maxCardinality rdf:datatype="&xsd;
                nonNegativeInteger">1</owl:maxCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasFeature"/>
            <owl:minCardinality rdf:datatype="&xsd;
                nonNegativeInteger">0</owl:minCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="algorithmInfo">
    <rdfs:domain rdf:resource="#AutomaticAnnotation"/>
    <rdfs:range rdf:resource="#AlgorithmInfo"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasFeature">
    <rdfs:domain rdf:resource="#AutomaticAnnotation"/>
    <rdfs:range rdf:resource="#Feature"/>
</owl:ObjectProperty>


<owl:DatatypeProperty rdf:ID="timeStamp">
    <rdfs:domain rdf:resource="#AutomaticAnnotation"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
</owl:DatatypeProperty>
<!-- ================ -->
<!-- Class to describe the information
 about an algorithm -->
<owl:Class rdf:ID="AlgorithmInfo">
    <rdfs:label>Algorithm Information</rdfs:label>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#id"/>
            <owl:maxCardinality rdf:datatype="&xsd;
                nonNegativeInteger">1</owl:maxCardinality>
        </owl:Restriction>
```

```
         </rdfs:subClassOf>
         <rdfs:subClassOf>
             <owl:Restriction>
                 <owl:onProperty rdf:resource="#name"/>
                 <owl:maxCardinality rdf:datatype="&xsd;
                     nonNegativeInteger">1</owl:maxCardinality>
             </owl:Restriction>
         </rdfs:subClassOf>
         <rdfs:subClassOf>
             <owl:Restriction>
                 <owl:onProperty rdf:resource="#reference"/>
                 <owl:maxCardinality
                     rdf:datatype="&xsd;nonNegativeInteger">1</
                         owl:maxCardinality>
             </owl:Restriction>
         </rdfs:subClassOf>
     </owl:Class>

     <owl:DatatypeProperty rdf:ID="id">
         <rdfs:domain rdf:resource="#AlgorithmInfo"/>
         <rdfs:range rdf:resource="&xsd;string"/>
     </owl:DatatypeProperty>

     <owl:DatatypeProperty rdf:ID="name">
         <rdfs:domain rdf:resource="#AlgorithmInfo"/>
         <rdfs:range rdf:resource="&xsd;string"/>
     </owl:DatatypeProperty>

     <owl:DatatypeProperty rdf:ID="reference">
         <rdfs:domain rdf:resource="#AlgorithmInfo"/>
         <rdfs:range rdf:resource="&xsd;string"/>
     </owl:DatatypeProperty>


     <!-- ================================ -->
     <!--  Class that represents a feature -->
     <owl:Class rdf:ID="Feature">
         <rdfs:label>Feature</rdfs:label>
     </owl:Class>

     <!-- ========================================= -->
     <!-- Class that represents a manual annotation -->
     <owl:Class rdf:ID="ManualAnnotation">
     </owl:Class>

    <owl:DatatypeProperty rdf:ID="scope">
         <rdfs:domain rdf:resource="#ManualAnnotation"/>
         <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:ID="mimeType">
         <rdfs:domain rdf:resource="#ManualAnnotation"/>
         <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

     <owl:ObjectProperty rdf:ID="freeTag">
         <rdfs:domain rdf:resource="#ManualAnnotation"/>
```

```
        <rdfs:range rdf:resource="#FreeTag"/>
    </owl:ObjectProperty>

   <!-- =============================== -->
    <!--Class that represents a free tag -->
    <owl:Class rdf:ID="FreeTag">
        <rdfs:label>Free tag</rdfs:label>
    <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#context"/>
                <owl:maxCardinality rdf:datatype="&xsd;
                    nonNegativeInteger">1</owl:maxCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

   <owl:DatatypeProperty rdf:ID="context">
        <rdfs:domain rdf:resource="#FreeTag"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:ID="tag">
        <rdfs:domain rdf:resource="#FreeTag"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <owl:ObjectProperty rdf:ID="log">
        <rdfs:domain rdf:resource="#FreeTag"/>
        <rdfs:range rdf:resource="&log;Log"/>
    </owl:ObjectProperty>

</rdf:RDF>
```

Fragment F.4: OWL schema describing image metadata.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF[
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
<!ENTITY owl "http://www.w3.org/2002/07/owl#">
<!ENTITY basic "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/BasicImageParam.owl#" >
<!ENTITY creat "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/ImageCreation.owl#" >
<!ENTITY cont "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/Content.owl#">
<!ENTITY annot "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/Annotation.owl#">
]>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="&xsd;"
    xmlns:owl="&owl;"
    xmlns:basic = "&basic;"
    xmlns:creat = "&creat;"
    xmlns:cont = "&cont;"
```

```
      xmlns:annot="&annot;"
      xmlns = "http://multimedialab.elis.ugent.be/users/gmartens/
          Ontologies/Pecman/v2.0/ImageMetadata.owl#"
      xml:base = "http://multimedialab.elis.ugent.be/users/gmartens/
          Ontologies/Pecman/v2.0/ImageMetadata.owl"
>

    <owl:Ontology rdf:about="">
          <rdfs:comment xml:lang="en">
              Pecman Ontology for describing metadata for digital
                  images
          </rdfs:comment>
          <owl:versionInfo rdf:datatype="http://www.w3.org/2001/
              XMLSchema#string">version 2.0</owl:versionInfo>
        <owl:imports  rdf:resource="&basic;"/>
          <owl:imports  rdf:resource="&creat;"/>
          <owl:imports  rdf:resource="&cont;"/>
        <owl:imports  rdf:resource="&annot;"/>
    </owl:Ontology>

    <!-- ================================= -->
    <!-- Class representing image metadata -->
    <owl:Class rdf:ID="ImageMetadata">
      <rdfs:subClassOf rdf:resource="&annot;ManualAnnotation"/>
      <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#basicParam"/>
          <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
              1</owl:maxCardinality>
        </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#imageCreation"/>
          <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
              1</owl:maxCardinality>
        </owl:Restriction>
      </rdfs:subClassOf>
    </owl:Class>

    <owl:ObjectProperty rdf:ID="basicParam">
        <rdfs:domain rdf:resource="#ImageMetadata"/>
        <rdfs:range rdf:resource="&basic;BasicParam"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:ID="imageCreation">
        <rdfs:domain rdf:resource="#ImageMetadata"/>
        <rdfs:range rdf:resource="&creat;ImageCreation"/>
     </owl:ObjectProperty>

     <owl:ObjectProperty rdf:ID="imageContent">
        <rdfs:domain rdf:resource="#ImageMetadata"/>
        <rdfs:range rdf:resource="&cont;ImageContentDescription"/>
     </owl:ObjectProperty>

  </rdf:RDF>
```

Fragment F.5: OWL schema describing image creation parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF[
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
<!ENTITY owl "http://www.w3.org/2002/07/owl#" >
<!ENTITY pers "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/Person.owl#" >
<!ENTITY proddet "http://multimedialab.elis.ugent.be/users/gmartens
    /Ontologies/Pecman/v2.0/ProductDetails.owl#" >
<!ENTITY dat "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/DateTime.owl#">
]>
    <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns
        #"
     xmlns:rdfs="&rdfs;"
     xmlns:xsd="&xsd;"
  xmlns = "http://multimedialab.elis.ugent.be/users/gmartens/
      Ontologies/Pecman/v2.0/ImageCreation.owl#"
     xml:base = "http://multimedialab.elis.ugent.be/users/gmartens/
         Ontologies/Pecman/v2.0/ImageCreation.owl"
  xmlns:owl="&owl;"
     xmlns:pers = "&pers;"
  xmlns:proddet = "&proddet;"
     xmlns:dat= "&dat;"
 >
  <owl:Ontology rdf:about="">
      <rdfs:comment xml:lang="en">
             Pecman: Ontology for Image Creation metadata
      </rdfs:comment>
      <owl:versionInfo rdf:datatype="http://www.w3.org/2001/
          XMLSchema#string">version 2.0</owl:versionInfo>
      <owl:imports rdf:resource="&pers;"/>
      <owl:imports  rdf:resource="&proddet;"/>
      <owl:imports rdf:resource="&dat;"/>
  </owl:Ontology>
  <!-- ==================== -->
  <!--  Image Creation class -->
  <owl:Class rdf:ID="ImageCreation">
    <rdfs:label>Image creation information</rdfs:label>
    <rdfs:subClassOf>
  <owl:Restriction>
        <owl:onProperty rdf:resource="#generalCreationInfo"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
            1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#detailedCreationInfo"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
            1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:ObjectProperty  rdf:ID="generalCreationInfo">
```

```
    <rdfs:domain rdf:resource="#ImageCreation"/>
    <rdfs:range rdf:resource="#GeneralImageCreation"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty  rdf:ID="detailedCreationInfo">
    <rdfs:domain rdf:resource="#ImageCreation"/>
    <rdfs:range rdf:resource="#DetailedImageCreation"/>
  </owl:ObjectProperty>


  <!-- ============================ -->
  <!-- General Creation Information -->

  <owl:Class rdf:ID="GeneralImageCreation">
       <rdfs:subClassOf>
         <owl:Restriction>
           <owl:onProperty rdf:resource="#creationTime"/>
           <owl:maxCardinality rdf:datatype="&xsd;
               nonNegativeInteger">1</owl:maxCardinality>
         </owl:Restriction>
       </rdfs:subClassOf>
       <rdfs:subClassOf>
         <owl:Restriction>
           <owl:onProperty rdf:resource="#creator"/>
           <owl:maxCardinality rdf:datatype="&xsd;
               nonNegativeInteger">1</owl:maxCardinality>
         </owl:Restriction>
       </rdfs:subClassOf>
       <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#source"/>
         <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger"
             >1</owl:maxCardinality>
       </owl:Restriction>
       </rdfs:subClassOf>
  </owl:Class>

      <owl:DatatypeProperty rdf:ID="creationTime">
  <rdfs:domain rdf:resource="#GeneralImageCreation"/>
  <rdfs:range rdf:resource="&xsd;dateTime"/>
      </owl:DatatypeProperty>

  <!-- creator-->
  <owl:ObjectProperty rdf:ID="creator">
    <rdfs:domain rdf:resource="#GeneralImageCreation"/>
    <rdfs:range  rdf:resource="&pers;Person"/>
  </owl:ObjectProperty>

      <!-- source -->
  <owl:ObjectProperty rdf:ID="source">
    <rdfs:comment>specifies the device source of the digital file</
        rdfs:comment>
    <rdfs:domain rdf:resource="#GeneralImageCreation"/>
    <rdfs:range  rdf:resource="#ImageSources"/>
  </owl:ObjectProperty>

 <owl:Class rdf:ID="ImageSources">
```

```
    <owl:oneOf rdf:parseType="Collection">
      <ImageSource rdf:about="#DigitalCamera"/>
      <ImageSource rdf:about="#ComputerGraphics"/>
   </owl:oneOf>
</owl:Class>

   <owl:Class rdf:ID="ImageSource">
     <rdfs:comment>this class represents the device source of a
         digital file</rdfs:comment>
   </owl:Class>

   <ImageSource rdf:ID="DigitalCamera">
     <rdfs:label>Digital Camera</rdfs:label>
     <rdfs:comment>Image created by digital camera </rdfs:comment>
   </ImageSource>

   <ImageSource rdf:ID="ComputerGraphics">
     <rdfs:label>Computer Graphics</rdfs:label>
     <rdfs:comment>Image digitally created on computers </
         rdfs:comment>
   </ImageSource>

   <!-- ============================ -->
   <!-- Detailed Image Creation info -->
   <owl:Class rdf:ID="DetailedImageCreation"></owl:Class>

   <!-- ================= -->
   <!-- Software Creation -->
   <owl:Class rdf:ID="SoftwareImageCapture">
     <rdfs:comment>this class represent the capture metadata </
         rdfs:comment>
     <rdfs:subClassOf rdf:resource="#DetailedImageCreation"/>
     <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#softwareInfo"/>
         <owl:maxCardinality rdf:datatype="&xsd;positiveInteger">1</
             owl:maxCardinality>
       </owl:Restriction>
     </rdfs:subClassOf>
   </owl:Class>

   <owl:ObjectProperty rdf:ID="softwareInfo">
     <rdfs:domain rdf:resource="#SoftwareImageCapture"/>
     <rdfs:range rdf:resource="&proddet;ProductDetails"/>
   </owl:ObjectProperty>


<!-- ======================================== -->
<!-- Class to represent camera image capture -->
<owl:Class rdf:ID="CameraImageCapture">
     <rdfs:comment>the digital image is created with a digital
         camera</rdfs:comment>
   <rdfs:subClassOf rdf:resource="#DetailedImageCreation"/>
     <rdfs:subClassOf>
         <owl:Restriction>
             <owl:onProperty rdf:resource="#lensInfo"/>
             <owl:maxCardinality rdf:datatype="&xsd;
```

```
              nonNegativeInteger">1</owl:maxCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#cameraInfo"/>
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1<
          /owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#cameraSettings"/>
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1<
          /owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#firmware"/>
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1<
          /owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

  <owl:ObjectProperty rdf:ID="cameraInfo">
    <rdfs:domain rdf:resource="#CameraImageCapture"/>
      <rdfs:range  rdf:resource="&proddet;ProductDetails"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="lensInfo">
    <rdfs:domain rdf:resource="#CameraImageCapture"/>
      <rdfs:range  rdf:resource="&proddet;ProductDetails"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="cameraSettings">
    <rdfs:domain rdf:resource="#CameraImageCapture"/>
    <rdfs:range  rdf:resource="&proddet;ProductDetails"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="firmware">
    <rdfs:domain rdf:resource="#CameraImageCapture"/>
    <rdfs:range  rdf:resource="&proddet;ProductDetails"/>
  </owl:ObjectProperty>


  <!-- ============================================= -->
  <!-- Class that describes camera capture settings  -->
  <owl:Class rdf:ID="CameraCaptureSettings">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#exposureTime"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
            1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#exposureProgram"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
            1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#meteringMode"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
            1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#focalLength"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
            1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#fNumber"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
            1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#flash"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
            1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>


  <owl:DatatypeProperty rdf:ID="exposureTime">
    <rdfs:domain rdf:resource="#CameraCaptureSettings"/>
    <rdfs:range  rdf:resource="&rdfs;Literal"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:ID="fNumber">
    <rdfs:domain rdf:resource="#CameraCaptureSettings"/>
    <rdfs:range  rdf:resource="&xsd;double"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:ID="exposureProgram">
    <rdfs:domain rdf:resource="#CameraCaptureSettings"/>
    <rdfs:range  rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:ID="meteringMode">
    <rdfs:domain rdf:resource="#CameraCaptureSettings"/>
    <rdfs:range  rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="focalLength">
    <rdfs:domain rdf:resource="#CameraCaptureSettings"/>
    <rdfs:range  rdf:resource="&xsd;double"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:ID="flash">
    <rdfs:domain rdf:resource="#CameraCaptureSettings"/>
    <rdfs:range  rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
</rdf:RDF>
```

Classes to describe the content of an image, are managed by the Content class, as shown in Fragment F.6. For example, Fragment F.7 depicts the OWL schema of an event, and Fragment F.8 shows an excerpt of the taxonomy of tangible things.

Fragment F.6: OWL schema holding content-related metadata.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF[
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
<!ENTITY owl "http://www.w3.org/2002/07/owl#">
<!ENTITY ev "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/Event.owl#" >
<!ENTITY tang "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/TangibleThing.owl#">
<!ENTITY log  "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/Log.owl#">
<!ENTITY rat  "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/Rating.owl#">
<!ENTITY pos  "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/Position.owl#">
]>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="&xsd;"
    xmlns:owl="&owl;"
    xmlns:ev = "&ev;"
    xmlns:tang = "&tang;"
    xmlns:log = "&log;"
    xmlns:pos="&pos;"
    xmlns:rat="&rat;"
    xmlns = "http://multimedialab.elis.ugent.be/users/gmartens/
        Ontologies/Pecman/v2.0/Content.owl#"
    xml:base = "http://multimedialab.elis.ugent.be/users/gmartens/
        Ontologies/Pecman/v2.0/Content.owl"
>

<owl:Ontology rdf:about="">
    <rdfs:comment xml:lang="en">
        Pecman: Ontology for describing image content
    </rdfs:comment>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema
        #string">version 2.0</owl:versionInfo>
    <owl:imports rdf:resource="&ev;"/>
    <owl:imports rdf:resource="&pos;"/>
    <owl:imports rdf:resource="&tang;"/>
    <owl:imports rdf:resource="&log;"/>
    <owl:imports rdf:resource="&rat;"/>
</owl:Ontology>

<!-- =================== -->
<!-- Content Description  -->
<owl:Class rdf:ID="ImageContentDescription">
  <rdfs:label>Image content description</rdfs:label>
  <rdfs:subClassOf rdf:resource="&log;LoggedItem"/>
  <rdfs:subClassOf>
     <owl:Restriction>
        <owl:onProperty rdf:resource="#position"/>
         <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger"
             >1</owl:maxCardinality>
     </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>


<owl:ObjectProperty rdf:ID="comment">
    <rdfs:domain rdf:resource="#ImageContentDescription"/>
    <rdfs:range rdf:resource="&log;StringLog"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="depictedItem">
  <rdfs:domain rdf:resource="#ImageContentDescription"/>
  <rdfs:range rdf:resource="&tang;TangibleThing"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="depictedEvent">
  <rdfs:domain rdf:resource="#ImageContentDescription"/>
  <rdfs:range rdf:resource="&ev;Event"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="rating">
  <rdfs:domain rdf:resource="#ImageContentDescription"/>
  <rdfs:range rdf:resource="&rat;Rating"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="position">
  <rdfs:domain rdf:resource="#ImageContentDescription"/>
  <rdfs:range rdf:resource="&pos;Position"/>
</owl:ObjectProperty>
</rdf:RDF>
```

Fragment F.7: OWL schema to describe an event.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF[
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
<!ENTITY dat "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/DateTime.owl#">
<!ENTITY tang "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/TangibleThing.owl#">
<!ENTITY log "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/Log.owl#">
<!ENTITY addr "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/Address.owl#">
<!ENTITY owl "http://www.w3.org/2002/07/owl#">
]>

<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd="&xsd;"
  xmlns:owl="&owl;"
  xmlns = "http://multimedialab.elis.ugent.be/users/gmartens/
      Ontologies/Pecman/v2.0/Event.owl#"
  xml:base = "http://multimedialab.elis.ugent.be/users/gmartens/
      Ontologies/Pecman/v2.0/Event.owl"
  xmlns:dat = "&dat;"
  xmlns:tang = "&tang;"
  xmlns:log="&log;"
  xmlns:addr="&addr;"
>

<owl:Ontology rdf:about="">
  <rdfs:comment xml:lang="en">
    Pecman ontology for describing  events
  </rdfs:comment>
  <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#
      string">version 0.1</owl:versionInfo>
  <owl:imports  rdf:resource="&dat;"/>
  <owl:imports  rdf:resource="&tang;"/>
  <owl:imports rdf:resource="&addr;"/>
  <owl:imports rdf:resource="&log;"/>
</owl:Ontology>

<!-- ============ -->
<!-- Event class  -->
<owl:Class rdf:ID="Event">
  <rdfs:label>Event</rdfs:label>
  <rdfs:subClassOf rdf:resource="&log;LoggedItem"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#id"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
          owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#type"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
          owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
```

```
      <owl:Restriction>
        <owl:onProperty rdf:resource="#description"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1<
            /owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#location"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1<
            /owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#time"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1<
            /owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#duration"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1<
            /owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>


  <owl:DatatypeProperty rdf:ID="id">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Event"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:ID="type">
    <rdfs:domain rdf:resource="#Event"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:ID="description">
    <rdfs:domain rdf:resource="#Event"/>
    <rdfs:range rdf:resource="&xsd;string"/>
   </owl:DatatypeProperty>

   <owl:ObjectProperty rdf:ID="location">
     <rdfs:domain rdf:resource="#Event"/>
     <rdfs:range rdf:resource="&addr;Address"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:ID="time">
     <rdfs:domain rdf:resource="#Event"/>
     <rdfs:range rdf:resource="&dat;DateTime"/>
   </owl:ObjectProperty>

   <owl:DatatypeProperty rdf:ID="duration">
```

```
    <rdfs:domain rdf:resource="#Event"/>
    <rdfs:range rdf:resource="&xsd;duration"/>
  </owl:DatatypeProperty>

  <owl:ObjectProperty rdf:ID="participation">
    <rdfs:domain rdf:resource="#Event"/>
    <rdfs:range rdf:resource="&tang;TangibleThing"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="relatedEvent">
    <rdfs:domain rdf:resource="#Event"/>
    <rdfs:range rdf:resource="#Event"/>
  </owl:ObjectProperty>

  <!-- ================================================= -->
  <!-- Class to describe the role of a thing in an event -->
  <owl:Class rdf:ID="Role">
    <rdfs:label>Role</rdfs:label>
    <rdfs:subClassOf rdf:resource="&log;LoggedItem"/>
    <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#name"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
          owl:cardinality>
    </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:DatatypeProperty rdf:ID="name">
    <rdfs:domain rdf:resource="#Role"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>

  <!-- ================================= -->
  <!--  Class to relate an event with a thing (participant) and the
      role -->
  <owl:Class rdf:ID="EventParticipation">
    <rdfs:label>Event participation</rdfs:label>
    <rdfs:subClassOf rdf:resource="&log;LoggedItem"/>
    <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#event"/>
         <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
             owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#participant"/>
         <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
             owl:cardinality>
       </owl:Restriction>
    </rdfs:subClassOf>
   </owl:Class>

   <owl:ObjectProperty rdf:ID="event">
     <rdfs:domain rdf:resource="#EventParticipation"/>
     <rdfs:range rdf:resource="#Event"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:ID="participant">
     <rdfs:domain rdf:resource="#EventParticipation"/>
     <rdfs:range rdf:resource="&tang;TangibleThing"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:ID="role">
     <rdfs:domain rdf:resource="#EventParticipation"/>
     <rdfs:range rdf:resource="#Role"/>
   </owl:ObjectProperty>

   </rdf:RDF>
```

Fragment F.8: OWL schema describing a taxonomy.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF[
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
<!ENTITY owl "http://www.w3.org/2002/07/owl#">
<!ENTITY tang "http://multimedialab.elis.ugent.be/users/gmartens/
    Ontologies/Pecman/v2.0/TangibleThing.owl#">
]>

<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="&xsd;"
    xmlns:owl="&owl;"
    xmlns = "http://multimedialab.elis.ugent.be/users/gmartens/
        Ontologies/Pecman/v2.0/Taxonomy.owl#"
    xml:base = "http://multimedialab.elis.ugent.be/users/gmartens/
        Ontologies/Pecman/v2.0/Taxonomy.owl"
>
 <owl:Ontology rdf:about="">
   <rdfs:comment xml:lang="en">
     Pecman ontology for describing a taxonomy of things
   </rdfs:comment>
   <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#
       string">version 0.1</owl:versionInfo>
   <owl:imports rdf:resource="&tang;"/>
 </owl:Ontology>

 <!-- ==================================== -->
 <!-- a (short) taxonomy of tangible things -->

  <owl:Class rdf:ID="Vehicle">
   <rdfs:label>Vehicle</rdfs:label>
   <rdfs:subClassOf rdf:resource="&tang;TangibleThing"/>
  </owl:Class>

  <owl:Class rdf:ID="Car">
   <rdfs:label>Car</rdfs:label>
   <rdfs:subClassOf rdf:resource="#Vehicle"/>
  </owl:Class>

```

```
    <owl:Class rdf:ID="Bike">
     <rdfs:label>Vehicle</rdfs:label>
     <rdfs:subClassOf rdf:resource="#Vehicle"/>
    </owl:Class>

    <owl:Class rdf:ID="Plane">
     <rdfs:label>Plane</rdfs:label>
     <rdfs:subClassOf rdf:resource="#Vehicle"/>
    </owl:Class>

     <owl:Class rdf:ID="Boat">
     <rdfs:label>Boat</rdfs:label>
     <rdfs:subClassOf rdf:resource="#Vehicle"/>
    </owl:Class>

    <owl:Class rdf:ID="Animal">
     <rdfs:label>Animal</rdfs:label>
     <rdfs:subClassOf rdf:resource="&tang;TangibleThing"/>
    </owl:Class>

    <owl:Class rdf:ID="Cat">
     <rdfs:label>Cat</rdfs:label>
     <rdfs:subClassOf rdf:resource="#Animal"/>
    </owl:Class>

    <owl:Class rdf:ID="Dog">
     <rdfs:label>Dog</rdfs:label>
     <rdfs:subClassOf rdf:resource="#Animal"/>
    </owl:Class>

    <owl:Class rdf:ID="Bird">
     <rdfs:label>Bird</rdfs:label>
     <rdfs:subClassOf rdf:resource="#Animal"/>
    </owl:Class>

  </rdf:RDF>
```

References

[1] John F. Gantz, C. Chute, A. Manfrediz, S. Minton, D. Reinsel, W. Schlichting, and A. Toncheva. The diverse and exploding digital universe, March 2008. An IDC White Paper.

[2] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):13491380, 2000.

[3] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible markup language (xml) 1.0 (fifth edition). Available on: `http://www.w3.org/TR/xml/`.

[4] P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation, 10 february 2004. Available on: `http://www.w3.org/TR/owl-semantics`.

[5] M. Leman, J. Dierickx, and G. Martens. The IPEM-archive conservation and digitalization project. *Journal Of New Music Research*, 30(4):389–393, 2001.

[6] G. Martens, H. De Meyer, B. De Baets, M. Leman, M. Lesaffre, and J.P. Martens. Tree-based versus distance-based key recognition in musical audio. *Soft Computing*, 9(8):565–574, 2005.

[7] C. Poppe, G. Martens, S. De Bruyne, P. Lambert, and R. Van de Walle. Robust spatio-temporal multimodal background subtraction for video surveillance. *Optical Engineering*, 47:110101, October 2008.

[8] C. Poppe, G. Martens, E. Mannens, and R. Van de Walle. Personal Content Management System a Semantic Approach. *Visual Communication and Image Representation*, 47:131 – 144, February 2009.

[9] G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Semi-supervised classification of robust texture features inspired by the human visual system. *Visual Communication and Image Representation*, 2010. Under review.

[10] G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Noise and compression robust biological features for texture classification. *The Visual Computer*, 26(6-8):915–922, June 2010.

[11] C. Poppe, G. Martens, P. De Potter, and R. Van de Walle. Semantic web technologies for surveillance metadata. *Multimedia Tools and Applications*.

[12] G. Martens, P. Lambert, and R. Van de Walle. *Self-Organizing Maps*, chapter Bridging the semantic gap using human vision system inspired features, pages 261–276. In-Tech, 2010.

[13] C. Poppe, G. Martens, E. Mannens, and R. Van de Walle. *Data Management in the Semantic Web*, chapter Creating Personal Content Management Systems Using Semantic Web Technologies. Distributed, Cluster and Grid Computing. Nova, 2011.

[14] G. Martens, H. De Meyer, B. De Baets, M. Leman, J.P. Martens, L. Clarisse, and M. Lesaffre. A tonality-oriented symbolic representation of musical audio generated by classiffication trees. In *The EUROFUSE Workshop on Information Systems*, 2002.

[15] M. Leman, L. Clarisse, B. De Baets, H. De Meyer, M. Lesaffre, G. Martens, J.P. Martens, and D. Van Steelant. Tendencies, perspectives, and opportunities of musical audio-mining. In A Calvo-Manzano, A Perez-Lopez, and J Salvador Santiago, editors, *Revista de Acustica*, volume 33, 2002.

[16] M. Lesaffre, K. Tanghe, G. Martens, D. Moelants, M. Leman, B. De Baets, H. De Meyer, and J.P. Martens. The MAMI Query-By-Voice Experiment: Collecting and annotating vocal queries for music information retrieval. In *Proceedings of the International Conference on Music Information Retrieval*, pages 65–71, 2003.

[17] M. Lesaffre, D. Moelants, M. Leman, B. De Baets, H. De Meyer, G. Martens, and J.P. Martens. User behavior in the spontaneous reproduction of musical pieces by vocal query. In *Proceedings 5th Triennial ESCOM Conference*, pages 208–211, 2003.

[18] G. Martens, C. Poppe, and R. Van de Walle. Enhanced grating cell features for unsupervised texture segmentation. In W. Ponweiser, editor, *Performance Evaluation for Computer Vision: 31ste AAPR/OAGM Workshop 2007*, pages 9–16, Wien, 5 2007. Osterreichische Computer Gesellschaft.

[19] C. Poppe, G. Martens, P. Lambert, and R. Van de Walle. Mixture Models Based Background Subtraction for Video Surveillance Applications. In *Lecture Notes in Computer Science: Proceedings 12th International Conference on Computer Analysis of Images and Patterns*, volume 4673, pages 28–35, August 2007.

[20] C. Poppe, G. Martens, P. Lambert, and R. Van de Walle. Improved Background Mixture Models for Video Surveillance Applications. *Lecture Notes in Computer Science, 8th Asian Conference on Computer Vision(ACCV 2007)*, 4843:251–260, 2007.

[21] C. Poppe, G. Martens, P. Lambert, and R. Van de Walle. Dealing with Quick Illumination Changes when using Background Mixture Models. In *Proceedings 8th Asian Conference on Computer Vision*, volume 4843, Tokyo, November 2007.

[22] C. Poppe, G. Martens, S. De Bruyne, P. Lambert, and R. Van de Walle. Dealing with Gradual Lighting Changes in Video Surveillance for Indoor Environments. In *Proceedings of 15th World Congress on Intelligent Transport Systems*, November 2008.

[23] C. Poppe, S. De Bruyne, G. Martens, P. Lambert, and R. Van de Walle. Intelligent Preprocessing for Fast Moving Object Detection. In *Proceedings of SPIE Security and Defense*, volume 6978, March 2008.

[24] D. Van Deursen, C. Poppe, G. Martens, E. Mannens, and R. Van de Walle. XML to RDF conversion : a generic approach. In P. Nesi, K. Ng, and J. Delgado, editors, *Fourth International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, pages 138–144. IEEE Computer Society, 2008.

[25] G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Unsupervised texture segmentation and labeling using biologically inspired features. In D. Feng, D. Sikora, W.C. Siu, J. Zhang, J. Guan, L. Dugelay, Q. Wu, and Li W., editors, *Proceedings of the 2008 IEEE 10th workshop on Multimedia Signal Processing*, pages 159–164, Cairns, 10 2008. IEEE Signal Processing Society.

[26] G. Martens, C. Poppe, P. Lambert, and R. Van de Walle. Perceptual-based textures for scene labeling: a bottom-up and a top-down approach, 5 2010.

[27] G. Martens, C. Poppe, and R. Van de Walle. Lifting a metadata model to the semantic multimedia world. In *The 2010 International Workshop on Advanced Future Multimedia Services*, 2010. Accepted.

[28] R. Baeza-Yates and B. Ribeiro-Neto. Modern information retrieval. *Addison-Wesley /ACM Press*, 1999.

[29] E.H. Land. Color vision and the natural image. *NAS*, 45:115–129, 1959.

[30] B. Heisele, T. Serre, M. Pontil, T. Vetter, and T. Poggio. Categorization by learning and combining object parts. In *in NIPS*, pages 1239–1245. MIT Press, 2001.

[31] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermdediate complexity and their use in classification. *Nature Neuroscience*, 5(7):682–687, 2002.

[32] L.W. Renninger and J. Malik. When is scene recognition just texture recognition? *Journal Vision Research*, 44:2301–2311, 2004.

[33] A.C. Bovik, M. Clark, and W.S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):55–73, January 1990.

[34] B.B. Chaudhuriand, N. Sarkar, and P. Kundu. Improved fractal geometry based texture segmentation technique. *VISP*, 140:233–241, 1993.

[35] R.M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

[36] W. Richards and A. Polit. Texture matching. *Kybernetic*, 16:155–162, 1974.

[37] J.K. Hawkins. Textural properties for pattern recognition. In *In Picture Processing and Psychopictorics*, pages 347–370, 1970.

[38] R.M. Haralickand K. Shanmugam and I. Dinstein. Textural features for image classification. *SMC*, 3(6):610–621, November 1973.

[39] G. N. Srinivasan and G. Shobha. Statistical texture analysis. In *PROCEEDINGS OF WORLD ACADEMY OF SCIENCE,ENGINEERING AND TECHNOLOGY*, volume 36, pages 1264–1269, 2008.

[40] G. Smith. Image texture analysis using zero crossing information, 1998. PhD. Thesis, University of Queensland.

[41] R. Chellappa and S. Chatterjee. Classification of textures using gaussian markov random fields. *ASSP*, 33:959–963, August 1995.

[42] H. Derin and H. Elliott. Modeling and segmentation of noisy and textured images using gibbs random fields. *T-PAMI*, 9:39–55, 1987.

[43] T. Ojalaand M. Pietikainen D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59, January 1996.

[44] G. Van de Wouwer, P. Scheunders, and D. Van Dyck. Statistical texture characterization from discrete wavelet representations. *IEEE Transactions on Image Processing*, 8(4):592–598.

[45] M.N. Do and M. Vetterli. Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance. *IEEE Transactions on Image Processing*, 11(2):146–158, 2002.

[46] H. Hassanpour, G.R. Rad, H. Yousefian, and A. Zehtabian. A novel pixon-based approach for image segmentation using wavelet thresholding method. In *ICIAR '09: Proceedings of the 6th International Conference on Image Analysis and Recognition*, pages 191–200, Berlin, Heidelberg, 2009. Springer-Verlag.

[47] Y. Xia, D. Feng, R.C. Zhao, and Y.N. Zhang. Multifractal signature estimation for textured image segmentation. *Pattern Recogn. Lett.*, 31(2):163–169, 2010.

[48] F.W. Campbell and J.G. Robson. Application of fourier analysis to the visibility of gratings. *J Physiol*, 197(3):551–566, August 1968.

[49] R.W. Conners and C.A. Harlow. A theoretical comparison of texture algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(3):204–222, 1980.

[50] D. Barba and J. Ronsin. Image segmentation using new measure of texture feature. *Digital Signal Processing*, 4:749–753, 1984.

[51] H. Voorhees and T. Poggio. Detecting textons and texture boundaries in natural images. In *Proceedings of the First International Conference on Computer Vision*, pages 250–258, 1987.

[52] D. Blostein and N. Ahuja. Shape from texture: Integrating texture-element extraction and surface estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:1233–1251, 1989.

[53] F. Tomita and S. Tsuji. *Computer Analysis of Visual Textures*. Kluwer Academic Publishers, Boston, 1990.

[54] S.W. Zucker. Toward a model of texture. *Computer Graphics Image Processing*, 5(2):190–202, June 1976.

[55] K.S. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, New Jersey, 1982.

[56] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, second edition, 1990.

[57] S.-Y. Hsu. The mahalanobis classifier with the generalized inverse approach for automated analysis of imagery texture data. *Computer Graphics and Image Processing*, 9(2), 1979.

[58] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[59] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 1995.

[60] T.E. Hall and G.B. Giannakis. Texture model validation using higher-order statistics. In *International Proceedings of ICASSP*, volume 4, pages 2673–2676, 1991.

[61] P. Brodatz. Textures: A photographic album for artists and designers, 1966. Dover, New York.

[62] T. Maenpaa, M. Turtinen, and M. Pietikainen. Real-time surface inspection by texture. *Real Time Imaging*, 9(5):289–296, October 2003.

[63] T. Ojala, M. Pietikainen, and T. Maenpaa. Gray scale and rotation invariant texture classification with local binary patterns. In *ECCV00*, pages 404–420, 2000.

[64] J. Beck, A. Sutter, and A. Ivry. Spatial frequency channels and perceptual grouping in texture segregation. *Computer Vision Graphics Image Process*, 37:299–352, 1987.

[65] D.G. Albrecht R.L. De Valois and L.G. Thorell. Spatial-frequency selectivity of cells in macaque visual cortex. *Vision Res.*, (5):545–559, 1982.

[66] J.G. Daugman. Uncertainty relation for resolution in space, spatial frequency and orientation optimization by two-dimensional visual cortical filters. *J. Opt. Soc. Am.*, 2:1160–1169, 1985.

[67] W. Heisenberg. über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. *Zeitschrift für Physik*, pages 172–198, 1927.

[68] M. Turner. Texture discrimination by gabor functions. *Biol Cybern*, 55:71–82, 1986.

[69] M. Clark and A. Bovik. Texture segmentation using gabor modulation/demodulation. *Pattern Recognition Letters*, 6:261–267, 1987.

[70] N.N. Kachouie and J. Alirezaie. Optimized multichannel filter bank with flat frequency response for texture segmentation. *Journal on Applied Signal Processing*, 12:1834–1844, 2005.

[71] D.A. Clausi and M.E. Jernigan. Designing gabor filters for optimal texture separability. *Pattern Recognition*, 33(11):1835–1849, 2000.

[72] P. Burt. Fast filter transforms for image processing. *Computer Graphics and Image Processing*, 16:20–51, 1981.

[73] P. Burt and E.H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2:217–236, 1983.

[74] J.M. Ogden, E.H. Adelson, J.R. Bergen, and P. Burt. Pyramid-based computer graphics. *RCA Engineer*, 30:4–15, 1985.

[75] P. Perona. Deformable kernels for early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[76] E.P. Simoncelli, W.T. Freeman, E.H. Adelson, and D.J. Heeger. Shiftable multiscale transforms. *IEEE Transactions on Information Theory, Special Issue on Wavelets*, 38:587–607, 1992.

[77] R. von der Heydt, E. Peterhans, and M.R. Dürsteler. Periodic-pattern-selective cells in monkey visual cortex. *Journal of Neuroscience*, 12(4):1416–1434, 1992.

[78] From natural to artificial neural computation, international workshop on artificial neural networks, iwann '95, malaga-torremolinos, spain, june 7-9, 1995, proceedings. volume 930 of *Lecture Notes in Computer Science*. Springer, 1995.

[79] D.A. Pollen and S.F. Ronner. Visual cortical neurons as localized spatial frequency filters. *IEEE Trans. Systs. Man. and Cybern.*, 13(5):907–916, 1983.

[80] J. Jones and A. Palmer. An evaluation of the two dimensional gabor filter model of simple receptive fields in cat striate cortex. *J. of Neurophysiology*, (58):1233–1258, 1987.

[81] P. Kruizinga and N. Petkov. Nonlinear operator for blob texture segmentation. In A.S. Cetin, editor, *Proceedings of the IEEE Workshop on Nonlinear Signal Processing*, volume 2, pages 881–885, 1999.

[82] N. Petkov and P. Kruizinga. Computational models of visual neurons specialised in the detection of periodic and aperiodic oriented visual stimuli: Bar and grating cells. *Biological Cybernetics*, 76:83–96, 1997.

[83] R. Bracewell. *The Fourier Transform and Its Applications*. 1986.

[84] C.F. Hall and E.L. Hall. A nonlinear model for the spatial characteristics of the human visual system. *IEEE Trans. Systems Man. Cybern.*, 7(3):161–170, 1977.

[85] G. Zhang, Z.M. Ma, Z. Cai, and H. Wang. Texture analysis using modified computational model of grating cells in content-based medical image retrieval. pages 125–132, 2008.

[86] S. Grossberg. How does the cerebral cortex work? learning, attention and grouping by the laminar circuits of visual cortex. 12(2):163–186, 1999.

[87] J.M.H. du Buf. Improved grating and bar cell models in cortical area v1 and texture coding. *Image and Vision Computing*, 25(6):873–882, 2007.

[88] T. Lourens, K. Nakadai, H.G. Okuno, and H. Kitano. A computational model of monkey grating cells for oriented repetitive alternating patterns. In *Proceedings of the ESANN 2001, 9th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 25-27, 2001*, pages 315–322, 2001.

[89] T. Lourens, E. Barakova, H.G. Okuno, and H. Tsujino. A computational model of monkey cortical grating cells. *Biological Cybernetics*, 92(1):61–70, 2005.

[90] S.B. Laughlin. A simple coding procedure enhances a neuron's information capacity. *Z. Naturforsch.*, 9-10(36):910–912, 1981.

[91] A.K. Jain and F. Farrokhnia. Unsupervised texture segmentation using gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

[92] T.Chang and C.C.J. Kuo. Texture analysis and classification with tree-structured wavelet transform. *IEEE Transactions on Image Processing*, 2(4):429–441.

[93] T.S. Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1–13, 1996.

[94] M.A. Garcia and D. Puig. Supervised texture classification by integration of multiple texture methods and evaluation windows. *Image and Vision Computing*, 25(7):1091–1106, 2007.

[95] R.R. Adhami J.F. Khan and S.M.A. Bhuiyan. A customized gabor filter for unsupervised color image segmentation. *Image and Vision Computing*, 27(4):489–501, 2009.

[96] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[97] B.E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classiffers. In *The Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press.

[98] J.A. Hartigan. *Clustering Algorithms*. Wiley, 1975.

[99] D. Pelleg and A. Moore. X-means: extending k-means with efficient estimation of the number of clusters. In *In Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 727–734, Stanford, California, 2000.

[100] T. Ishioka. Extended k-means with an efficient estimation of the number of clusters. In *Proceedings of the Second International Conference on Intelligent Data Engineering and Automated Learning (IDEAL2000)*, pages 17–22, Hong Kong, PR China, 2000.

[101] G.E. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[102] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[103] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 79(9), September 1990.

[104] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas. Self-organizing map in matlab: the som toolbox. In *Proceedings of the Matlab DSP Conference*, pages 35–40, 2000.

[105] T. Ojala, K. Valkealahti, E. Oja, and M.M. Pietikäinen. Texture discrimination with multidimensional distributions of signed gray level differences. *Pattern Recognition*, 34(3):727–739, 2001.

[106] G. Smith and I. Burns. Meastex image texture database and test suite, version 1.1. Available on `http://www.texturesynthesis.com/meastex/meastex.html`.

[107] R.W. Picard and T.P. Minka. Vision texture for annotation. *Multimedia Systems*, 3(1):3–14, 1995.

[108] D.K. Iakovidis, E. Keramidas, and D. Maroulis. Fuzzy local binary patterns for ultrasound texture characterization. In *Proc. Image Analysis and Recognition, 5th International Conference (ICIAR 2008)*, volume 5112 of *Lecture Notes in Computer Science*, pages 750–759. Springer, 2008.

[109] V. Murino, C. Ottonello, and S. Pagnan. Noisy texture classification: A higher-order statistics approach. *Pattern Recognition*, 31(4):383–393, 1998.

[110] S.R. Fountain and T.N. Tan. Extraction of noise robust rotation invariant texture features via multichannel filtering. In *Proceedings of the 1997 International Conference on Image Processing*, volume 3, pages 197–200, 1997.

[111] S.V. Aschkenasy, J. Muntwyler, B. van der Loo, E. Oechslin, and R. Jenni. Texture analysis in digitally-acquired echocardiographic images: The effect of jpeg compression and video storage. 31(3):361–366, 2005.

[112] Independent jpeg group. `http://www.ijg.org/`.

[113] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The qbic system. *Computer*, 28:23–32, 1995.

[114] S. Mehrotra, Y. Rui, M. Ortega-Binderberger, and T.S. Huang. Supporting content-based queries over images in mars. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pages 632–633, 1997.

[115] J. Laaksonen, M. Koskela, and E. Oja. Picsom - self-organizing image retrieval with mpeg-7 content descriptors. *IEEE Transactions on Neural Networks*, 13(4):841–853, 2002.

[116] R. W. Picard. Digital libraries: Meeting place for low-level and high-level vision. In *ACCV '95: Invited Session Papers from the Second Asian Conference on Computer Vision*, pages 3–12, London, UK, 1996. Springer-Verlag.

[117] M. Szummer and R.W. Picard. Indoor-outdoor image classification. In *CAIVD '98: Proceedings of the 1998 International Workshop on Content-Based Access of Image and Video Databases (CAIVD '98)*, page 42, Washington, DC, USA, 1998. IEEE Computer Society.

[118] A. Vailaya, A. Jain, and H. Zhang. On image classification: City versus landscape. *Pattern Recognition*, 31:1921–1936, 1998.

[119] P. Duygulu, K. Barnard, J.F.G. Freitas, and D.A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 97–112, London, UK, 2002. Springer-Verlag.

[120] S. Feng and R. Manmatha. A discrete direct retrieval model for image and video retrieval. In *CIVR '08: Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 427–436, New York, NY, USA, 2008. ACM.

[121] L. Zhu, A. Zhang, A. Rao, and R. Srihari. Keyblock: An approach for content-based image retrieval. pages 157–166, 2000.

[122] W. Wang, Y. Song, and A. Zhang. Semantics retrieval by content and context of image regions. In *Proceedings of the 15th International Conference on Vision Interface*, pages 17–24, 2002.

[123] J. Li. and J.Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1075–1088, September 2003.

[124] D. Depalov, T.N. Pappas, D.G. Li, and B. Gandhi. Perceptual feature selection for semantic image classification. In *ICIP06*, pages 2921–2924, 2006.

[125] J. Li J. Yuan and B. Zhang. Exploiting spatial context constraints for automatic image region annotation. In *6th ACM International Conference on Multimedia*, pages 595–604, 2007.

[126] M. Turtinen and M. Pietikinen. Visual training and classification of textured scene images. In *Proc. The 3rd International Workshop on Texture Analysis and Synthesis*, pages 101–106, 2003.

[127] T. Athanasiadis, P. Mylonas, Y.S. Avrithis, and S.D. Kollias. Semantic image segmentation and object labeling. *CirSysVideo*, 17(3):298–312, March 2007.

[128] J.P. Schober, T. Hermes, and O. Herzog. Content-based image retrieval by ontology-object recognition. In *The KI-Workshop on Applications of Description Logics (ADL-2004)*, 2004.

[129] Y. Liu, D. Zhang, G. Lu, and W. Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, 2007.

[130] F. Heitger, L. Rosenthaler, R. von der Heydt, and O. Kübler E. Peterhans. Simulation of neural contour mechanisms: from simple to end-stopped cells. *Image and Vision Computing*, 32(5):963–981, 1992.

[131] P. Kruizinga and N. Petkov. Computational model of dot-pattern selective cells. *Biol. Cybern.*, 83:313–325, 2000.

[132] J. Rodrigues and J.M.H. du Buf. Visual cortex frontend: Integrating lines, edges, keypoints and disparity. In *International Conference on Image Analysis and Recognition*, pages 664–671, 2004.

[133] F. Heitger, R. von der Heydt, E. Peterhans, L. Rosenthaler, and O. Kübler. Simulation of neural contour mechanisms: representing anomalous contours. *Image and Vision Computing*, 16(6-7):407421, 1998.

[134] Contour detection based on nonclassical receptive field inhibition. *IEEE Transactions on In Image Processing*, 12(7):729–739, 2003.

[135] N. Voss and B. Mertsching. Design and implementation of an accelerated gabor filter bank using parallel hardware. In *Proceedings of the 11th International Conference on Field-Programmable Logic and Applications*, volume 2147 of *Lecture Notes in Computer Science*, pages 451–460. Springer, 2001.

[136] J. Futrelle and J.S. Downie. Interdisciplinary research issues in music information retrieval: Ismir 2000-2002. *Journal of New Music Research*, 32(2):121–131, 2003.

[137] W.M. Hartmann. Pitch, periodicity, and auditory organization. 100(6):3491–3502, 1996.

[138] J Machlis and K. Forney. *The Enjoyment of Music*. W.W. Norton, 10 edition, 2007.

[139] D. Gerhard. *Computer Music Analysis*. School of Computing Science, Simon Fraser University, Canada, 1998.

[140] C. Krumhansl and E. Kessler. Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological Review9*, pages 334–368, 1982.

[141] D. Moelants and L. Van Noorden. Resonance in the perception tempo. *Journal of New Music Research*, 28:4366, 1999.

[142] R. Francès. *La Perception de la Musique (The Perception of Music)*. Erlbaum, Hillsdale, New Jersey, 1958. W.J. Dowling translation, 1988.

[143] W.J. Dowling. Scale and contour: Two components of a theory of memory for melodies. *Psychological Review*, 85(4):341–354, 1978.

[144] W.J. Dowling and D.S. Fujitani. Contour, interval, and pitch recognition in memory for melodies. *J. Acoust. Soc Am.*, 49(2), 1971.

[145] D. Deutsch, editor. *The Pyschology of music*. Academic Press inc, 1982.

[146] R. Van Egmond and D.J. Povel. Perceived similarity of exact and inexact transpositions. *Acta Psychologica*, 92:283–295, 1996.

[147] J. Edworthy. Interval and contour in melody processing. *Music Perception*, 2:375–388, 1985.

[148] Y.E. Kim, W. Chai, R. Garcia, and B. Vercoe. Analysis of a contour-based rep-resentation for melody. In *Proceedings of the International Symposium on Music Information Retrieval ISMIR2000*, 2000.

[149] F. Dorritie. *Essentials of Music for Audio Professionals*. Artistpro, 2000.

[150] M. Ryynänen. *Automatic Transcription of Pitch Content in Music and Selected Applications*. PhD thesis, Tampere University of Technology, 2008.

[151] R.P. Paiva. *Melody Detection in Polyphonic Audio*. PhD thesis, University of Coimbra, Portugal, 2006.

[152] D.J. Levitin. Memory for musical attributes. In P.R. Cook, editor, *Music, Cognition and Computerized Sound*, pages 214–215. MIT Press, 1999.

[153] J. Yudkin. *Understanding Music*. Prentice Hall, Upper Saddle River, NJ, 2 edition, 1999.

[154] D. Huron. Music information processing using the humdrum toolkit:concepts, examples, and lessons. *Computer Music Journal*, 26(2):11–26, 2002.

[155] J.C.R. Licklider. Auditory frequency analysis. In C. Cherry, editor, *Information theory*, pages 253–268. Butterworth, London, 1956.

[156] J.C.R. Licklider. Three auditory theories. In S. Koch, editor, *Psychology, a study of a science*, pages 41–144. McGraw-Hill, New York, 1959.

[157] J.C.R. Licklider. Periodicity pitch and related auditory process models. *International Audiology*, pages 11–36, 1962.

[158] R.F. Lyon. Computational models of neural auditory processing. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1–4, 1984.

[159] L. van Noorden. Two channel pitch perception. In M. Clynes, editor, *Music, mind, and brain*, pages 251–269. Plenum press, London, 1982.

[160] M. Slaney and R. Lyon. A perceptual pitch detector. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 357–360, 1990.

[161] R. Meddis and M. J. Hewitt. Virtual pitch and phase sensitivity of a computer model of the auditory periphery i: pitch identification. *J. Acoust. Soc. Amer.*, 89(6):2866–2882, 1991.

[162] S.W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997.

[163] F.J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[164] A.M. Noll. Cepstrum pitch detection. *J. Acoust. Soc. Am.*, 41(2):293309, 1967.

[165] B. Gold. and L. Rabiner. Parallel processing techniques for estimating pitch periods of speech in the time domain. *Journal of the Acoustical Society of America*, 46(2):442–448, 1969.

[166] A. Lahat, R.J. Niederjohn, and D.A. Krubsack. A spectral autocorrelation method for measurement of the fundamental frequency of noise-corrupted speech. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(6):741–750, 1987.

[167] B. Doval and X. Rodet. Estimation of fundamental frequency of musical sound signals. In *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP'91*, pages 3657–3660, 1991.

[168] R.C. Maher and J.W. Beauchamp. Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *Journal of the Acoustical Society of America*, 95(4):2254–2263, 1993.

[169] D. Talkin. A robust algorithm for pitch tracking. In W. B. Kleijn and K. K. Paliwal, editors, *Speech Coding and Synthesis*. Elseview Science B.V., 1995.

[170] A. de Cheveigné and H. Kawahara. Yin, a fundamental frequency estimator for speech and music. *Journal of the Acoustic Society of America*, 111(4):1917–1930, 2002.

[171] L. Clarisse, J.P. Martens, M. Lesaffre, B. De Baets, H. De Meyer, and M. Leman. An auditory model based transcriber of singing sequences. In *International Conference on Music Information Retrieval, ISMIR'2002*, pages 16–20, 2002.

[172] A.P. Klapuri. *Signal Processing Methods for the Automatic Transcription of Music*. PhD thesis, Tampere University of Technology, Finland, 2004.

[173] J. C. Brown and B. Zhang. Musical frequency tracking using the methods of conventional and "narrowed" autocorrelation. *J. Acoust. Soc. Amer.*, 89(5):23462354, 1991.

[174] N. Kunieda, T. Shimamura, and J. Suzuki. Robust method of measurement of fundamental frequency by aclos: autocorrelation of log spectrum. In *IEEE International Conf. on Acoust., Speech, and Signal Processing, ICASSP*, pages 232–235, 1996.

[175] L. Van Immerseel and J.P. Martens. Pitch and voiced/unvoiced determination with an auditory model. *J. Acoust. Soc. Am.*, 91:3511–3526, 1992.

[176] M. Slaney and R.F. Lyon. On the importance of time - a temporal representation of sound. In M. Cooke, S. Beet, and M. Crawford, editors, *Visual Representations of Speech Signals*. John Wiley & Sons, 1993.

[177] M. Piszczalski and B.A. Galler. Automatic music transcription. *Computer Music Journal*, (4):24–31, 1977.

[178] M. Goto. Music scene description: A predominant-f0 estimation method for detecting melody and bass lines. In *The Seventh Meeting of Special Interest Group on AI Challenges, SIG-Challenge-9907-8*, pages 45–52, 1999.

[179] M. Goto. A predominant-f0 estimation method for cd recordings: Map estimation using em algorithm for adaptive tone models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP 2001*, pages 3365–3368, 2001.

[180] M. Goto. A real-time music scene description system: Predominant-f0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication (ISCA Journal)*, 43(4):311–329, 2004.

[181] M. Marolt. On finding melodic lines in audio recordings. In *International Conference on Digital Audio Effects, DAFx'04*, 2004.

[182] J. Eggink and G.J. Brown. Extracting melody lines from complex audio. In *International Conference on Music Information Retrieval, ISMIR'2004*, 2004.

[183] R.P. Paiva, Teresa T. Mendes, and A. Cardoso. Melody detection in polyphonic musical signals: Exploiting perceptual rules, note salience, and melodic smoothness. *Comput. Music J.*, 30(4):80–98, 2006.

[184] X. Serra. Musical sound modeling with sinusoids plus noise. In C. Roads, S. Pope, A. Picialli, and G. De Poli, editors, *Musical Signal Processing*. 1997.

[185] K. Dressler. Extraction of the melody pitch contour from polyphonic audio. In *Music Information Retrieval Exchange MIREX'2005*, 2005.

[186] K. Dressler. Audio melody extraction for mirex 2009. In *Music Information Retrieval Exchange MIREX'2009*, 2009.

[187] J. Seokhwan and D.Y. Chang. Melody extraction from polyphonic audio based on particle filter. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[188] B. Ong, X. Serra, S. Streich, N. Wack, P. Cano, E. Gomez, F. Gouyon, and P. Herrera. Ismir 2004 audio description contest. Technical report, 2006.

[189] S. Joo, S. Jo, and C.D. Yoo. Audio melody extraction for mirex from polyphonic audio signal. In *Music Information Retrieval Exchange MIREX'2009*, 2009.

[190] G. Poliner and D. Ellis. A classification approach to melody transcription. In *International Conference on Music Information Retrieval ISMIR'2005*, 2005.

[191] J.C.R. Licklider. A duplex theory of pitch perception. Dowden, Hutchinson and Ross, Inc, Stroudsburg, PA, 1979.

[192] R.F. Lyon. A computational model of filtering, detection and compression in the cochlea. In *IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP'82*, pages 1282–1285, 1982.

[193] M. Slaney. Auditory toolbox: A matlab toolbox for auditory modeling work (version 2). Technical Report 1998-010, Interval Research Corporation, Palo Alto, USA, 1998.

[194] A.S. Shihab, S. Naiming, and G. Preetham. Stereausis: Binaural processing without neural delays. *Journal of the Acoustical Society of America*, 86(3):989–1006, 1989.

[195] E. Terhardt, G. Stoll, and M. Seewann. Algorithm for extraction of pitch and pitch salience for complex tonal signals. *J. Acoust. Soc. Am.*, 71:679–688, 1982.

[196] J. W. Cooley and J. W. Tukey. An algorithm for the machine computation of the complex fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[197] N. Reid and D. A. S. Fraser. Mean log-likelihood and higher-order approximations, 2010.

[198] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Stat. Soc. B*, 39(1):138, 1977.

[199] T. De Mulder, J.P. Martens, M. Lesaffre, M. Leman, B. De Baets, and H. De Meyer. An auditory model based transcriber of vocal queries conference. In *The Fourth International Conference on Music Information Retrieval, ISMIR'2003*, pages 245–246, 2003.

[200] F. Zheng, G. Zhang, and Z. Son. Comparison of different implementations of mfcc. *J. Computer Science & Technology*, 16(6):582–589, 2001.

[201] S.Z.K. Khine, L.N. Tin, and L. Haizhou. Singing voice detection in pop songs using cotraining algorithm. In *ICASSP'2008*, 2008.

[202] T.L. Nwe, A. Shenoy, and Y. Wang. Singing voice detection in popular music. In *ACM MULTIMEDIA*, 2004.

[203] R. Nóbrega and S. Cavaco. Detecting key features in popular music: case study - singing voice detection. In R. Ramirez, D. Conklin, and C. Anagnostopoulou, editors, *Workshop on Machine Learning and Music of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 7–12, Bled, Slovenia, 2009.

[204] Y. Li and D. Wang. Separation of singing voice from music accompaniment for monaural recordings. *IEEE Transactions on Audio, Speech, and Language Processing*, 2007.

[205] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno. Instrument identification in polyphonic music: Feature weighting to minimize influence of sound overlaps. *EURASIP Journal on Advances in Signal Processing*, 51979, 2007.

[206] P. Vos. Tonal induction. In M. Leman, editor, *Special issue of Music Perception*, volume 17, page 401544. 2000.

[207] C. Krumhansl. *Cognitive Foundations of Musical Pitch*. Oxford University Press, New York, 1990.

[208] P. Janata, J.L. Birk, J.D. Van Horn, M. Leman, B. Tillmann, and J.J. Bharucha. The cortical topography of tonal structures underlying western music. *Science*, 298(5601):2167–2170, 12 2002.

[209] M. Leman. *Music and Schema Theory, Cognitive Foundations of Systematic Musicology*. Springer-Verlag, 1995.

[210] M. Leman, L.P. Clarisse, B. De Baets, H. De Meyer, M. Lesaffre, G. Martens, J.P. Martens, and D. Van Steelant. Tendencies, perspectives, and opportunities of musical audio-mining. In *Proceedings of the 3rd EAA European Congress on Acoustics*, 2002.

[211] M.P. Kastner and R.G. Crowder. Perception of the major/minor distinction: Iv. emotional connations in young children. *Music Perception*, 8(2):189–202, 1990.

[212] H.C. Longuet-Higgens and M.J. Steedman. On interpreting bach. *Machine Intelligence*, 6:221–241, 1971.

[213] S.R. Holtzman. A program for key determination. *Interface*, 6:2956, 1970.

[214] P. Vos and E.W. Van Geenen. A parallel-processing key-finding music. *Music Perception*, 14:185–223, 1996.

[215] I. Shmulevich and J. Coyle. The use of recursive median filters for establishing the tonal context in music. 1997.

[216] I. Shmulevich and O. Yli-Harja. Localized key finding: Algorithms and applications. *Music Perception*, 17:531–544, 2000.

[217] I. Shmulevich, O. Yli-Harja, E. Coyle, and K. Lemström D.-J. Povel. Perceptual issues in music pattern recognition: complexity of rhythm and key finding. *Computers and the Humanities*, 35:23–35, 2001.

[218] M. Leman. An auditory model of the role of short-term memory in probe-tone ratings. *Music Perception*, 17:435–464, 2000.

[219] O. Izmirli. Tonal similarity from audio using a template based attractor model. 2005.

[220] S. Pauws. Extracting the key from music. *Intelligent Algorithms in Ambient and Biomedical Computing*, pages 119–132, 2006.

[221] T. Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. In *International Computer Music Conference*, pages 464–467, 1999.

[222] Y.W. Zhu and M.S. Kankanhalli. Precise pitch profile feature extraction from musical audio for key detection. *IEEE Transactions on Multimedia*, 8(3), 2006.

[223] E. Gómez and P. Herrera. Estimating the tonality of polyphonic audio files cognitive versus machine learning modelling strategies. In *International Symposium on Music Information Retrieval*.

[224] J.P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *6th International Conference on Music Information Retrieval (ISMIR'2005)*, pages 304–311, London, 2005.

[225] A. Shenoy and Y. Wang. Key, chord, and rhythm tracking of popular music recordings. *Computer Music Journal*, 29(3):75–86, 2005.

[226] Benoit Catteau, Jean-Pierre Martens, and Marc Leman. A probabilistic framework for audio-based tonal key and chord recognition. In *Advances in Data Analysis*, pages 637–644, 2006.

[227] P. Herrera, A. Yeterian, and F. Gouyon. Automatic classification of drum sounds: A comparison of feature selection methods and classification techniques. In *ICMAI*, pages 69–80, 2002.

[228] J.T. Foote. A similarity measure for automatic audio classification. In *The AAAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video and Audio Corpora*, Standford, 1997.

[229] K. Jensen and K. Arnspang. Binary decision tree classification of musical sounds. In *The 1999 ICMC*, 1999.

[230] G. Tzanetakis, P. Cook, and G. Essl. Automatic musical genre classification of audio signals. In *The International Symposium for Audio Information Retrieval*, pages 205–210, 2001.

[231] D. Pye. Content-based methods for the management of digital music. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2437–2440, 2000.

[232] M. Liu, C. Wan, and L. Wang. Content-based audio classification and retrieval using a fuzzy logic system: towards multimedia search engines. *Soft Computing*, 6:357–364, 2002.

[233] M. Leman, M. Lesaffre, and K. Tanghe. An introduction to the ipem toolbox for perception-based music analysis. *Mikropolyphonie - The Online Contemporary Music Journal*, 7, 2001.

[234] R. Shepard. Circularity in judgements of relative pitch. *J. Acoust. Soc. Amer.*, 36:2346–2353, 1964.

[235] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression trees*. Wadsworth Int. Group, Belmont, California, USA, 1984.

[236] *CART Decision Tree Software*. San Diego, CA, USA, 1998. Available on `http://www.salford-systems.com`.

[237] J.R. Smith and P. Schirling. Metadata standards roundup. *IEEE Multimedia*, 13(2):84–88, 2006.

[238] A. Pentland, R.W. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. 18(3):233–254, 1996.

[239] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:1026–1038, 1999.

[240] Resource description framework (rdf):concepts and abstract syntax.

[241] D. Brickley, R.V. Guha, and B. McBride. Rdf vocabulary description language 1.0: Rdf schema, 2004. Available on: `http://www.w3.org/TR/rdf-schema`.

[242] B. Motik, P.F. Patel-Schneider, and B. Parsia. Owl 2 web ontology language structural specification and functional-style syntax, 2009. Available on: `http://www.w3.org/TR/owl2-syntax`.

[243] C. Golbreich and E.K. Wallace. Owl 2 web ontology language new features and rationale, 2009. Available on: `http://www.w3.org/TR/owl-new-features`.

[244] Japan Electronics and Information Technology Industries Association. Exchangeable image file format for digital still cameras: Exif version 2.2, 2002. Available on `http://www.exif.org/Exif2-2.pdf`.

[245] M. Kanzaki. The kanzaki exif rdf schema, 2003. Available on `http://www.kanzaki.com/ns/exif`.

[246] The norm walsh exif rdf schema, 2003. Available on= `http://sourceforge.net/projects/jpegrdf`.

[247] Dublin Core Metadata Initiative. Dublin core metadata element set, version 1.1: Reference description, 2004. Available at: `http://www.dublincore.org/documents/dces`.

[248] Describing and Retrieving Photos using RDF and HTTP, W3C Note, 2002. Available on `http://www.w3.org/TR/photo-rdf/`.

[249] E. Grosso, H. Eriksson, R. Fergerson, S. Tu, and M. Musen. Knowledge modeling at the millennium the design and evolution of protégé-2000. In *Proceedings of KAW-99*, 1999.

[250] Mpeg-7 overview, international organization for standardisation, klagenfurt iso/iec jtc1/sc29/wg11, july 2002.

[251] R. Arndt, R. Troncy, S. Staab, and M. Vacura L. Hardman. COMM: Designing a Well-Founded Multimedia Ontology for the Web. In *Lecture Notes of Computer Science: The Semantic Web*, volume 4825, pages 30–43, 2007.

[252] J. Hunter. Adding Multimedia to the Semantic Web - Building an 7 ontology. In *Proceedings of the First Semantic Web Working Symposium (SWWS)*, pages 261–281, 2001.

[253] I. Horrocks, F. van Harmelen, P. Patel-Schneider, T. Berners-Lee, D. Brickley, D. Connolly, M. Dean, S. Decker, D. Fensel, P. Hayes, J. Heflin, J. Hendler, O. Lassila, D. McGuinness, and L. Stein. Daml+oil, 2001. Available on: http://www.daml.org/2001/03/daml+oil-index.

[254] C. Tsinaraki, P. Polydoros, and S. Christodoulakis. Interoperability support for ontology-based video retrieval applications. In *The 3rd International Conference on Image and Video Retrieval (CIVR 2004)*, pages 21–23, 2004.

[255] R. Garcia and O. Celma. Semantic integration and retrieval of multimedia metadata. In *The 5th International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2005)*, 2005.

[256] R. Arndt, R. Troncy, S. Staab, L. Hardman, and M. Vacura. Comm: Designing a well-founded multimedia ontology for the web. In *The 6th International Semantic Web Conference (ISWC 2007)*, 2007.

[257] Photo metadata, white paper, international press telecommunications council (iptc).

[258] H. Löffler and W. Baranger. Photo metadata white paper, iptc, 2007. Available on: http://www.iptc.org/goto?phmdwp2007.

[259] Visual resources association data standards committee, vra core categories, version 4.0. Available on: http://www.vraweb.org/projects/vracore4.

[260] Digital Imaging Group (DIG). Dig35 specification - metadata for digital images - version 1.1, 2001. Available on: http://www.i3a.org/technologies/metadata.

[261] G. Martens and C. Poppe. Dig35 owl dl ontology, 2007. Available on: http://multimedialab.elis.ugent.be/ontologies/DIG35/v1.0.

[262] Zope, noteAvailable on: http://www.zope.org.

[263] Zope object database. Available on: http://pypi.python.org/pypi/ZODB3/3.8.1.

[264] Python programming language. Available on: http://www.python.org.

[265] Plone cms: Open source content management. Available on: http://plone.org.

[266] Drupal. Available on: `http://drupal.org`.

[267] Php: Hypertext preprocessor. Available on: `http://php.net/index.php`.

[268] Joomla. Available on: `http://www.joomla.org`.

[269] Exponent cms: Open source content management system. Available on: `http://www.exponentcms.org/`.

[270] Exif module for drupal. Available on: `http://drupal.org/project/exif`.

[271] Iptc module for drupal. Available on: `http://drupal.org/project/iptc`.

[272] Iptc module for plone. Available on: `http://plone.org/products/iptc`.

[273] Fedora. Available on: `http://www.fedora-commons.org/`.

[274] BRICKS - Building Resources for Integrated Cultural Knowledge Services, 2007. Available on `http://www.brickscommunity.org/prj`.

[275] N. Sluijs, K. Vlaeminck, T. Wauters, B. Dhoedt, F. De Turck, and P. Demeester. Caching strategies for Personal Content Storage Grids. In *Proc. the International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 396–404, 2007.

[276] Sparql query language for rdf, w3c recommendation 15 january 2008. Available on `http://www.w3.org/TR/rdf-sparql-query/`.

[277] S. Bechhofer and R. Volz. Patching syntax in owl ontologies. In *International Semantic Web Conference*, pages 668–682, 2004.

[278] D. Brickley and L. Miller. Foaf vocabulary specification, 2006. Available on: http://xmlns.com/foaf/0.1/.

[279] A.J. Miles, N. Rogers, and D. Beckett. Skos core rdf vocabulary, 2004. Available on: http://www.w3.org/2004/02/skos/core.

[280] X. Wang, T. DeMartini, B. Wragg, M. Paramasivam, and C. Barlas. The mpeg-21 rights expression language and rights date dictionary. *IEEE Transactions on Multimedia, International Conference on Parallel and Distributed Processing Techniques and Applications*, 7:408–417, 2005.

[281] C. Poppe, I. Wolf, S. Wischnowsky, S. De Zutter, and R. Van de Walle. Licensing System in an online MPEG-21 Environment. In *Proceedings of the IADIS international conference WWW/internet 2006*, volume II, pages 315–319, Murcia, October 2006.

[282] D.L. Hull. The effect of essentialism on taxonomy, two thousand years of stasis. *Br. J. Phil. Sci*, 15:314–326, 1965.

[283] T. Berners-Lee, J. Hendler, and O. Lassila. *The Semantic Web*. Scientific American, 2001.

[284] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M.C.A. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The semantic web: The roles of xml and rdf. *IEEE Internet Computing*, 4(5):63–74, 2000.

[285] I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. Swrl: A semantic web rule language - combining owl and ruleml, w3c member submission, 21 may 2004. Available on `http://www.w3.org/Submission/SWRL/`.

[286] D.H. Hubel and T.N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.*, 148:574–591, 1959.

[287] M. Riesenhuber and T. Poggio. How the visual cortex recognizes objects: The tale of the standard model. *The Visual Neurosciences*, 2:1640–1653, 2003.

[288] E. Hering. Zur lehre vom lichtsinn, 1874. Vienna.

[289] J.D. Cutnell and W.J. Kenneth. *Physics*. Wiley, New York, 4 edition, 1998.

[290] W.M. Hartmann. *Signals, Sound and Sensation*. AIP Press, 1997.

[291] A. Simmons, A.N. Popper, and R. R. Fay, editors. *Acoustic Communication*. Handbook of Auditory Research. Springer, 2002.

[292] J. Syka and L.M. Aitkin. *Auditory pathways: struture and function*. Plenum, 1988.

[293] A. Abbott. Neurobiology: Music, maestro, please! *Nature*, 416(6876):12–14, 2002.