COMENIUS UNIVERSITY IN BRATISLAVA

FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

TEXTURE AWARE VIDEO AND IMAGE ERROR CONCEALMENT

DISSERTATION THESIS

2016 **RNDr. Ivana Ilčíková**

COMENIUS UNIVERSITY IN BRATISLAVA

FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

TEXTURE AWARE VIDEO AND IMAGE ERROR CONCEALMENT

DISSERTATION THESIS

Study Programme:	Geometry and Topology
Study Field:	1116 GEOMETRY AND TOPOLOGY
Department:	Department of Algebra, Geometry and Didactics of Mathematics
Supervisor:	prof. Ing. Jaroslav Polec, CSc.

Bratislava 2016 **RNDr. Ivana Ilčíková**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: RNDr. Ivana Ilčíková
Študijný program: geometria a topológia (Jednoodborové štúdium, doktorandské III. st., denná forma)
Študijný odbor: geometria a topológia
Typ záverečnej práce: dizertačná
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Texture Aware Video and Image Error Concealment
Maskovanie chýb vo videu a obraze zachovávajúce textúry

Literatúra:
[1] J.W. Woods. Multidimensional Signal, Image, and Video Processing and Coding. Elsevier Science, Second edition, 2012.
[2] Iain E. Richardson. The H.264 Advanced Video Compression Standard. John Wiley & Sons, Ltd, 2010.
[3] R. Szeliski, Computer vision: algorithms and applications, Springer Science & Business Media, 2010.

Cieľ: Videá a obrazy sú často prenášané s chybami v bezdrôtových sieťach, obzvlášť v oblastiach so slabým pokrytím signálu alebo na miestach s príliš veľkým počtom zariadení. Cieľom práce je navrhnúť metódu maskovania chýb, ktorá by zachovala textúrovaný vzhľad obrazu alebo video snímky v zamaskovaných poškodených regiónoch. Metóda by mala segmentovať oblasti v obraze na základe textúry, následne odhadne tvar chýbajúcej časti každého segmentu a nakoniec extrapoluje textúru každého segmentu do jeho poškodenej časti, aby ju zamaskovala.

Kľúčové slová: maskovanie chýb, fuzzy segmentácia, extrapolácia textúry, kódovanie videa a obrazu, extrakcia čŕt textúry

Školiteľ: prof. Ing. Jaroslav Polec, CSc.
Katedra: FMFI.KAGDM - Katedra algebry, geometrie a didaktiky matematiky
Vedúci katedry: prof. RNDr. Pavol Zlatoš, PhD.

Dátum zadania: 01.09.2012

Dátum schválenia: 16.02.2011

prof. RNDr. Július Korbaš, CSc.
garant študijného programu

.....
študent

.....
školiteľ

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	RNDr. Ivana Ilčíková
Study programme:	Geometry and Topology (Single degree study, Ph.D. III. deg., full time form)
Field of Study:	Geometry And Topology
Type of Thesis:	Dissertation thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Texture Aware Video and Image Error Concealment

Literature:
[1] J.W. Woods. Multidimensional Signal, Image, and Video Processing and Coding. Elsevier Science, Second edition, 2012.
[2] Iain E. Richardson. The H.264 Advanced Video Compression Standard. John Wiley & Sons, Ltd, 2010.
[3] R. Szeliski, Computer vision: algorithms and applications, Springer Science & Business Media, 2010.

Aim: Videos and images are frequently transferred with errors in wireless networks, especially in areas with poor signal coverage or in places with too many devices. The aim of the work is to propose an error concealment method that would preserve textured appearance of the image or video frame in concealed corrupted regions. The method should segment image regions based on their texture, then it estimates a shape of missing part of each segment and lastly it extrapolates the texture of each segment into its corrupted part to conceal it.

Keywords: error concealment, fuzzy segmentation, texture extrapolation, video and image coding, texture features extraction

Tutor:	prof. Ing. Jaroslav Polec, CSc.
Department:	FMFI.KAGDM - Department of Algebra, Geometry and Didactics of Mathematics
Head of department:	prof. RNDr. Pavol Zlatoš, PhD.
Assigned:	01.09.2012

Approved: 16.02.2011 prof. RNDr. Július Korbaš, CSc.
Guarantor of Study Programme

.....
Student

.....
Tutor

Acknowledgements

I would like to thank to prof. Ing. Jaroslav Polec, CSc. for all his help, time, support and priceless advices he has given me throughout my studies, the research and also the writing period of this thesis.

I am very grateful to my family and all my friends for their encouragement and loving support.

Declaration

I herewith declare that I produced this dissertation thesis without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources that have not been identified as such.

This document has not previously been presented in identical or similar form to any other Slovak or foreign examination board.

This research was conducted under the supervision of prof. Ing. Jaroslav Polec, CSc. at Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava, Slovakia from 2012 to 2016.

Bratislava, April 2016 ...……...

RNDr. Ivana Ilčíková

Abstract

Nowadays we are facing the transition from desktop PCs to mobile devices. We use them to watch TV, stream video from the Internet and to make video calls and video conferences. The video is compressed before being stored and transmitted, which makes it sensitive to errors. The errors frequently appear in wireless networks, especially in areas with poor signal coverage or in places with too many devices. Lost or corrupted packets cause subsequent macro blocks to be marked as corrupted by the decoder. The impact of transmission errors can be minimized in the receiver through *error concealment*.

Error concealment is an error control technique capable of mitigating the error effects on multimedia using all the available decoded data (both correctly received and erroneous). Recently published over-segmentation algorithms operating in real-time have opened new ways of error concealment for streamed videos. We introduce a fast error concealment technique where the corrupted regions are restored by texture extrapolation from the surrounding regions logically associated through image segmentation. Our method can faithfully complete the texture and edges in the missing areas. However, areas in the image containing unsharp edges or gradients, which are difficult to segment properly, are the main problem producing artifacts in the result. Therefore, in addition we propose an extended form of segmentation which adds soft edges to the segments and allows them to overlap. As result the unsharp edges and gradients are maintained in the concealed parts of the image.

In the second part of the thesis we address the image segmentation in more detail. The Final segmentation of an over-segmented image is usually obtained by merging neighboring regions based only on their color similarity. We propose a novel method that in addition classifies the image regions based on their texture features and we show that this improvement leads to better results.

Keywords: error concealment, fuzzy segmentation, texture extrapolation, video and image coding, texture features extraction

Abstrakt

V súčasnej dobe čelíme prechodu od stolových počítačov k mobilným zariadeniam. Používame ich na pozeranie televízie, streamujeme video z internetu a robíme video hovory a video konferencie. Video je pred uložením a prenosom komprimované a to ho robí citlivým na chyby. Chyby sa často objavujú v bezdrôtových sieťach, predovšetkým v oblastiach so slabým pokrytím signálu alebo na miestach s príliš veľkým počtom zariadení. Stratené alebo poškodené pakety spôsobia, že za sebou idúce makro bloky sú dekodérom označené ako poškodené. Vplyv prenosových chýb môže byť v prijímači minimalizovaný pomocou maskovania chýb.

Maskovanie chýb je technika regulácie chýb schopná zmierniť efekty chýb v multimédiách s využitím všetkých dostupných dekódovaných dát (správne prijatých a tiež poškodených dát). Nadsegmentačné algoritmy bežiace v reálnom čase otvorili cestu k ich využitiu v aplikáciách pre streamované videá. Predstavujeme rýchlu techniku maskovania chýb, v ktorej sú poškodené oblasti obnovené extrapoláciou textúry z okolitých oblastí logicky asociovaných pomocou segmentácie obrazu. Naša metóda dokáže verne doplniť textúry a hrany v chýbajúcich oblastiach. Avšak oblasti v obraze obsahujúce neostré hrany alebo prechody, ktoré je ťažké správne vysegmentovať, sú hlavnou príčinou vzniku artefaktov vo výsledku. Preto navyše navrhujeme rozšírenú formu segmentácie, ktorá segmentom pridáva jemné hrany a umožňuje im prekrývať sa. Výsledkom je zachovanie neostrých hrán a prechodov v zamaskovaných častiach obrazu.

V druhej časti práce sa hlbšie venujeme segmentácií obrazu. Finálna segmentácia z nadsegmentovaného obrazu sa zvyčajne získa zlúčením susedných oblastí len na základe ich farebnej podobnosti. Navrhujeme novú metódu, ktorá klasifikuje oblasti obrazu aj na základe čŕt ich textúr, čo vedie k lepším výsledkom.

Kľúčové slová: maskovanie chýb, fuzzy segmentácia, extrapolácia textúry, kódovanie videa a obrazu, extrakcia čŕt textúry

Contents

Chapter 1

Introduction

Communication plays an important role in our lives. This is as much true today as it has been in the past and will be in the future. After the language, visual communication is the most important way of communicating. Among many types of visual communication, the video is one of the most powerful ways of visual communication and it has become an integral part of the modern culture. In the last decade many video services and devices have changed and improved. Mobile phones are frequently used to capture video or to stream video from internet. Video calls and video conferences over the internet are very common. Digital television is widespread in the world in contrary to analogue television that has already been switched off in many countries. Quality of video services like video streaming or video calls is still variable but it continues to improve. In order to stream video data over channels with a limited bandwidth it is being compressed before storage and transmission, which makes it sensitive to errors. Transferring images and videos requires a fast and reliable transmission channel. However, in wireless networks we often encounter high error rates, especially in areas with poor signal coverage or in places with too many devices. Fortunately, various signal processing algorithms allow to detect and conceal such errors.

Image texture analysis is a fundamental problem in image processing and an important area in computer vision [26]. Indeed, textures are important parts of natural and architectural scenes including both man-made and natural objects which in turn makes them to essential attributes in visual content based image analysis. Bricks, stones, sand, leaves, or grass, are just a few examples of regular and irregular structures that make objects appear textured.

The term *texture* is widely used and easily comprehensible. However, our intuitive understanding of this phenomena is so strong, that no exact definition exists. Some of the technical definitions describe a texture as a "discrete 2D stochastic field with a given governing joint probability density function"[44] or a "repetitive arrangement of a unit pattern over a given

area"[52]. Human observers mostly describe a given texture by its qualitative attributes like smoothness or roughness, fineness or coarseness which are easily perceived by their senses. These properties are again intuitively clear, yet relative and hard to measure[60].

Automatic processing of images by their visual content has become an interesting and dynamic research area [26] in recent years. Its goal is to automatically process and analyze the image based on different features such as color, shape and texture. These features are then used to segment images into regions and to classify them.

The aim of the segmentation is to simplify an image into a more meaningful representation that is simpler to analyze [62, 55]. Image segmentation is typically used to locate objects and boundaries in images. Among the prominent practical applications of image segmentation are content-based image retrieval, machine vision and medical imaging for surgery planning and diagnostics. Recently published real-time over-segmentation algorithms [13, 48] opened ways to utilize segmentation in applications for streamed videos.

1.1 Mathematical Formulation of the Image and Video Error Concealment Problem

Error effects in transmitted image or video depend on the position of the error in the multimedia file and on the error resilience mode chosen to secure the information bits. The position of an error is a factor affecting the overall file validity. An error located in the header can cause a degradation prohibiting valid recognition of the multimedia file. However, due to a small amount of header bits relative to the information bits and due to the possibility of implementing an Automatic Repeat reQuest (ARQ) method to retransmit the corrupted bits, error control techniques commonly assume correct header reception and errors located exclusively in payload. The most commonly used error resilience modes for images and videos work with blocks. Therefore, erroneous or missing data immediately damages whole blocks or block clusters. We build upon these assumptions in the image and video error concealment problem formulation.

Mathematically, image and video error concealment is an ill-posed inverse problem since there is no well-defined unique solution. Let Ω be the corrupted part of an image I (see Fig. 1.1), then the error concealment problem can be defined as: $\forall p \in I$

$$\chi : \left| \begin{array}{c} \Omega \subset R^n \rightarrow R^m \\ p \mapsto I(p) \end{array} \right. \tag{1.1}$$

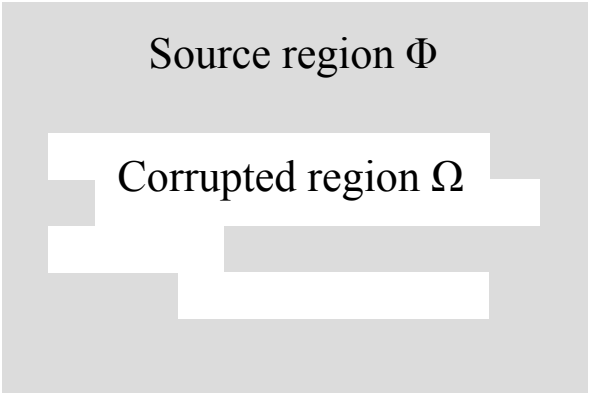


Figure 1.1: An example of regions in a corrupted image.

where $m = 3$ for *RGB* color images and n refers to the data dimension, i.e. $n = 2$ ($p = (x, y)$) for image pixels, respectively $n = 3$ and $p = (x, y, t)$ for videos. The known region named the *source region* is then defined as $\Phi = I \backslash \Omega$.

1.2 Contributions

This thesis contains two main contributions that are new methods for:

- image error concealment
- texture classification of arbitrarily shaped regions with an application in image segment merging

We briefly describe our motivation for both methods and the methods themselves in following two sections.

1.2.1 Proposed Error Concealment Method

In this work we present a novel technique for concealment of corrupted image macroblocks utilizing basic principles of extrapolation from uncorrupted neighboring regions into the corrupted areas. Fast frequency selective extrapolation methods [54, 34] use a straight forward approach which extrapolates every pixel falling within the rectangular surroundings of the lost area. Such approximation of various textures leads to a mixture and in the resulting extrapolation it creates undesirable artifacts. Similarly, we often encounter artifacts when using simple patch-based methods [15, 16] which inpaint the corrupted area by copying patches

from the uncorrupted surroundings of the missing area. To avoid such artifacts we propose to add a segmentation step into the process of error concealment. By using one of the real time over-segmentation algorithms [48, 13] the added segmentation step has minimal effect on the computational time. Its high processing speed is very important for the algorithm intended for streamed video.

High quality image reconstruction can be achieved using e.g. complex patch-based method [17] or group-based method [84], but run times of these methods are several orders of magnitude longer compared to the mentioned fast methods. Our algorithm is focused on improvement of quality for fast image restoration in the time span of a few seconds.

Our method starts with image segmentation. Based on image segmentation, it selects only the most relevant uncorrupted pixels for the texture extrapolation. By dealing with each segmented region separately, we can avoid artifacts. However, the segmentation in the corrupted areas is missing and needs to be estimated first. We developed a new algorithm for completion of the segmentation that is based on shape error concealment techniques for single objects. Each corrupted segment is considered as an object the shape of which should be completed. Our key contribution is the extension of shape error concealment techniques for single objects to segmented images with each corrupted segment being a small object to conceal. Connectivity of neighboring segments and of segment pieces cut by the lost area into several pieces is the main challenge.

The main drawback of using the segmentation is a high level of uncertainty regarding the segment borders in image areas containing unsharp edges or gradients. Repaired segments consequently include unexpected sharp edges. Therefore, we propose an extension for our method that utilizes a fuzzy segmentation. Segments in the fuzzy segmentation include soft edges and are allowed to overlap. As result the soft edges and gradients are maintained in the concealed image.

The last step of our algorithm is the texture extrapolation into lost areas from their surrounding regions logically associated by the segmentation. First, we approximate the texture of the uncorrupted part of a segment using an orthogonal transform such as Discrete cosine transform or Discrete Hartley transform. We define basis functions of the orthogonal transform over the entire area of the segment and therefore each approximation of the uncorrupted part also provides an estimation of the missing texture in the corrupted part of the segment.

To assess the quality and speed of our concealment method we compared the results with three state of the art inpainting techniques: *Image Melding* [17], *Spatial Patch Blending* [16] and *Group-based Sparse Representation* [84]. Experimental results clearly demonstrate a qualitative improvement of the error concealment capability of our texture aware method

over the simple patch-based method that just copies patches from the neighborhood of the corrupted regions. The sophisticated patch and group-based methods are able to provide better results, but their run-times are not feasible in real-time applications such as video streaming.

1.2.2 Texture Classification of Arbitrarily Shaped Regions with an Application in Image Segment Merging

For the image over-segmentation in our error concealment method we use a fast superpixel generation algorithm based on k-means clustering in the CIE $L^*a^*b^*$ space extended by spatial pixel coordinates. To decide whether the neighboring superpixels should be merged (in order to create the resulting segments) the mean $L^*a^*b^*$ values are compared. However, it reflects only the color similarity of two superpixels. We propose to compare not just the color of superpixels, but also their texture. Most of the existing methods for extracting texture features are not suitable for arbitrarily shaped regions and they can only extract the features from a rectangle inscribed into the region. We developed a novel approach that is able to extract texture features from the whole irregular region using Discrete orthogonal transforms (DOTs). Orthogonal transforms applied to arbitrarily shaped superpixels produce spectral matrices with various dimensions. Before comparing spectral coefficients the dimensions have to be unified. We prove that by a specific insertion of zeros into the spectra of the Discrete Walsh-Hadamard transform or into the separable Discrete Hartley transform we can achieve a texture periodification in spatial domain.

To demonstrate the proposed method's functionality we provide results of the classification obtained for arbitrarily shaped regions in several images with artificially created textures and results with images containing textures from natural scenes. We also made several experiments with corrupted images in order to examine the method capability to be used in superpixel classification before they get merged into the final image segments. The results are extremely promising and suggest the proposed method has wide applicability in image processing.

1.3 Outline of the Thesis

The rest of the thesis is organized as follows. In next chapter we explain why the errors in videos and images appear and we also describe how they can be detected.

Chapter 3 starts with the state of the art report on the existing image error concealment methods. Based on limitations of existing methods we propose a new method that we describe in detail in the same chapter. In the end of chapter 3 we evaluate our method against the state of the art methods.

In the first part of chapter 4 we analyze existing methods for texture features extraction. Based on the analysis we introduce a new method for texture classification of arbitrarily shaped regions. We close the chapter by the evaluation of the proposed method and we also propose an application of our method in tasks of image segment merging.

Chapter 5 contains conclusions of the thesis and suggestions for future work.

Chapter 2

Video and Image Errors

In this chapter we explain why errors appear in videos and images. We also describe how they can be detected and treated.

2.1 Video and Image Compression

Video and image compression or encoding is the process of reducing the amount of data required to represent a digital video / image signal [50]. It is usually performed before the data is stored or transmitted. The complementary operation, decompression or decoding, recovers a digital video / image signal from its compressed representation. This process is usually performed before displaying the data. Raw digital video / image data is usually very large and therefore encoding and decoding are essential for any application in which storage capacity or transmission bandwidth is limited. Almost all common applications for digital video fall into this category. These are few examples of such applications (see also Fig. 2.1):

- *Digital television broadcasting:* TV programs are coded before their transmission over a limited bandwidth of satellite, cable or terrestrial channel.
- *Internet / mobile video streaming:* Video is coded and stored on a server. The coded video is streamed over the internet / mobile network (e.g. GSM, GPRS, EDGE), decoded on a client device and displayed.
- *Video on Digital Storage Media (DSM):* Source video is coded and stored on a DSM (e.g. Blu-ray Disc, DVD, Flash Disc). A digital media player reads the DSM and decodes video for display.

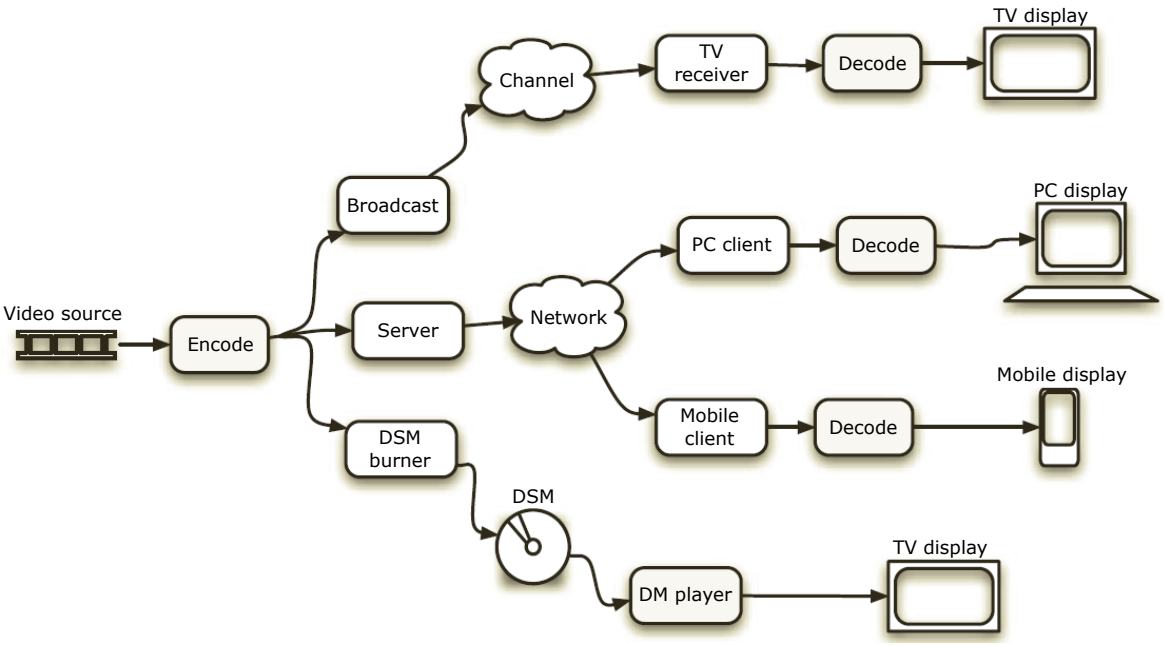


Figure 2.1: Video coding scenarios[50].

All examples above include an encoder, which encodes (compresses) an input video signal into a coded bitstream, and a decoder, which decodes (decompresses) the coded bitstream and produces the output video signal. The encoders are often built into the devices such as a video cameras. Decoders can be built in as well, for example into video players.

Data compression is achieved by removing redundancy, i.e. components that are not necessary for faithful reproduction of the data [51]. If some data contains *statistical* redundancy, it can be effectively compressed using lossless compression. By decoding data compressed losslessly we get the exact copy of the original data. However, lossless compression of image and video data gives very small amount of data reduction. Therefore, to reduce the amount of image and video data a lossy compression is commonly used. When using a lossy compression, after decoding the resulting data is not identical to the source data anymore. Higher compression ratios can be obtained at the expense of lower visual quality. Lossy compression systems are based on the principle of removing *subjective* redundancy [51], i.e. elements of the image or video sequence which can be discarded without significant decrease of the visual quality.

Image compression methods exploit spatial redundancy, video compression utilizes both spatial and temporal redundancy to obtain data reduction. In the temporal domain, there is often high correlation (similarity) between frames which were captured at around the same time. Temporally adjacent frames are usually highly correlated, in particular in videos with high frame rates. In the spatial domain, there is often high correlation between pixels that are close to each other. For illustration of both types of redundancy see Fig. 2.2.

Figure 2.2: Spatial and temporal correlation in two successive video frames [51].

Compressed videos and images are sensitive to transmission errors. In wireless networks we often encounter high error rates, especially in areas with poor signal coverage or in places with too many devices. Transmission errors such as bit errors or packet loss can cause large damage in decoded data. In compressed images and videos, where the spatial redundancy was removed, the errors can spatially propagate to neighboring pixels. And in compressed video, where the temporal redundancy was removed, the errors can propagate to the following frames. Spatial and temporal error propagation are illustrated in Fig. 2.3.

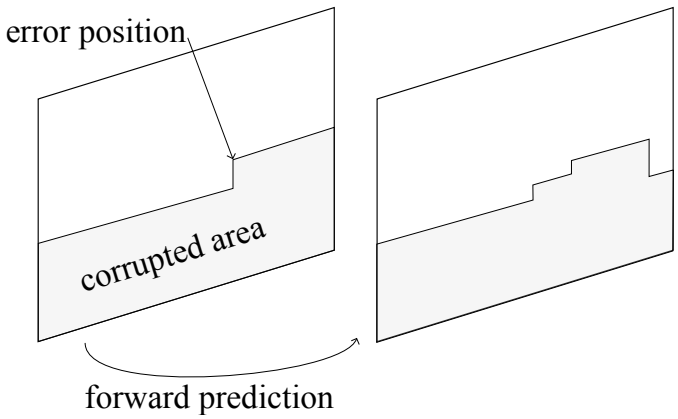


Figure 2.3: Spatial and temporal error propagation [51].

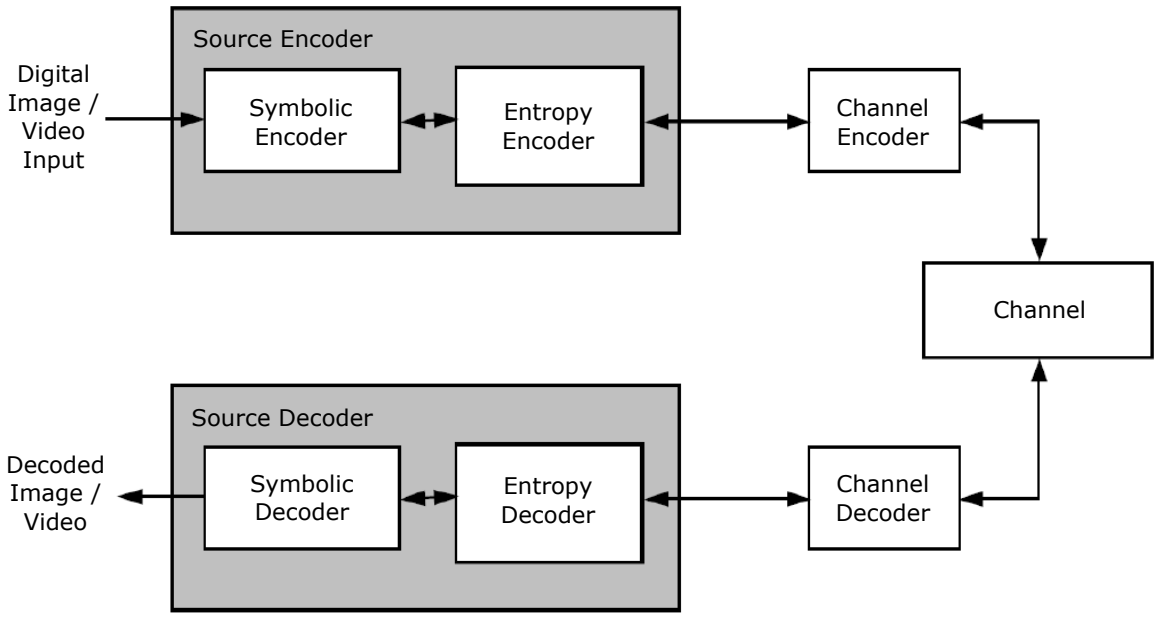


Figure 2.4: Simplified architecture of a video / image communication system [58, 79].

2.2 Error Resilient Video and Image Coding

A video / image communication system (see simplified architecture in Fig. 2.4) starts by compression of the input digital video / image data. The compression is performed by a *source encoder*, which contains *symbolic* and *entropy* encoders [58].

The *symbolic encoder* removes most of the redundancy that the input data has. For example, image can be transformed using some orthonormal transforms based on frequency decompositions like Discrete cosine transform (DCT) or Discrete Fourier transform (DFT) [79]. These transforms can be applied to the entire image, but usually they are applied to relatively small blocks of pixel values. The block transform is invertible and no data compression actually occurs there. The compression is performed by a quantizer that chooses representative values of the transformed data.

The *entropy encoder* converts the representative values (outputs of the quantizer) to efficient variable-length code words. Variable-length codes (VLC), such as Huffman codes or arithmetic codes, are able to significantly decrease the amount of data. On the other hand, if during the transmission over the channel some error occurs, VLC with a complete code tree (in the sense that all binary strings are possible, e.g. Huffman codes) can cause that the decoder will not notice the error and will continue to decode with errors. The solution for this is to insert markers such as end of block (EOB) and other synchronization words into the coded bitstream. Markers are used to terminate or at least to control the error propagation. They are added into the coded bitstream by the *channel encoder*.

After the data has been transmitted over the communication channel the inverse operations are executed in order to get the decoded video / image. Double sided arrows are used in Fig. 2.4 to show that the decoder can ask the encoder to retransmit some important failed data.

To transmit a compressed video or image over error-prone networks, error resilience techniques are needed. According to Soares [59] there are three main types of error resilience techniques:

1. *Error resilient source coding* – These techniques deal with the conversion of the input digital video / image into an efficient, resilient representation. More error resilient representation generally means less compression efficiency, but it helps to stop error propagation. The problem of these techniques is that they influence the coding syntax used by both encoder and decoder for their communication and have to be standardized. It generally means that after a standard is defined, these techniques can no longer be changed or updated.

2. *Channel coding and decoding* – We have already mentioned that channel encoder systematically inserts additional bits into the coded bitstream to make it possible to detect errors. Channel coding and decoding is mostly independent from source coding and decoding. But sometimes the joint source-channel coding is used. It is useful when we want to scale the level of protection for different types of data to be transmitted, e.g. motion vectors or texture data. Since channel coding clearly influences the communication syntax between the encoder and decoder, it has to be standardized as well. Usually it is not part of the video / image coding standards, but it is defined by transport standards.

3. *Error resilient decoding and concealment* – These techniques minimize the negative impact of transmission errors in the final video / image that will be displayed. For the error concealment all available decoded data (correctly received and also erroneous data) are used. Since this type of techniques does not depend on the coding syntax of the encoder and decoder, they do not have to be standardized and new improvements can be easily adopted in practice.

In our work we focus on the last type of error resilient techniques and we will describe them in more detail in the following sections.

2.2.1 Error Resilient Decoding and Concealment

The error resilient decoding and concealment includes all the techniques that allow the decoder to lower the negative effect of errors by utilizing the available corrupted and correctly

decoded data. The decoder generally goes through three successive steps [59]:

1. *Error detection* – recognizes if any errors occurred
2. *Error localization* – finds out with the highest possible precision, where the detected error(s) appeared
3. *Error concealment* – decreases the negative impact of the localized errors

2.2.1.1 Error Detection

One of the error detection techniques is the detection of syntactic inconsistencies in the source coded bitstream. An example for a syntactic inconsistency is an unexpected occurrence of a marker in the middle of the data or the absence of a marker where it was expected, e.g. at the end of a data block.

The errors can also be detected based on semantic inconsistencies in the source coded bitstream. For instance a semantic inconsistency occurs if the count of macroblocks is different from the one indicated after a resynchronization marker.

2.2.1.2 Error Localization

Localization of errors is essential for their concealment, since the better the error localization, the smaller amount of correct data must be discarded. The error localization can be performed by exploiting the error resilient entropy coding. For example, Reversible Variable Length codes (RVLC) [63] can be used to localize the errors as illustrated in Fig. 2.5. If RVLC codes are used and an error occurs, all the data up to the next marker is skipped and then a backward decoding process is started. It helps to better locate the occurrence of the error and to minimize the discarding of correct data.

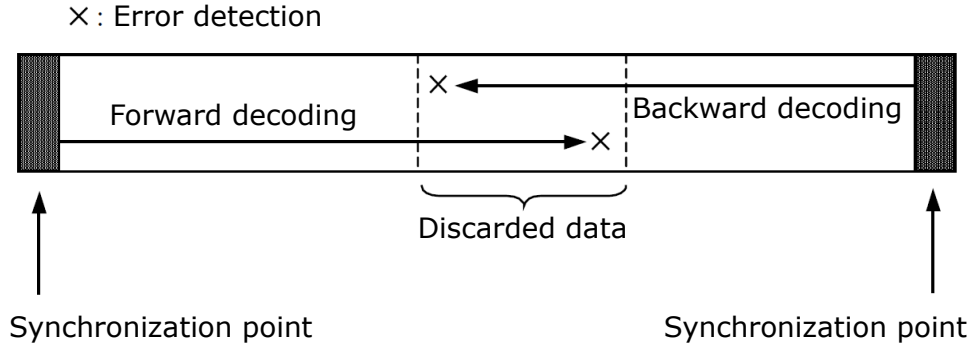
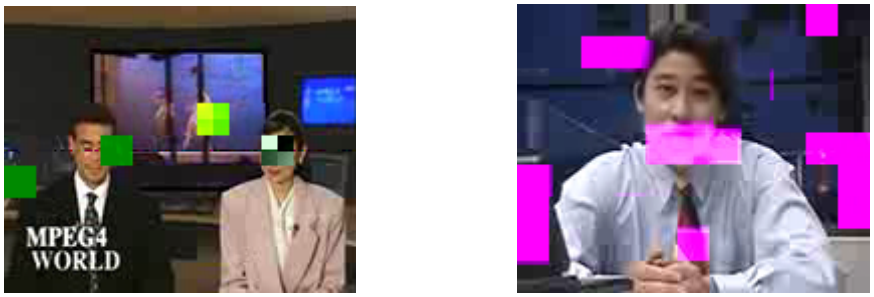


Figure 2.5: Error localization capability of RVLC codes [58].



(a) (b)

Figure 2.6: Video frames with erroneous 8×8 blocks aligned with the 8×8 block grid [58].

Since errors are many times detected too late after they occur, some corrupted texture data is "correctly" decoded. It can lead to very strange and visible image artifacts like for example green 8×8 blocks aligned with the 8×8 block grid in Fig. 2.6 (a). These artifacts may be detected by post-processing techniques such as looking for strong image discontinuities aligned with the coding grid or defining some constraints, e.g. in terms of color saturation. For instance, in a video telephone application, it is reasonable to assume that 8×8 blocks with highly saturated colors e.g. of green or pink are unlikely (see Fig. 2.6 (b)). Hence, such blocks can also be considered erroneous. However, criteria that might be reasonable for one application may not be for another. 8×8 blocks with saturated colors might exist in an application including synthetic image or video content. The post-processing techniques can be switched off if desired since they always involve a probability of detecting false positives, although only a very low one.

2.2.1.3 Error Concealment

After the errors are detected and localized, the decoder can start to conceal them. The concealment process is always based on some sensible assumptions, which may depend on the type of application and content involved. Video error concealment techniques can be classified into the three categories, depending on the data that is used [58]:

1. *Spatial error concealment* – only data from the current time instant is used to conceal the errors. Corrupted areas in the current video frame are recovered using the data from the surrounding correctly decoded areas of the same frame. This approach can have serious problems, if the corrupted area is quite large, in particular if it is very inhomogeneous. On the other hand, these techniques work relatively well for smaller or homogeneous areas and in sequences where there is very little temporal redundancy such as a first frame of video sequence after a scene cut or in the first frame of an entirely new video sequence.

2. *Temporal error concealment* – data from other (mostly previous) time instants is used to conceal errors. When these techniques are applied to sequences where a large temporal redundancy exists, the results are usually very good. But if the decoder tries to fix the image in the current time instant with data from images in surrounding time instants which are completely different, for example at a scene cut, serious problems can appear.

3. *Spatio-temporal error concealment* – data from both the current time instant and other time instants is used. These techniques try to get the best of both the spatial and the temporal concealment techniques. Some parts of the image might be concealed using spatial concealment, some parts by using temporal concealment and other parts by using both methods. Ideally, each of these solutions should be used for the parts of the image that are spatially, temporally or both spatially and temporally very homogeneous, respectively.

In our work we focus on spatial error concealment techniques because these techniques can be used not just for videos, but also for static images. We target to real time or nearly real time techniques that can be used for the error concealment of images and also for concealment of video frames of transmitted videos.

Chapter 3

Image Error Concealment

3.1 Related Work

One of the first algorithms for concealment of transmission errors proposed by Shirani et al. [56] is based on the Markov random fields method. It focuses on preserving important visual features and edges, but it fails in completing of textures.

Zhai et al. [83] introduced a block-based bilateral filtering algorithm which extends the classic bilateral filtering by operating in block-wise manner. The method has the ability to capture the block-level similarity which suits the needs of error-concealment for block based image compression. Although this method can complete a texture by copying the entire blocks, it works well only if the blocks have uniform textures. The method also has problems if several consecutive blocks in a row are corrupted, what happens unfortunately very frequently.

Sparse linear prediction estimator proposed by Koloda et al. [35] recovers lost regions in images by filling them sequentially with a weighted combination of patches that are extracted from the available neighborhood. The weights are obtained by solving a convex optimization problem that arises from a spatial image model. However, this method is again suitable only for concealment of isolated blocks.

3.1.1 Frequency-selective Methods

A different approach for image error concealment is the frequency-selective extrapolation (FSE) by Kaup et al. [32]. Correctly received parts of the image are approximated by a

Figure 3.1: Concealment of block losses using the frequency selective extrapolation technique by Kaup et al. [32]. Left: Consecutive 16 × 16 block losses. Right: Image with errors concealed.

set of basis functions, which are defined over a square area covering both uncorrupted and corrupted parts of the image. Approximation of correctly received data is an iterative process of minimizing error through successive selection of the most dominant basis functions. The FSE is able to consistently expand the signal (i.e. image) with different content such as image with texture or edges or a simple smooth color image (see the example result in Fig. 3.1).

A more recent implementation of a complex FSE method by Seiler et al. [54] provides similar quality reconstruction with a 10× shorter runtime. This algorithm iteratively generates a model of the signal to be extrapolated as a weighted superposition of Fourier basis functions.

The latest modification of the FSE by Koloda et al. [34] is based on a priori information about the low pass behavior of the natural images. Without this assumption, FSE may introduce high frequency artifacts into the image reconstruction. The added low pass filtering of the residual errors in the iterative process of FSE produces slightly improved results, but it also increased the computational complexity a bit.

The problem of FSE methods in general is that they always approximate the square surrounding of the lost area that may often contain parts of the image with unrelated textures. Approximation of the square area containing multiple different textures is more time consuming than approximation of the area with a single texture and the result does not achieve

sufficient quality.

3.1.2 Inpainting Methods

Another significant group of methods that are not primarily designed for concealment of transmission errors, but can be used for this purpose are the so called inpainting methods. Inpainting methods can be divided into two principal categories: *geometry-based* methods and *pattern-oriented* methods.

3.1.2.1 Geometry-based

The concept behind of one of the first geometrically-based inpainting methods by Bertalmio et al. [12] is to smoothly propagate information from the surrounding areas in the isophotes[1] direction. The most important problem lies in the fact that the method is unable to reproduce the texture into larger areas. Moreover, it takes a few minutes to obtain the result using a standard desktop PC.

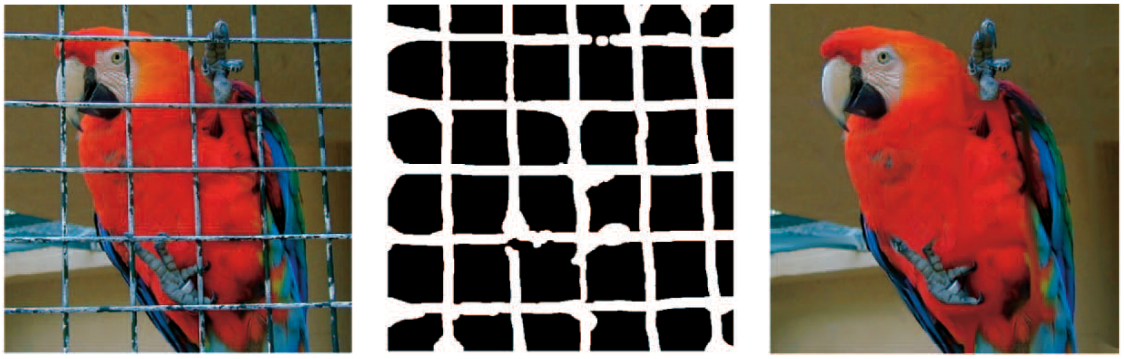
Vector-valued image regularization with PDEs [64] smooths the input image before it is converted into a vector image. The method has various applications, such as e.g. enlargement of the image, smoothing and noise removal, and can also be used to complete small corrupted areas. But as you can see in the Fig. 3.2 the inpainting result is too blurred.

The method by Sun et al. [61] allows the user to identify important structures in the image through an application interface. Line segments or curves can be used to mark how the structures should continue in an unknown area. An example of a such a structure can be a window frame, ladder or the border of the horizon or the silhouette of mountains in an image from nature (see Fig. 3.3). On the one hand it is an advantage that the user has the opportunity to improve the result, but on the other hand the image completion is not fully automatic, thus, not suitable for concealment of transmission errors.

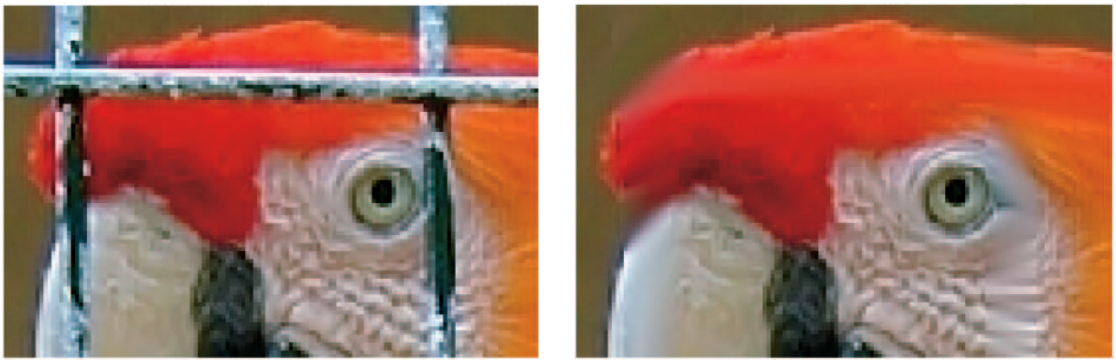
3.1.2.2 Pattern-based

One of the first pattern-oriented inpainting methods by Efros and Leung [21] synthesizes the texture by growing a new image outward from an initial seed, one pixel at a time. A Markov random field model is assumed and the conditional distribution of a pixel given all

[1] isophote - a contour of equal luminance in an image



(a) Original image (b) Inpainting mask (c) Inpainted image



(d) Detail of original image (e) Detail of inpainted image

Figure 3.2: Using *Vector-valued regularization PDE's* [64] for image inpainting.

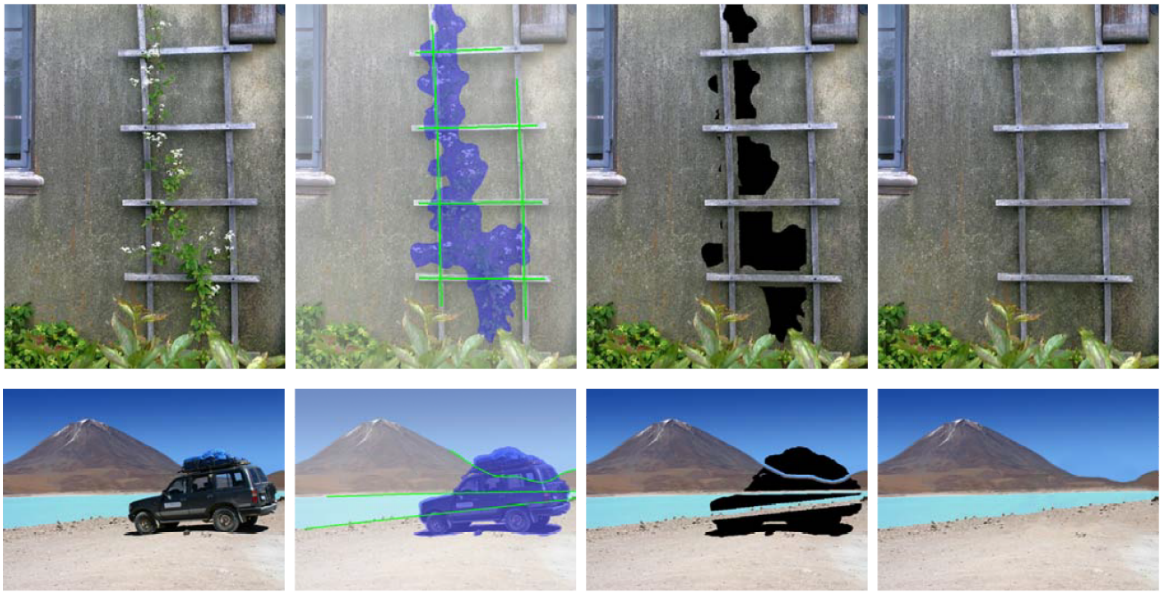


Figure 3.3: Image Completion with Structure Propagation by Sun et al. [61]: The first column shows original images. The second column shows unknown regions (blue) and input curves (green). The third and fourth columns are completion results after structure propagation and the final results after texture propagation, respectively.

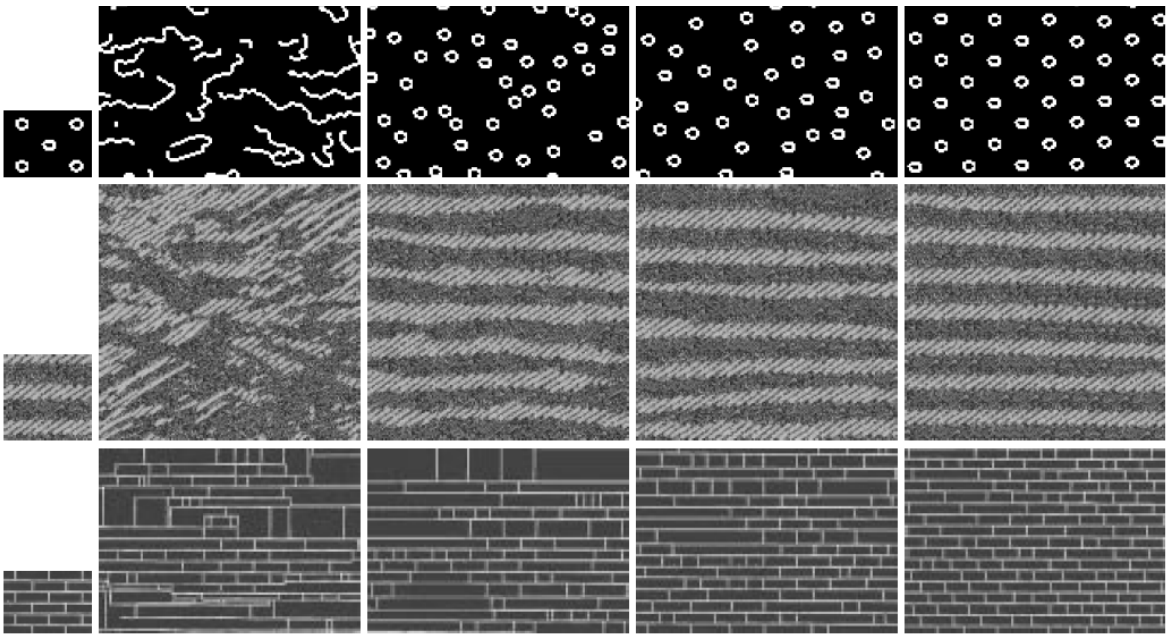


Figure 3.4: Texture Synthesis by Efros and Leung [21]: given a sample image (left), the algorithm synthesized four new images with square neighborhood windows of width 5, 11, 15, and 23 pixels, respectively. The window size perceptually intuitively corresponds to the degree of randomness in the resulting textures.

its neighbors synthesized so far is estimated by querying the sample image and finding all similar neighborhoods. The degree of randomness is controlled by a single perceptually intuitive parameter (see the Fig. 3.4). The method aims at preserving as much of the local structure as possible and produces satisfying results for a wide variety of synthetic and real-world textures. It can be used as an inpainting method if the hole to be completed can be filled by a single texture. For more complex images with lots of different textures, this method fails. Moreover, it is very slow, since the texture is synthesized pixel-wise.

The inpainting method by Criminisi et al. [15] is based on patch-based modeling of an image. When compared to the previous method [21] which progresses by pixels, this method fills the missing region by copying large parts of the surrounding image, i.e. patches. This approach improves the processing speed and the accuracy of the completed structures. In addition, the method introduced the principle of following of saliency edges and their primary completion (see the Fig. 3.5). However, this principle only works for straight lines and can not cope with a rounded or curved edges.

The key idea of the method by Mansfield et al. [37] is that the patches remain similar under a variety of transformations such as scale, rotation and brightness change (see an example with rotation in the Fig. 3.6). The problem is that enumeration of the all possible transformations is computationally too expensive. Without setting up a few basic user restrictions (e.g. limiting or not using some transformations) this method could run for several hours.

(a) (b) (c) (d) (e)

Figure 3.5: Removing large objects from photographs by Criminisi et al. [15]. (a) Original image. (b) The target region (in white with red boundary) covers 12% of the total image area. (c,d) Different stages of the filling process. Notice how the isophotes hitting the boundary of the target region are propagated inwards while thin appendices (e.g., the arms) in the target region tend to disappear quickly. (e) The final image where the bungee jumper has been completely removed.

Therefore the authors of this method proposed to explore the usage of different optimization methods to search for the best patches and they also use methods such as automatic detection of recurrent elements or symmetry.

So far, probably the best visual results are achieved by *Image Melding* [17] (see Fig. 3.7) built upon a patch-based optimization foundation with three key generalizations: (1) The enriched patch search space with additional geometric and photometric transformations. (2) Image gradients integrated into the patch representation and replacement of the usual color averaging with a screened Poisson equation solver. (3) A new energy computation based on mixed L_2/L_0 norms for colors and gradients that produces a gradual transition between sources without sacrificing texture sharpness. The issue with Image Melding is that it is very time consuming.

Daisy et al. proposed a spatial patch blending technique [16] to reduce artifacts of patch-based inpainting methods (see Fig. 3.8). The technique first searches for artifacts and subsequently it conceals them. Authors of this technique implemented both: the method by Criminisi et al. [15] together with their proposed concept [16] as a free plug-in for the graphic software GIMP, making it available to the public. The method is very fast, but it does not achieve as good results as the Image Melding [17].

Authors of the *Group-based Sparse Representation* (GSR) method [84] claim that the traditional patch-based sparse representation modeling of natural images usually suffer from two problems: First, it has to solve a large-scale optimization problem with high computational complexity in dictionary learning. Second, each patch is considered independently in dictionary learning and sparse coding, which ignores the relationship among patches, resulting

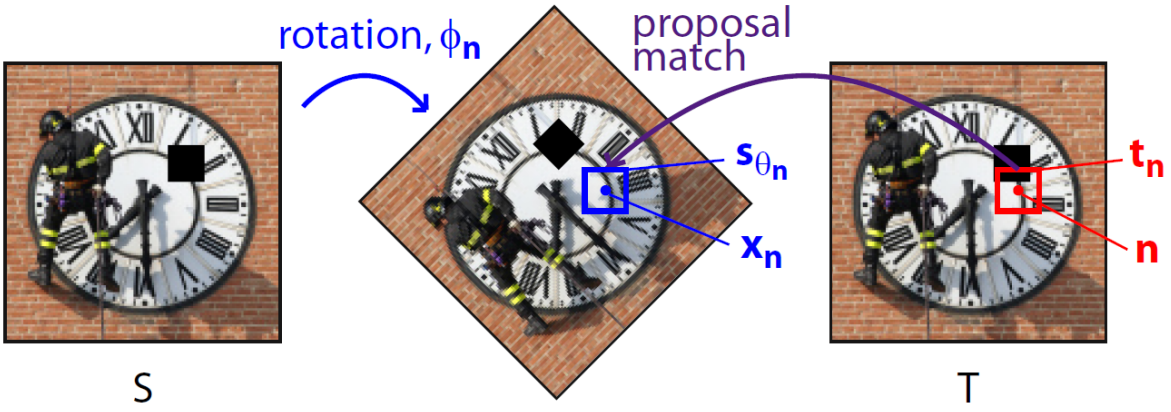
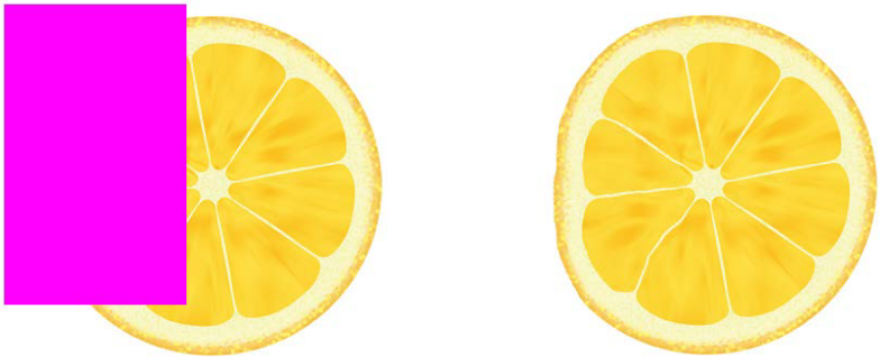


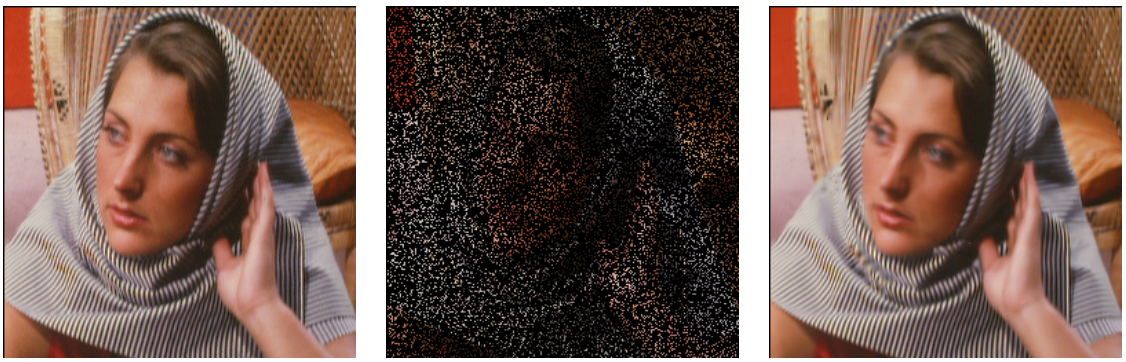
Figure 3.6: Transformations for image completion by Mansfield et al. [37]: For each patch in the target image T are found matching patches in the source image S under transformations, e.g. rotation.



(a) (b)

Figure 3.7: *Image Melding* [17] successfully fills large holes using a richer search space - it can exploit rotational and reflection symmetry, complete edges and textures using examples from different orientations, scales and colors. Left: Original image with pink mask that marks region to be filled. Right: Filled image.

Figure 3.8: Illustration of spatial patch blending algorithm by Daisy et al. [16]. From left to right: Image with marked area to reconstruct; reconstruction result with the inpainting algorithm from Criminisi et al. [15]; spatial patch blending applied to Criminisi's reconstruction result.



(a) (b) (c)

Figure 3.9: Reconstruction with GSR method [84]: Left: Original image. Middle: The degraded image with only 20% random samples available. Right: The recovered image by GSR method.

in inaccurate sparse coding coefficients. Therefore, they proposed to replace the *patch* as a basic unit of sparse representation by a *group* that consists of non-local patches with similar structures. Moreover, they proposed the dictionary learning for each group with a lower complexity compared to dictionary learning from natural images. This method achieves great results if areas that need to be filled are very small. However, in case of larger areas the results are often too blurry. Moreover, despite various improvements, the method is still considerably slower than other techniques. To repair one image it needs several minutes.

3.2 Proposed Method for Texture Aware Image Error Concealment

Based on analysis of existing methods for image error concealment we identified main problems and we proposed a method that addresses these problems. FSE methods [32, 54, 34] approximate and extrapolate square surroundings of corrupted area while they do not take into account content of the square area that is usually very diverse. Such approximation of various textures leads to mixture and in resulting extrapolation it brings undesirable artifacts. Similarly we often encounter artifacts when using simple patch-based methods [15, 16] that inpaint the corrupted area by copying patches from uncorrupted surroundings of the missing area. This observation led us to design one additional step into the process of error concealment. We propose to add a segmentation step which helps to prevent the production of artifacts.

A step-wise overview of our method is displayed in Fig. 3.10. As the input of the algorithm we have a corrupted image. In Fig. 3.10 (a) there is a black stripe in the middle of the

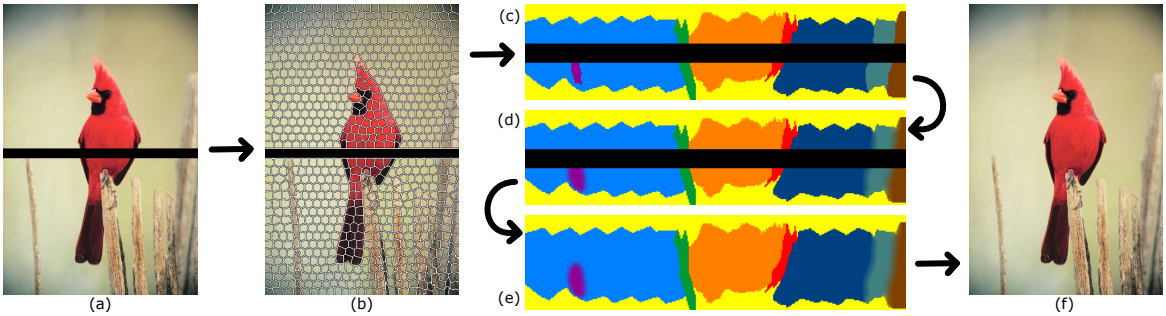


Figure 3.10: The proposed method: (a) Erroneous input image; (b) Superpixels; (c) Segmentation of areas surrounding corrupted area; (d) Fuzzy segmentation (note the fuzzy borders of the purple and gray segments); (e) Concealed errors of the segmented areas; (f) Concealed output image.

image that represents corrupted successive macroblocks. The first step of our algorithm is over-segmentation of the corrupted image by superpixels (Fig. 3.10 (b)). Superpixels should have uniform color and texture, similar size, they should be regularly distributed and their borders should follow significant edges in the image. There are several superpixel generating methods that we briefly describe in section 3.2.1.1. The important fact is that the superpixels can be generated in real-time. By merging similar superpixels we get image segmentation. In Fig. 3.10 (c) there are merged only superpixels adjacent to corrupted area and all the remaining superpixels are marked by yellow color. We adapted a superpixels merging algorithm to be able to merge superpixels across corrupted regions. It is described in section 3.2.1.3.

The main drawback of using the segmentation is a high level of uncertainty regarding the segment borders in image areas containing unsharp edges or gradients. Therefore, in addition we propose an extended form of the segmentation which adds soft edges to the segments and allows them to overlap (see Fig. 3.10 (d)). By introducing fuzzy segment borders, uncertain decisions are omitted.

The next step of the algorithm is completion of the segmentation in corrupted regions of the image (Fig. 3.10 (e)). For this task we developed an original algorithm that is based on shape error concealment techniques for single objects. We describe whole process in section 3.2.2.

The last step of our method for image error concealment is the texture extrapolation. Based on image segmentation, our method selects only the most relevant uncorrupted pixels for the extrapolation. By dealing with each segmented region separately, the quality of the results increases. Fast frequency selective extrapolation methods [54, 34] use a straight forward approach which extrapolates every pixel falling within the rectangular surroundings of the lost area. High quality image reconstruction is based on complex patch-based [17] and group-based methods [84] with run times several orders of magnitude longer. Our algorithm is

focused on improvement of quality for fast image restoration in the time span of a few seconds.

All steps of our texture aware image error concealment method are described in detail in following sections of this chapter.

3.2.1 Segmentation with Superpixels

Superpixels are produced by a deliberate over-segmentation of the image with the goal to create small compact areas which can be used as basic units in the following image processing steps. Each superpixel should enclose a homogeneous area in terms of color and texture. Additionally, the superpixel edge should follow the salience edges in the image. On the other hand, a regular size and distribution of superpixels is preferable, too. The main aim of superpixels is to reduce the redundancy in the image and also to increase the calculation efficiency.

Superpixels have been already used in several application domains: salient object segmentation [30], object tracking [73, 85], salience detection [40], video analysis and segmentation [20], interactive 3D reconstruction from video [69] etc.

3.2.1.1 Superpixel Producing Methods

A number of methods for producing superpixels has already been proposed. A grouping algorithm formulated as a graph partitioning problem was proposed by Ren and Malik [49]. It introduced normalized cut criteria for the graph segmentation.

The *Constant intensity superpixels method* [71] optimizes superpixels in an energy minimization framework using graph cuts. The utilized energy function favors regular superpixels.

Felzenszwalb's efficient graph based segmentation [22] has a single scale parameter influencing the segment size. The actual size and number of segments varies depending on the local contrast.

The method known as *Quickshift image segmentation* [23] is based on an approximation of a kernelized mean-shift. It belongs to the local mode-seeking algorithms, utilizing the LUV color and location of the pixel.

One of the most recent algorithms proposed by Benešová and Kottman [11] uses morphological reconstruction of input image to remove local extrema. Morphological reconstruction helps following watershed procedure to not create too small superpixels.

Simple linear iterative clustering (SLIC) [5] is a superpixel generation algorithm based on the k-means image segmentation in the CIE $L^*a^*b^*$ space extended by spatial pixel coordinates. In the initialization step, positions of the seeds are sampled on a regular grid. Afterwards, the k-means clustering is calculated for each seed and subsequently the position of the seed is iteratively updated. Typically, 5-10 iterations are necessary. At last, connectivity is enforced by stitching small superpixel fragments to neighboring superpixels. For balance between a regular form of the superpixels and the actual color differences, a compactness constant as in the case of the Quickshift is included as a weighting factor in the distance measure.

A fast version of SLIC implemented on the GPU is called *gSLIC* [48]. The recent *Accelerated gSLIC* [13] allows interactive rates for images up to the size of 1280×960 pixels.

In this work, we applied SLIC method for a good trade-off between the run-time and the accuracy. In our algorithm we had to apply few modifications of the original SLIC method to make it applicable on corrupted images. We describe the method and our modifications in detail in following sections.

3.2.1.2 Modified SLIC Method for Corrupted Images

SLIC algorithm is based on principle of k-means clustering. Every pixel is associated with a 5-dimensional vector $[L^*a^*b^*\ x\ y]$, where $L^*a^*b^*$ are coordinates in CIE $L^*a^*b^*$ color space and x, y are spatial coordinates of a given pixel in the image. $L^*a^*b^*$ color space was proposed in a way that color differences measured as Euclidean distance in $L^*a^*b^*$ space corresponds with color differences given by human perception [29]. Although transfer from RGB to $L^*a^*b^*$ color space brings conversion error (due to missing information about spectral properties of the used camera), this error seem to be irrelevant and by using $L^*a^*b^*$ color coordinates it is possible to achieve better results compared to using RGB coordinates.

In the initial step of SLIC algorithm for uncorrupted images k leading centers of clusters $C_i = [L_i^*\ a_i^*\ b_i^*\ x_i\ y_i]$ are distributed into a regular grid with spacing of size $S = \sqrt{N/k}$, where N is a number of image pixels and k is a required number of superpixels.

In our modified algorithm the centers of clusters are distributed always out of the corrupted area, therefore depending on the size of the corrupted area we initialize smaller number of

superpixels. Also for this reason, it would be hard to estimate what is the optimal number of superpixels for the given image and therefore as the input into our algorithm we require constant S, from which is derived number of superpixels based on following formula $S = \sqrt{(N-N_p)/k}$, where N_p is number of corrupted pixels in the image. Also in following steps of our algorithm we skip corrupted pixels.

Every initial cluster center is moved within its 3×3 surroundings into such point, where there is the smallest gradient. This is necessary to avoid centering of the corresponding superpixel on an edge, and to decrease probability of seeding the superpixel into noisy point.

In a next step we assign each uncorrupted pixel of the image to some cluster. At the beginning each uncorrupted pixel (excluding cluster centers) has set infinite distance to the nearest center of a cluster and it is not assigned to any cluster. We go through each cluster center C parallely and for each pixel p in the area with dimensions $2S \times 2S$ around the cluster center C we count distance D between the p and C. Definition of distance D can be found in following section 3.2.1.2.1. If the calculated distance is less than the current value of the pixel's distance to the nearest cluster center then this distance is set to a new lower value and the pixel is assigned to a given cluster.

After passing through each cluster center each uncorrupted pixel p is assigned to such cluster where the distance D between p and the cluster center is the shortest. If a new point or points were added into some cluster, it is necessary to calculate a new center. We calculate it as mean $[L^*a^*b^*\ x\ y]$ vector of all pixels belonging to the cluster. Moreover we have to update the residual error E, that we count as L^2 norm between old and new cluster center. Steps of assigning and updating can be repeated iteratively until the error will converge. Typically, 5-10 iterations are necessary [5]. In our implementation we do not count residual error we always make 10 iterations.

At last it is necessary to perform a post-processing step to ensure the compactness of the resulting superpixels. Details can be found in the section The whole algorithm except for post-processing step is summarized in the algorithm 3.1.

3.2.1.2.1 Distance Measurement

Color of the point is represented in CIE $L^*a^*b^*$ color space as a three component vector $[L^*a^*b^*]$ and its range of possible values is known. Pixel position $[x\ y]$, on the other hand, can have a range of values which vary according to the size of the picture.

Defining the distance D between the pixel and the cluster center C_k in the algorithm 3.1 as a simple Euclidean distance in a 5-dimensional $[L^*a^*b^*\ x\ y]$ space would create inconsistencies

Algorithm 3.1 SLIC for superpixel segmentation of the corrupted image

/* Initialization */
Initialize cluster centers $C_k = \left[L_k^* \, a_k^* \, b_k^* \, x_k \, y_k\right]$ into a regular grid
with the spacing size S outside of the corrupted area.
Move the cluster centers within their
3×3 surrounding to the point with the smallest gradient.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.
for j = 0 ... 9

    /* Assigning */
    foreach cluster center C_k **do**
        foreach pixel i in $2S \times 2S$ area around C_k **do**
            Calculate the distance D between C_k and i.
            if $D < d(i)$ **then**
                set $d(i) = D$
                set $l(i) = k$
            end if
        end foreach
    end foreach
    /* Updating */
    Calculate new cluster centers.

end for

in clustering procedure for different sizes of superpixels. For large superpixels the spatial distance would override color similarity, thus, the spatial proximity would have relatively bigger importance than a color similarity. In this way, we would create compact superpixels, but they would not follow the edges in the image. For smaller superpixels it would be the opposite.

In order to combine the two distances (d_c as a color distance and d_s as a spatial distance) into a one scale, they need to be normalized based on the maximum color and space distances within the cluster. We mark maximum color distance as N_c and maximum space distance as N_s. Then the distance D' we get by the following derivation

$$d_c = \sqrt{(L_k^* - L_i^*)^2 + (a_k^* - a_i^*)^2 + (b_k^* - b_i^*)^2} \tag{3.1}$$

$$d_s = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \tag{3.2}$$

$$D' = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2} \tag{3.3}$$

Authors of the SLIC method approximates the maximum spatial distance within a given cluster as the size of the grid spacing from the initialization, thus, they define $N_s = S$. Determination of the maximum color distance N_c is not that easy because the color distances may vary significantly from one cluster to another and from one image to another. The authors of the SLIC method nevertheless decided to define N_c value using a constant m, that on the basis of experiments they determined to the value of the 10. Therefore the equation of the resulting distance D is

$$D = \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_s}{S}\right)^2} \tag{3.4}$$

In our experiments, we came to the conclusion that the proposed approximation of the maximum space and color distances within a cluster is too inaccurate. So we decided to calculate the maximum span of all five components of the vector $[L^* a^* b^* \; x \; y]$ exactly and therefore we work with a modified formula of the distance D

$$D = \sqrt{\left(\frac{L_k^* - L_i^*}{N_L^k}\right)^2 + \left(\frac{a_k^* - a_i^*}{N_a^k}\right)^2 + \left(\frac{b_k^* - b_i^*}{N_b^k}\right)^2 + \left(\frac{x_k - x_i}{N_x^k}\right)^2 + \left(\frac{y_k - y_i}{N_y^k}\right)^2} \tag{3.5}$$

Algorithm 3.2 Supplemented part of the Assigning from the algorithm 3.1 by the calculation of new values N_L^k, N_a^k, N_b^k, N_x^k,N_y^k

```
/* Assigning */
foreach cluster center C_k do
    foreach pixel i in 2S × 2S area around C_k do
        Calculate the distance D between C_k and i.
        if D < d(i) then
            set d(i) = D
            set l(i) = k
            calculate new values N_L^k, N_a^k, N_b^k, N_x^k,N_y^k
        end if
    end foreach
end foreach
```

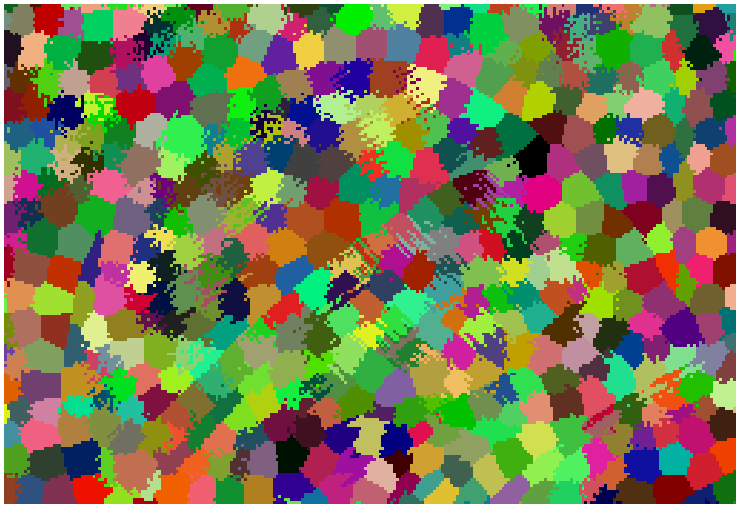
In the algorithm 3.1 it means only a minor change. After some pixel is assigned to the cluster k, we calculate the new values of the maximum range of the individual $[L^*a^*b^*\ x\ y]$ components of the vector for a given cluster k. A supplemented part of the assigning can be seen in the algorithm 3.2.

3.2.1.2.2 Ensuring Superpixel Compactness

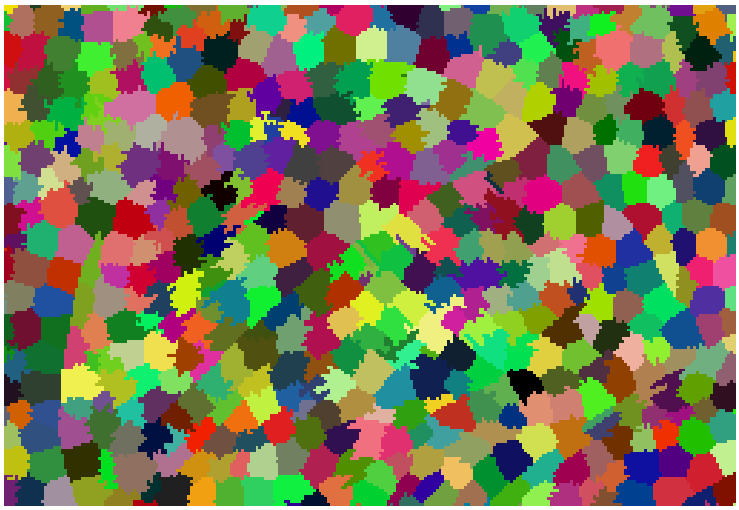
The result of the SLIC algorithm is that each uncorrupted pixel in the image has been assigned to the one of the clusters. But as yet we have no information about the compactness of the clusters. In order to obtain this information, we go through all pixels by the Flood Fill algorithm. We begin in the upper left pixel and with Flood Fill we fill all the surrounding pixels that are assigned to the same cluster as the first pixel. Pixels passed by Flood Fill create a first superpixel. The superpixel carries information about coordinates of all pixels belonging into it and what is their average $L^*a^*b^*$ value. We store the superpixel in the associative array with the key equal to the cluster index.

We pass through each uncorrupted pixel, that is not yet assigned to any superpixel, by the Flood Fill algorithm, i.e. we fill all the surrounding pixels that are assigned to the same cluster and we create a new superpixel for these pixels. Again we store created superpixel in the associative array with the key equal to the cluster index. After passing through all the uncorrupted pixels we have in the associative array for each key stored at least one superpixel. Each key, which has allocated more than one superpixel, does not have a compact cluster. The cluster is formed by several superpixels. Our goal is that each cluster is compact (see Fig. 3.11).

(a) The input image - hat of Lenna



(b) Non-compact clusters



(c) Compact clusters = resulting superpixels

Figure 3.11: Comparison of compact and non-compact clusters.

Algorithm 3.3 The process of assigning parts of the non-compact clusters to the most similar neighbors

Sort the array S of superpixels that should be assigned to another cluster from the smallest (with the least number of pixels).
foreach superpixel s from array S **do**

Assign superpixel s to its the most similar neighboring superpixel t.
Update the list of pixels belonging to superpixel t (add to the list all pixels of s).
Recalculate the average $L^*a^*b^*$ value of t.
Update the list of neighbors of t and sort it from the most similar neighbor.
Ask neighboring superpixels to update their lists of neighbors.

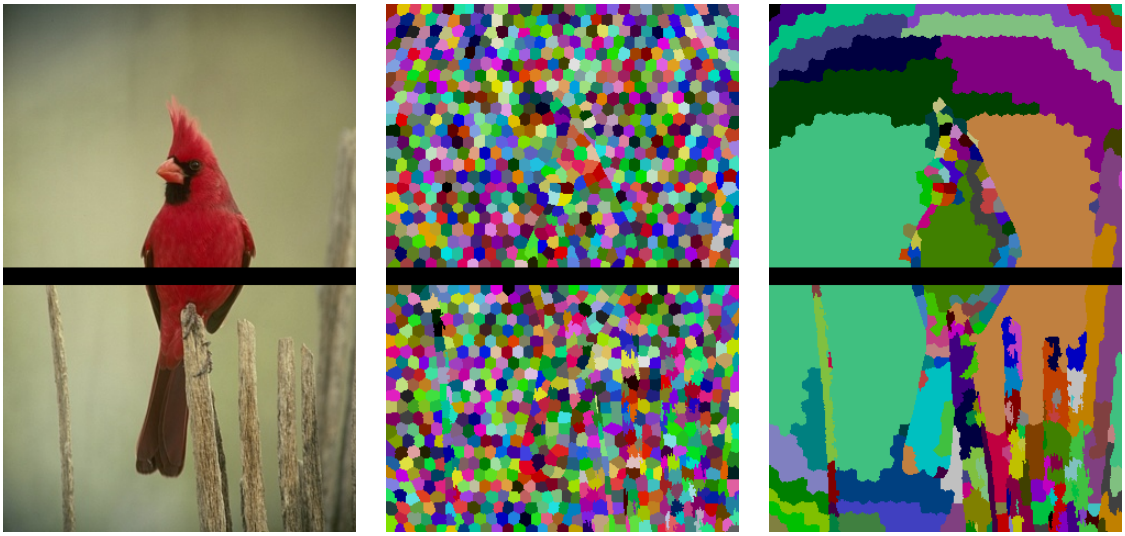
end foreach

For each non-compact cluster we sort all its superpixels starting with the largest (with the biggest number of pixels) to the smallest. We leave the largest superpixel in a given cluster, and we will assign each smaller superpixel to another cluster adjacent to it. Such smaller superpixels from all clusters we collect into one array and we sort them starting with the smallest. The smallest superpixels we assign to the neighboring clusters first. Our goal is to assign a superpixel to a neighbor with a most similar color. We determine a similarity of the two superpixels according to the formula

$$D_{Lab} = \sqrt{(L_2^* - L_1^*)^2 + (a_2^* - a_1^*)^2 + (b_2^* - b_1^*)^2} \tag{3.6}$$

where $L_1^* a_1^* b_1^*$ are the average $L^*a^*b^*$ values of one superpixel and $L_2^* a_2^* b_2^*$ are average values of the second superpixel. The smaller the D_{Lab} value the more similar the two superpixels are.

Since we want to assign superpixels from non-compact clusters to their neighbors with the most similar color, in the data structure of each superpixel, we will also keep a list of its neighbors sorted from the most similar. Then the process of assignment of a superpixel s to the most similar neighbor t consists of several steps. To the list of pixels of t we add all pixels of s. Then we recalculate the average $L^*a^*b^*$ value of t. And we update the list of neighbors of t - we add into the list all neighbors of s, except t, and we delete s from the list, because it actually already does not exist. Finally, we ask neighboring superpixels to also update the list of their neighbors. The process of assigning of the non-compact parts of the clusters to the most similar neighbor is clearly described in the algorithm 3.3. The result of the algorithm is a low level image segmentation with superpixels (see Fig. 3.11c).



(a) (b) (c)

Figure 3.12: The process of segmentation: (a) Corrupted input image; (b) Low level image segmentation with superpixels; (c) High-level image segmentation.

3.2.1.3 Merging Superpixels in Corrupted Images

To achieve a high-level image segmentation, similar superpixels must be merged (see Fig. 3.12). We adapted the merging algorithm proposed by Birkus [13] to merge across the lost areas. Algorithm proposed by Birkus is designed so that each superpixel is compared with each its neighbor to decide whether to merge them into a single segment. The basic idea of our algorithm is that two superpixels are considered neighbors even when they are separated by the lost area.

When we implemented the rule, that superpixels are considered neighbors even when they are separated by the lost area, we encountered an issue. Sometimes happen that two superpixels, that were very distant from each other (each lay on the other side of the lost area), were merged into one segment. Therefore, we had to add a condition that the distance of two superpixels which will be considered as neighbors, must not exceed a certain length. This length have been determined as $1,5$-times the size of the longest side from the shorter sides of bounding boxes of all lost areas in the image.

At the beginning of the merging process of superpixels we create as many segments, as there are superpixels. To each segment, we assign one superpixel as the first component of connectivity. Segment data structure carries the same information as the data structure of superpixel - there is information about which pixels belong to the segment, what is the average $L^*a^*b^*$ value and the list of adjacent segments. In this case, we add into the list of neighbors also neighbors over the lost area.

We pass through all segments parallely to find out whether we can associate to the segment some of its neighbors. Whether the segment t should be assigned to the segment s, depends on their similarity. The most convenient way to determine the similarity of two segments is to compare their mean $L^*a^*b^*$ values, i.e. to compare their colors. But it is also possible to compare texture structure of the segments. More about comparing segments based on their textures is in the next Chapter 4. Here for the simplicity we compare only segments color.

The color similarity of two segments can be determined using the formula 3.6. If the value D_{Lab} is less than a threshold given by the user, then the segment t will be assigned to the segment s. It means, that the entire list of components of t will be added to the list of components of s. Next, we add all the pixels of t to the list of pixels of s and also we recalculate the average $L^*a^*b^*$ value of s. In addition, we update the list of neighbors of s - we add into its list all neighbors of t, except s, and we delete t from the list, because it actually already does not exist. Finally, we ask neighboring segments, that they also update the list of their neighbors. Cycle of going through all the segments is repeated, until there are found at least two segments to be merged, thus, they are more similar as the specified minimum similarity. After this cycle of merging segments we have to merge adjacent components belonging to the same segment. This is necessary because we want that the resulting components are components of the connectivity of the given segment.

After we finish the process of merging segments, we have created high-level segmentation, but it is only the segmentation of corrupted image (see Fig. 3.12 (c)). Therefore we will have to complete the segmentation in the lost areas (see section 3.2.2). Based on the full segmentation, we will already know, which textures adjacent to the lost area we should extrapolate to which part of the lost area. The process of texture extrapolation is described in section 3.2.4.

3.2.1.4 Fuzzy Segmentation in Corrupted Images

Some parts of the image in Fig. 3.10(a) can not be segmented properly because of soft edges or gradients. Therefore, in addition to our original method [68] we create a segmentation with overlaps (see Fig. 3.10(d)). There is no longer a bijective mapping between the pixels and segments. The assignment is determined by certainty values, thus the term *fuzzy segmentation*.

Each segment disrupted by the lost area gets extended by similar pixels from its neighboring segments. We determine a similarity of a pixel and a segment according to the formula

$$D_{Lab} = \sqrt{(L_2^* - L_1^*)^2 + (a_2^* - a_1^*)^2 + (b_2^* - b_1^*)^2} \tag{3.7}$$

where $L_1^*a_1^*b_1^*$ are the average $L^*a^*b^*$ values of the segment and $L_2^*a_2^*b_2^*$ are values of the pixel. The smaller the D_{Lab} value the more similar are. Similarity threshold is given by the user.

The added pixels are required to be 4-connected to the original or to the extended segment area. The certainty is represented in the range of 0.0 to 1.0 which is basically a normalized distance to the original segment border. The certainty of original pixels of the segment is set to 1.0. The further away, the less certain an added pixel is. Actually, the certainty values need to be computed only for the repaired segments within the corrupted regions. Details are provided in section 3.2.3.

3.2.2 Shape Error Concealment for Segmented Images

One of our key contributions is the extension of shape error concealment techniques for single objects to segmented images with each corrupted segment being a small object to conceal. Connectivity of neighboring segments and of segment pieces cut by the lost area into several pieces is the main challenge.

In Fig. 3.10(c), areas with the same color belong to the same segment. Each segment can be composed of: one (e.g. the purple segment), or more (e.g. the light blue segment) components of connectivity. The black stripe in the middle marks the lost area. Our goal is to conceal it by connecting equally colored areas, in other words to join components of each segment (see example output image in Fig. 3.10(e)).

In general, it is more likely that a contour of a segment will have a rather curved shape. Therefore, our approach for reconstruction of the missing contours is based on the related work for binary images known as *shape error concealment* which cover the topic of curved shapes reconstruction quite well. Various techniques [53, 59, 65] were proposed for object-based video where the scenes are understood as a composition of objects [28]. Missing parts of video objects boundaries can be smoothly completed using approximate curves like Hermite splines [53] or Bézier curves [59] or B-spline curves [65].

Our original shape error concealment method [68] is an extension of the approach by Tsiligianni et al. [65]. We adjusted some steps where additional information from a colored input image can be utilized to speed up the algorithm and to improve the results. Each step of our shape error concealment method [68] is explained in detail in following subsections of this section. In this section we deal with basic segmentation and we do not address fuzzy segmentation. Details about completion of fuzzy segmentation can be found in following section 3.2.3.

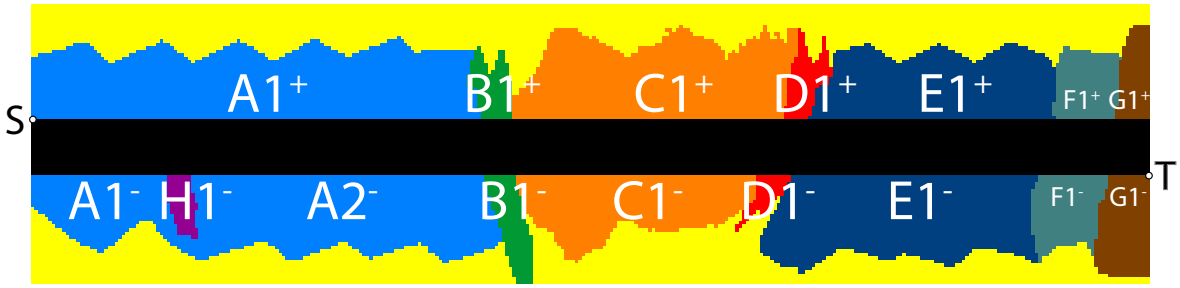


Figure 3.13: Labels and indices of corrupted segments obtained by tracing the lost area border. The segment *A* consists of two connectivity components, the lower one has two border parts.

3.2.2.1 Indexing Corrupted Segment Components

In the first step we identify segments disrupted by the lost area and mark them along the lost area border. We select two points (*S*,*T*) of the lost area – the top-left most and the bottom-right most. Then, we trace the contour of the corrupted area clockwise starting from *S* up to *T* and check pixels in a 4-connected neighborhood for multiple components or parts of the same segment. When encountering a segment for the first time we mark it by the index 1^+. When reaching the segment for the next time we increase the index to 2^+, etc. Thus, each part of a segment border in Fig. 3.13 is marked by a number consisting of the segment label and index number. To cover the whole lost area border, the same procedure is applied counter-clockwise while assigning negative indices.

3.2.2.2 Concealment of Simple Segments

Segments indexed only by positive or only by negative indices are denoted as *simple*. In order to be concealed no pairing of components is necessary for them. In Fig. 3.13 the purple segment *H* is the only simple segment. To repair its corrupted part we have to estimate its shape within the lost area.

The simplest way to connect the contour endpoints is to use a straight line segment. It would result in a simple G^0 geometric continuity of the contour. In most of the cases, better results can be achieved by targeting a higher degree of continuity. Existing methods [53, 59, 65] produce results with C^1 continuity by performing the following steps:

1. Approximate the known contour in the surrounding of the endpoints by a pair of curves, one for each endpoint.
2. Find tangent vectors of the curves in both endpoints.

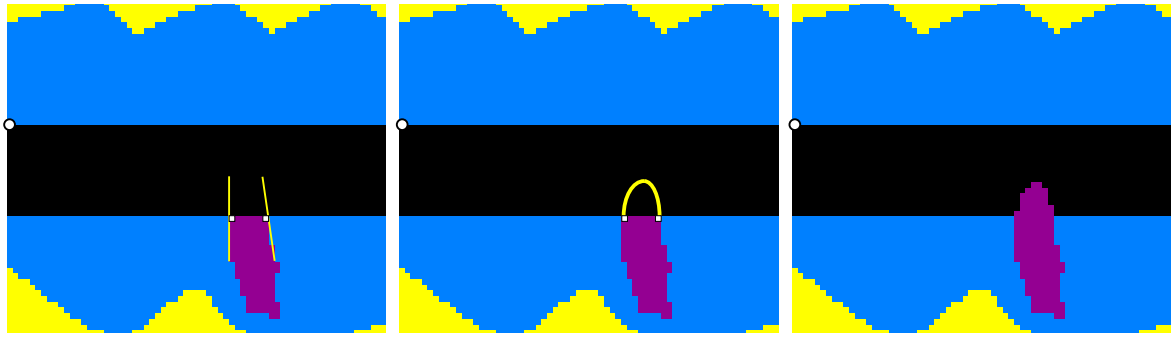


Figure 3.14: Tangents of the known segment contour (left) determine the concealment curve (middle) to enclose the lost pixels of the segment (right).

3. Use the tangent vectors to construct a concealment curve in the corrupted region such that it smoothly joins the endpoint curves from step 1.

We decided to use the same algorithm for all three steps as it was proposed in [65]. A B-spline curve is constructed to obtain a missing contour, based on a T-spline [10] representation of the extracted contour. T-splines produce shape preserving approximations and do not change the characteristics of the original contour. This representation ensures a good estimation of the tangent vectors at the endpoints. The reader is referred to [65] for more details. After the concealment curve is constructed we assign the enclosed area to the segment (see Fig. 3.14). The same procedure is applied to all simple segments.

3.2.2.3 Concealment of Paired Segments

Once all simple segments are repaired, we can start to process segments consisting of several components. The situation is more difficult, since segment components from different sides of the lost area need to be connected. For each broken part we find the closest one from the opposite side and connect their endpoints using the same algorithm as for simple segments. The endpoints are connected so that the concealment curves do not cross.

Segments $B \ldots G$ in Fig. 3.13 all consist of two border components, thus the pairing is straight forward. Segment A consists of $A1^+$, $A1^-$ and $A2^-$ which yield the ordered pairs

$$\left(A1^+, A1^-\right), \left(A1^-, A1^+\right), \left(A2^-, A1^+\right).$$

After transforming them into unordered pairs, there remain

$$\left(A1^+, A1^-\right), \left(A1^+, A2^-\right).$$

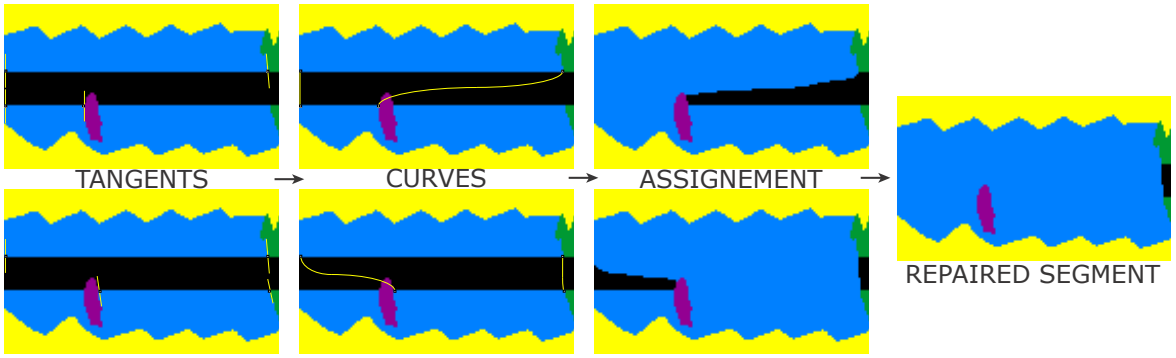


Figure 3.15: Shape error concealment of a the segment A from Fig. 3.13 connects each of its border components to the closes opposite one. In the top row $A1^+$and $A1^-$are paired, in the bottom row $A1^+$and $A2^-$ are paired. The final segment shape is given as the union of both partial results.

Shape error concealment for both pairs is demonstrated in Fig. 3.15. Overlapping of the respective areas is not an issue since the pixels will be all assigned to the same segment.

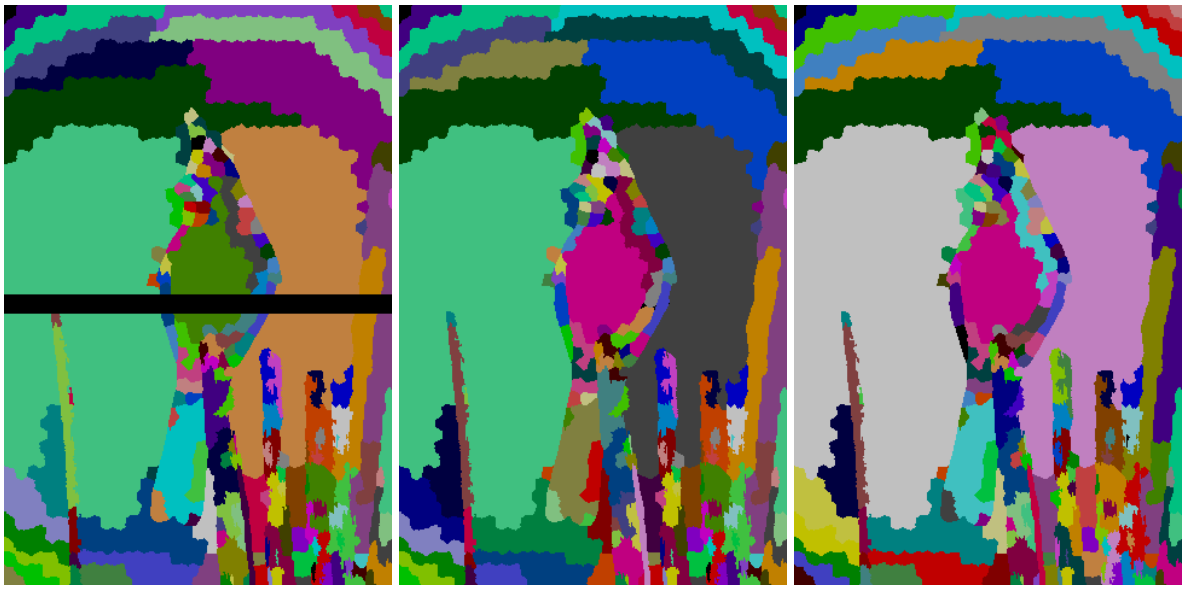
3.2.2.4 Concealment of Remaining Errors

The previous steps do not guarantee to conceal all errors. If there are simple corrupted segments on opposite sides of the lost area, some pixels between them might stay unassigned (see Fig. 3.16). In the last step of the concealment procedure, such pixels are iteratively repaired by a special mode filter. If at least the half of pixel's 9×9 neighborhood is already resolved, then the pixel gets assigned to the most common segment from its neighborhood.

Segments produced by pairing are located on opposite sides of the lost area. Therefore, we consider the shape of their connection to be optimal. On the other hand, the missing shape is repaired with much less confidence for simple segments. In order to improve the concealment results of the mode filter, a pixel is always assigned to a simple segment from its neighborhood, if any exists. After processing all segments we obtain the estimated segmentation of the lost area.

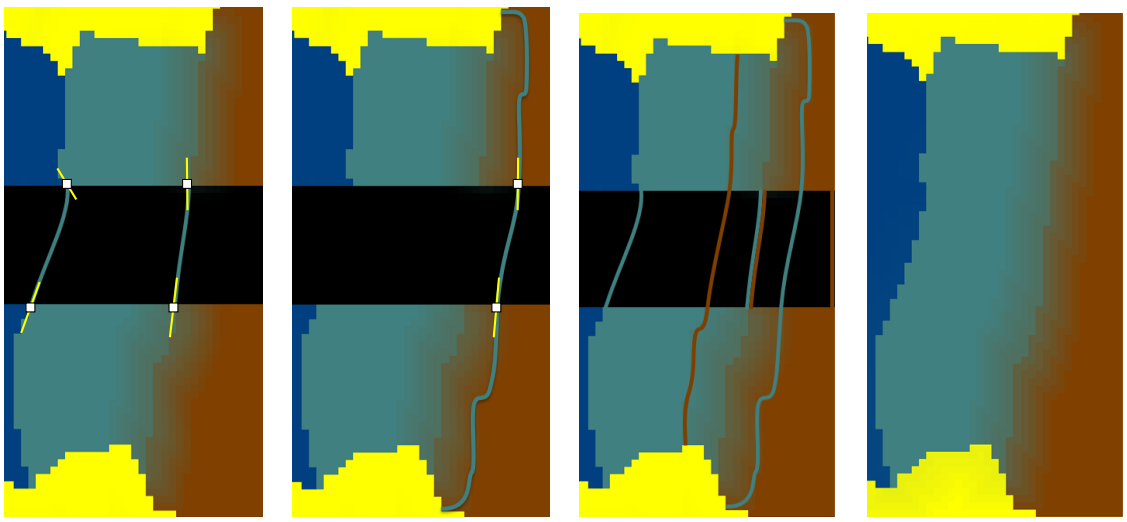
3.2.3 Completing the Fuzzy Segmentation

In a classic segmentation, for each segment we find tangents of its shape at the border of the corrupted region and construct B-spline curves to complete the corrupted segment shape (see Fig. 3.17(a)). After the concealment curves are constructed, we assign the enclosed area to the segment. A similar procedure is applied to the fuzzy part of the segment (Fig. 3.17(b)). We find its border, then its tangents in the endpoints of the fuzzy border and afterwards repair



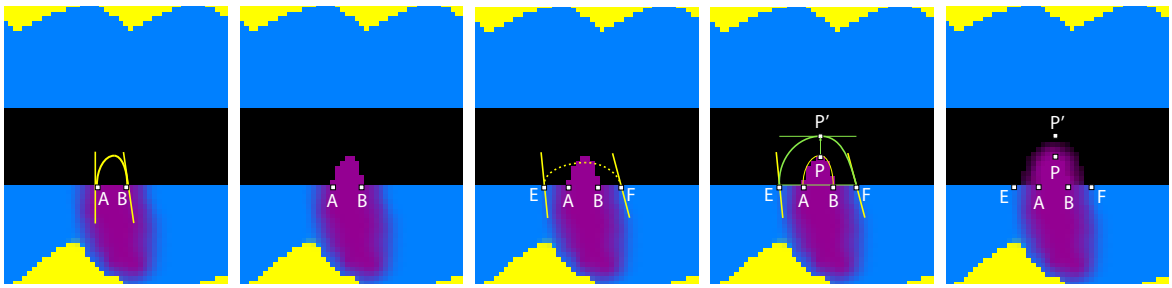
(a) (b) (c)

Figure 3.16: Concealment of remaining errors: (a) Input segmentation out of the corrupted area; (b) Concealed simple and paired segments; (c) Concealed remaining errors.



(a) (b) (c) (d)

Figure 3.17: Repairing a fuzzy segmentation in corrupted image: (a) Tangents and concealment curves of the original gray segment; (b) Tangents and concealment curve of the extended grey segment; (c) All concealment curves of the grey and brown segments; (d) The complete fuzzy segmentation.



(a) (b) (c) (d) (e)

Figure 3.18: Tangents of the known segment contour determine the concealment curve (a) to enclose the lost pixels of the segment (b). Tangents of the known extended segment contour (c) are not sufficient to determine a proper concealment curve for the fuzzy border. The point P' and its tangent have to be added to enhance the concealment curve (d) and the final fuzzy shape error concealment (e).

the extended border by a B-spline curve. Again we assign the enclosed area to the segment but in this case we additionally compute the certainty values of the assigned pixels. The certainty c_p of the pixel p is given as

$$c_p = \frac{d_{extended}}{d_{extended} + d_{original}} \tag{3.8}$$

where $d_{extended}$ is the Euclidean distance of the pixel p to the nearest pixel in the extended segment border and $d_{original}$ is the Euclidean distance of the pixel p to the nearest pixel in the original segment border. We apply the same procedure to all segments to complete the fuzzy segmentation of the image (see Fig. 3.17(d)).

3.2.3.1 Concealment of Simple Segments

In Fig. 3.10(c) the purple segment is the only simple segment. To repair its corrupted part we estimate its shape within the lost area by finding tangents in the endpoints determining the concealment curve and assign the enclosed area to the segment (see Fig. 3.18(a,b)). If we would apply the same procedure also to the extended part of the segment the concealment curve could cross the concealed segment shape as it is illustrated in Fig. 3.18(c). Therefore, we propose to find and use one more point with its tangent as the input for the concealment curve. We find such point P from the concealed part of the segment that has the biggest distance from the line segment EF connecting the endpoints of the extended segment contour (see Fig. 3.18(d)). Then we find the normal vector n of the line segment EF that points into the lost area and we normalize it. Afterwards we move the point P in the direction of the normalized normal vector n for the mean length of AE and FB line segments to get the point

P'. The tangent in point P' is parallel to the tangent of the line segment EF. We use the point P' with its tangent as the additional input for the concealment curve.

3.2.3.2 Concealment of remaining errors

In our original shape error concealment method [68] we proposed to use a special mode filter to conceal remaining errors. With fuzzy segmentation it is no more necessary. If a pixel gets assigned to at least one extended part of a segment, we consider such pixel corrected. But still, there might remain some small areas with uncorrected pixels. For each such area we find all neighboring segments. Then we assign each pixel from the uncorrected area to the extended part of all neighboring segments. The certainty $c_p(S)$ of the pixel p in the neighboring segment S is given as

$$c_p(S) = d_S^{-1}, \tag{3.9}$$

where d_S is the Euclidean distance of the pixel p to the nearest pixel from the neighboring segment S.

3.2.4 Texture Extrapolation

Based on the completed segmentation we already have information where we should extrapolate which part of the image texture and thus we can conceal the errors in the image. Wang et al. [74] developed a method for approximation of non-square areas. The texture of an area is successively approximated and then cut to the shape of the segment. Our proposed algorithm implements this principle for completion of texture in corrupted segments. We estimate a missing texture of a segment by extrapolating texture from the uncorrupted components of the given segment. Our goal is to extrapolate the texture so that it preserve its structure and to smoothly join extrapolated and uncorrupted parts.

3.2.4.1 Selection of Suitable Basis Function

Let A be an extended segment representing an arbitrary shaped region of a 2D discrete image. The uncorrupted part of the segment has an internal texture structure denoted as $f(n_1, n_2)$. First, the circumscribing rectangle L is found for A. Then, L is padded with zeros so that its width and height is a power of 2 to allow the use of fast transform algorithms. The size

of the rectangle L is now $N_1 \times N_2$. The texture structure of A within the rectangle L is then approximated using appropriate basis functions.

We use a set of orthogonal functions $u_{k_1,k_2}(n_1,n_2)$ where $k_1,n_1 \in \{0,1,\ldots,N_1-1\}$, $k_2,n_2 \in \{0,1,\ldots,N_2-1\}$. The texture approximation for L after v steps is denoted as $g^{(v)}(n_1,n_2)$. Using linear approximation, it can be expressed as a sum of basis functions weighted by appropriate spectral coefficients

$$g^{(v)}(n_1,n_2) = \sum_{k_1,k_2 \in K_v} c^{(v)}_{k_1,k_2} u_{k_1,k_2}(n_1,n_2),$$

where K_v denotes the set of basis function indices used in $g^{(v)}(n_!,n_2)$ and $c^{(v)}_{k_1,k_2}$ denotes the spectral coefficients. The residing difference between the original texture and its approximation is then

$$r^{(v)}(n_1,n_2) = f(n_1,n_2) - g^{(v)}(n_1,n_2).$$

Now we can approximate this difference by a suitable basis function to minimize the following error function

$$E_A = \sum_{(n_1,n_2) \in A_{uncorrupted}} [f(n_1,n_2) - g(n_1,n_2)]^2,$$

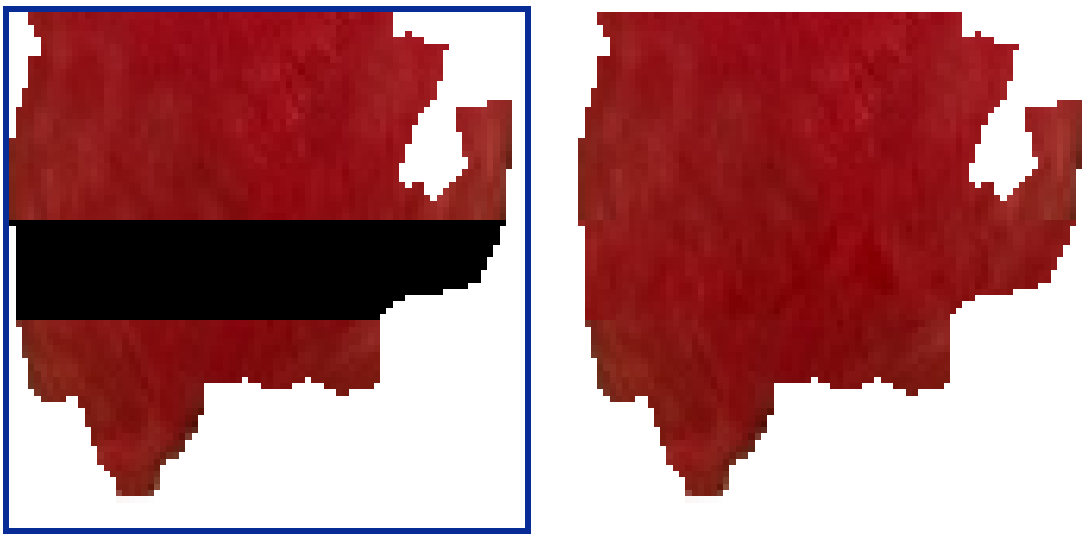
where $A_{uncorrupted}$ represents the pixels of the uncorrupted part of the segment. According to Polec et al. [42] the solution is given by maximizing

$$\Delta E^{(v)}_{A_{uncorrupted}} = \frac{\left[\sum_{(n_1,n_2) \in A_{uncorrupted}} r^{(v)}(n_1,n_2)\, u_{k_1,k_2}(n_1,n_2)\right]^2}{\sum_{(n_1,n_2) \in A_{uncorrupted}} \left[u^2_{k_1,k_2}(n_1,n_2)\right]}.$$

More details are provided in [42]. Figure 3.19 demonstrates application to a single segment using Discrete cosine transform (DCT) as described in work of Kaup [31].

3.2.4.2 Texture Extrapolation into Corrupted Part of the Segment

We iterate the texture approximation of correctly received parts of the segment until we reach sufficient quality. There are several methods for video and image quality measurement. In section 3.2.5.1 we describe four of them (Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), DCT-based video quality metric (VQM) and Structural Similarity (SSIM) index) because we use them to measure the quality of results of our and other error concealment methods. Here to determine the quality of the approximation we use the PSNR measure and its required value is specified by the user. After the uncorrupted part of the



(a) (b)

Figure 3.19: The texture of an arbitrarily shaped segment is successively approximated and then cut to the shape of the segment: Circumscribing blue rectangle L of the corrupted textured segment A (a); Texture extrapolation of the segment A into its corrupted part using successive Discrete cosine transform (S-DCT) (b).

segment has been approximated with sufficient quality, the corrupted part is extrapolated as the appropriate part of the rectangle circumscribing the whole segment.

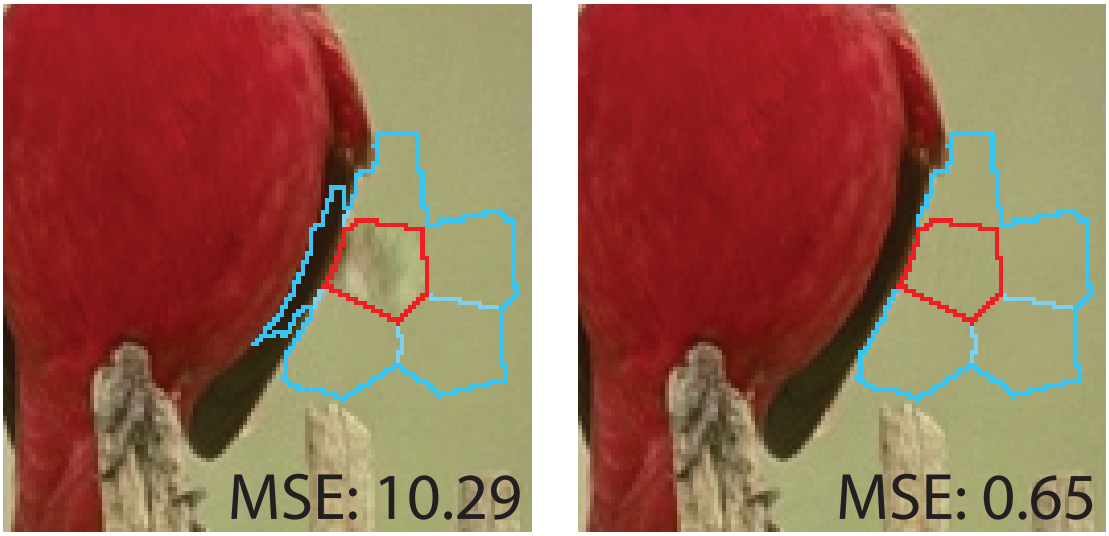
For color images, the approximation and extrapolation is performed separately for all channels of the YCbCr color space. Firstly we tried to apply the transformation to RGB color channels, but the results were not satisfactory, so we tried to use YCbCr color space and the achieved results were much better. Thus before the texture approximation we convert the image into YCbCr color space:

$$\begin{bmatrix} \mathrm{Y} \\ \mathrm{Cb} \\ \mathrm{Cr} \end{bmatrix} = \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} \tag{3.10}$$

The extrapolated textures we convert back to RGB color space:

$$\begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.400 \\ 1.000 & -0.343 & -0.711 \\ 1.000 & 1.765 & 0.000 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{Y} \\ \mathrm{Cb} - 128 \\ \mathrm{Cr} - 128 \end{bmatrix} \tag{3.11}$$

If all the direct superpixel neighbors would be used for texture restoration, any of them could distort the repair. Our method controls the quality by restricting the extrapolation to derive



(a) (b)

Figure 3.20: Successive Discrete cosine transform (S-DCT) extrapolation from all surrounding superpixels (a) compared to a selection based on superpixels merging (b). MSE significantly dropped after removing the left-most superpixel.

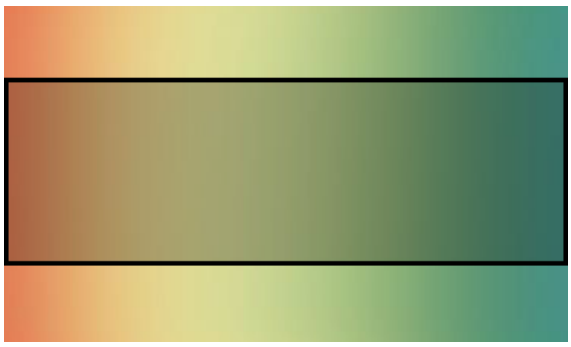
from homogeneous regions only. Moreover, the processing of homogeneous and smaller regions requires less time to reach the required PSNR.

In Fig. 3.20 the red superpixel gets repaired by extrapolation from the superpixels marked blue. On the left, the left-most superpixel matching the dark wing mixed its texture into the blurred background during the extrapolation. On the right, removal of the problematic superpixel leads to an almost perfect reconstruction.

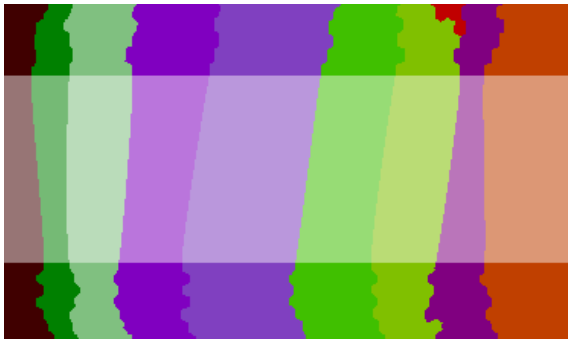
A precise segmentation of textures is very important, but not always possible. Discrete segmentation fails for example for unsharp edges where it introduces errors at the segment borders which propagate further during the restoration (see Fig. 3.21 (b, d)). Therefore we proposed to use the fuzzy segmentation that helps to correctly restore gradients and unsharp edges (see Fig. 3.21 (c, e)). Neighboring segments often overlap in the fuzzy segmentation approach. The final color for pixels that are covered by more than one segment is computed as a normalized sum of the extrapolated data weighted by the respective certainty values (see section 3.2.3).

3.2.5 Method Evaluation

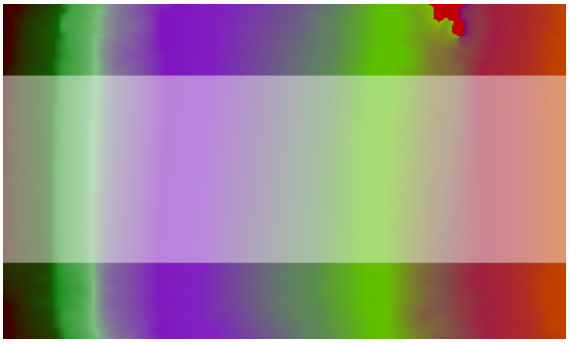
To assess the quality of our error concealment method we tested it on various images with different localized errors and we compared our results with results of the three state of the art image inpainting methods: simple patch-based method [16] and the sophisticated patch



(a) Original gradient image with marked corrupted area



(b) Simple segmentation with highlighted repaired area



(c) Fuzzy segmentation with highlighted repaired area

(d) Result of our original method [68] using the simple segmentation from (b)

(e) Result of the proposed method using the fuzzy segmentation from (c)

Figure 3.21: Results comparison of our error concealment method with and without fuzzy segmentation in gradient image. Our original method with simple segmentation (b, d) resulted into many artifacts. The result of the proposed method with fuzzy segmentation (c, e) has very good visual quality.

and group-based methods [17, 84]. For the objective evaluation of the quality of the achieved results we used four widely used quality measurement methods that we describe in following section. In section 3.2.5.2 we provide results of our experiments.

3.2.5.1 Video and Image Quality Measurement

It is necessary to compare the various concealment methods by some objective criteria. We have to determine the quality of the images obtained by concealment with different techniques of the same corrupted video / image to decide which one is closer to the original. Measuring of the visual quality is a difficult and often imprecise art because there are many factors that can affect the results. Visual quality is strongly subjective because it is influenced by many subjective factors. Viewer's rating of visual quality depends very much on the specific task, such as passively watching a movie, actively participating in a video conference or trying to identify a person in a surveillance video scene. Measuring the visual quality using objective criteria gives accurate, repeatable results. But there are not objective measurement systems that would completely reproduce the subjective experience of a human observer.

Many objective measurement methods have been developed that can more or less reflect the impact of concealed images and videos on the human visual system (HVS). Some of these methods can be fully described by a computable analytical model, which makes them very easy to use.

Let us define an image as

$$I(x,\, y) = (\mathrm{R},\, \mathrm{G},\, \mathrm{B}), \tag{3.12}$$

where x, y are spatial coordinates ranging from 0 to image width $w-1$ and image height $h-1$, respectively and R, G and B are components of RGB color space ranging from 0 to 255 for 24-bit color images. The function $I(x,\, y)$ assigned to each image pixel some RGB color. Quality measurements are usually applied only to luminance component of the color space and they are evaluated for whole image and not just for the concealed corrupted parts of the image. We used the following conversion from RGB color space to Y component of YUV color space:

$$\mathrm{Y} = 0.2989 \times \mathrm{R} + 0.5870 \times \mathrm{G} + 0.1140 \times \mathrm{B} \tag{3.13}$$

Then the **Mean Squared Error (MSE)** is defined as

$$\mathrm{MSE} = \frac{1}{w \times h} \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} (\mathrm{Y}_{original}(i,\ j) - \mathrm{Y}_{concealed}(i,\ j))^2, \tag{3.14}$$

where $\mathrm{Y}_{original}(i,\ j)$ is the luminance value in YUV color space for pixel $(i,\ j)$ in original uncorrupted image and $\mathrm{Y}_{concealed}(i,\ j)$ is the luminance value in YUV color space for pixel $(i,\ j)$ in concealed corrupted image. MSE metric is one of the standardized video and image error measurements and due to its simplicity it is widely used in practice.

Peak Signal to Noise Ratio (PSNR) is measured on a logarithmic scale and depends on the mean squared error between the original and concealed corrupted image (or video frame), relative to the square of the highest possible signal value in the image, that is given as $(2^n - 1)^2$, where n is the number of bits per image sample (in our case $n = 8$, because we converted the 24-bit color image to the 8-bit grayscale image):

$$\mathrm{PSNR} = 10 log_{10} \frac{(2^n - 1)^2}{\mathrm{MSE}} \tag{3.15}$$

Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, provided the bit depth is 8 bits, where higher is better [9]. Acceptable values for wireless transmission quality loss are considered to be about 20 to 25 dB [36]. Since PSNR and MSE are based on a comparison of individual pixels, they have a limited relationship with distortion or perception of the quality by the human visual system [78].

Xiao [80] proposed a modified **DCT-based video quality metric (VQM)** based on Watson's DVQ model [76], which exploits a property of visual perception. Human sensitivity to spatio-temporal pattern decreases with high spatial and temporal frequency [80]. Therefore, we can represent high spatial and temporal information with less precision while human's eyes are not sensitive to the loss of this information. And DCT quantization exploits this property directly.

In Fig. 3.22 there is displayed a block diagram of VQM. It starts with image / video frame transform to YUV color space. After that the DCT transform is applied. Third step is the conversion of each DCT coefficient to local contrast (LC) using following equation:

$$\mathrm{LC}(i,\ j) = \mathrm{DCT}(i,\ j) \frac{\sqrt[0.65]{DC/1024}}{DC}, \tag{3.16}$$

where DC is the DC component of each block. For 8-bit image, 1024 is mean DCT value. 0.65 is the best parameter for fitting psychophysics data [80]. After this step, most values

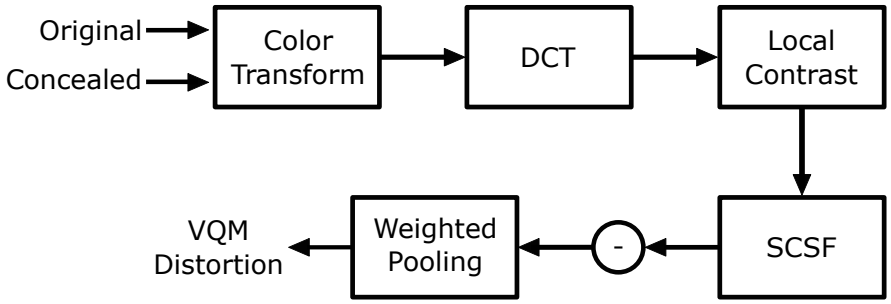


Figure 3.22: Block diagram of VQM [80]

lie between -1 and 1. The first three steps are identical to Watson's DVQ model [76]. The next step is different from Watson's model. Instead of applying temporal filtering and human spatial contrast sensitivity function (SCSF) separately, Xiao chose to apply one SCSF matrix for static frames and one matrix for dynamic frames in one step to reduce the computation and memory load. The DCT coefficients are converted to just-noticeable differences by multiplying each DCT coefficient by its corresponding entry in the SCSF matrix.

In the last step called weighted pooling the two sequences are subtracted first. At this step VQM also differs from DVQ by incorporating contrast masking into a simple maximum operation and then weights it with the pooling mean distortion. This reflects the facts that a large distortion in one region will suppress human sensitivity to other small distortion, for this kind of situation, weighted maximum distortion into pooled distortion is much better than pooled distortion alone.

$$\text{VQM} = (Mean_{dist} + 0.005 \times Max_{dist}), \tag{3.17}$$

where

$$Mean_{dist} = 1000 \times mean(mean(abs(diff))) \tag{3.18}$$

and

$$Max_{dist} = 1000 \times maximum(maximum(abs(diff))) \tag{3.19}$$

Maximum distortion weight parameter 0.005 is chosen based on several primitive psychophysics experiments. Parameter 1000 is the standardization ratio.

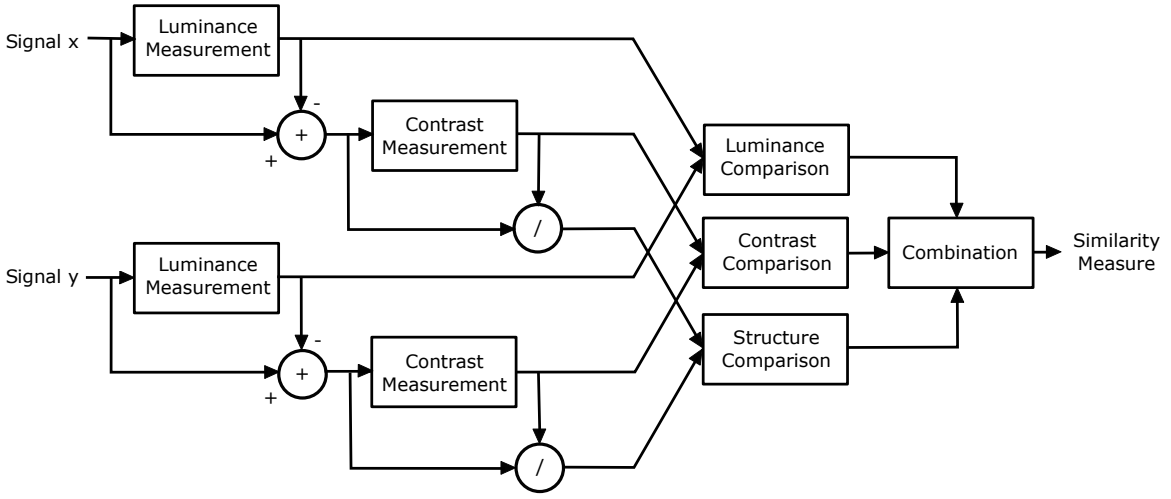


Figure 3.23: Diagram of the structural similarity (SSIM) measurement system [75].

Xiao based on his experiments determined the threshold for detecting a distortion to 1 unit. Thus VQM lower than 1 means not noticeable distortion. Starting from 1 and going higher the distortions are noticeable and higher number means bigger distortion. VQM was standardized in 2003 by the organization ANSI [4] and in 2004 by the organization ITU-T [1, 2].

Structural Similarity (SSIM) index evaluates the visual impact of shifts in the luminance of the image, changes of the contrast and other defects known as the structure changes. SSIM quality measurement method is based on the premise that the human visual system is highly adapted for extracting structural information of the scene. Measurement of structural distortions should provide a better correlation with subjective perceptions [19, 82].

The SSIM quality assessment index is based on the computation of three terms, namely the *luminance* term, the *contrast* term and the *structural* term (see Fig. 3.23). The overall index is a multiplicative combination of the three terms [75]:

$$\mathrm{SSIM}(x,\, y) = [l(x,\, y)]^{\alpha} \times [c(x,\, y)]^{\beta} \times [s(x,\, y)]^{\gamma}, \tag{3.20}$$

where

$$l(x,\, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \tag{3.21}$$

$$c(x,\, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \tag{3.22}$$

$$s(x,\, y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3}, \tag{3.23}$$

where μ_x, μ_y, σ_x, σ_y, and σ_{xy} are the local means, standard deviations, and cross-covariance for images $x,\, y$.

If $\alpha = \beta = \gamma = 1$ (the default for Exponents), and $C_3 = C_2/2$ (default selection of C_3) the index simplifies to:

$$\mathrm{SSIM}(x,\, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.24}$$

SSIM has several properties:

- symmetry: $\mathrm{SSIM}(x,\, y) = \mathrm{SSIM}(y,\, x)$
- boundedness: $\mathrm{SSIM}(x,\, y) \leq 1$
- unique maximum: $\mathrm{SSIM}(x,\, y) = 1 \iff x = y$

SSIM provides greater consistency with HVS as PSNR, even that it fails in assessing the quality of images that are too blurred [72]. SSIM index belongs to the frequently used methods for the objective evaluating of the video and image quality.

3.2.5.2 Experimental Results

The main goal of our research is to propose an efficient method for fast texture aware error concealment of color images and video frames. Experiments indicate that the presented method exceeds the quality of comparably fast frequency selective extrapolation methods [54, 34], simple patch-based methods [15, 16] and our original method with simple segmentation [68]. At the same time it outperforms the sophisticated patch and group-based methods [17, 84] in time needed to achieve results of a comparable quality. Experiments were conducted on images degraded with artificially generated localized errors using an Intel Xeon X5550. Tables 3.1 and 3.2 provide a comparison of objective error measurements (measured by MSU Video Quality Measurement Tool [3]) and timings for all presented results. Using the standard error measurements techniques implies a comparison to the ground truth. However, we believe that concealment should be also judged by the subjective impression of the image consistency regardless of the original picture. The measured timings exclude the image over-segmentation step which can be implemented in real-time on the GPU [13].

Table 3.1: Error measurements and timings for the fast methods in Fig. 3.24, 3.25, 3.26 and 3.27.

	Concealment Method	Quality Measures (whole image)				Time [min]
		MSE	PSNR	VQM	SSIM	
Cardinal	Without segmentation	**30.11**	**33.34**	**1.15**	**0.9320**	**1:41**
	GIMP	10.89	37.76	1.04	0.9768	**0:01**
	Simple Segmentation	11.47	37.54	0.75	0.9738	0:04
	Fuzzy segmentation	**10.57**	**37.89**	**0.73**	**0.9780**	0:09
Lenna	Without segmentation	79.73	29.11	2.88	**0.9265**	**2:43**
	GIMP	**103.20**	**27.99**	**3.49**	0.9415	**0:02**
	Simple Segmentation	31.08	33.21	1.94	0.9586	0:05
	Fuzzy segmentation	**26.40**	**33.91**	**1.55**	**0.9633**	0:16
Swan	GIMP	**159.73**	**26.10**	**5.32**	**0.9630**	**0:01**
	Simple Segmentation	**9.76**	**38.24**	**0.84**	**0.9858**	00:18
	Fuzzy segmentation	9.80	38.22	1.09	0.9840	**00:30**
Raft	GIMP	**51.54**	**31.01**	**3.00**	**0.9678**	**0:02**
	Simple segmentation	**37.13**	**32.43**	**2.24**	**0.9698**	**0:28**

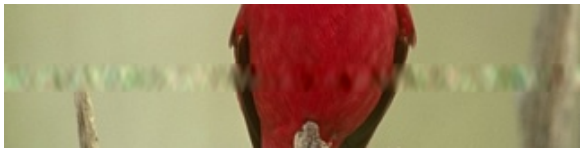
In our experiments we used successive Discrete cosine transform (S-DCT) and successive separable 2D Hartley Transform with sequentially ordered basis functions [41]. Hartley transform in 2D corresponds to its separable 1D realization for rows and subsequently for columns. We will refer to it as cas-cas [47] and abbreviate it as DCCT. The acronym S-DCCT corresponds to successive (iterative) approximation using the DCCT basis functions in the sequential order.

3.2.5.2.1 Comparison with Fast Methods

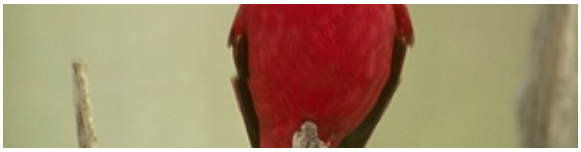
Frequency selective methods extrapolate every pixel falling within the rectangular surroundings of the lost area. Consequently, any object within the source window contributes to the frequency-based approximation. Unrelated features are often mixed between different regions. If the source pixels for the reconstruction are limited to the same segment and its shape is concealed beforehand, the tone and texture of the repaired area keeps similar attributes. Fig. 3.24(b) shows S-DCT-based and Fig. 3.25(b) S-DHYT-based error concealment compared to the results of our method with segmentation (f). Segmentation minimized the texture extrapolation artifacts.

On the last four images in Fig. 3.24, 3.25 and 3.26 we compare results of our method with simple segmentation and fuzzy segmentation. Fuzzy segmentation outperforms simple segmentation in error concealment of images with gradients and soft edges. The most significant

(a) Original image of bird cardinal with marked corrupted area



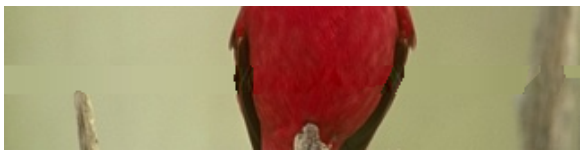
(b) Direct S-DCT extrapolation from whole surrounding of the corrupted area



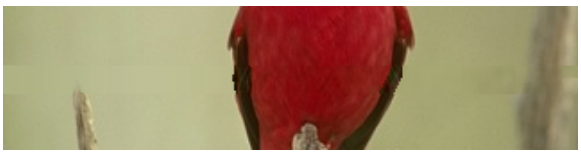
(c) The result of method proposed in [16] implemented in graphic software GIMP [25]

(d) Simple segmentation with highlighted repaired area

(e) Fuzzy segmentation with highlighted repaired area

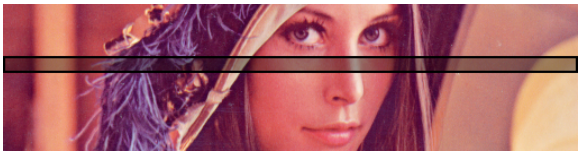


(f) Result of our original method [68] using the simple segmentation from (d) and S-DCT

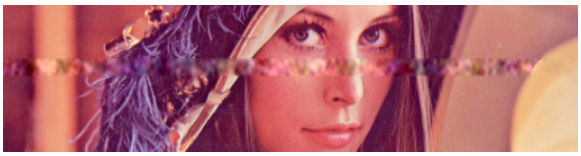


(g) Result of proposed method using the fuzzy segmentation from (e) and S-DCT

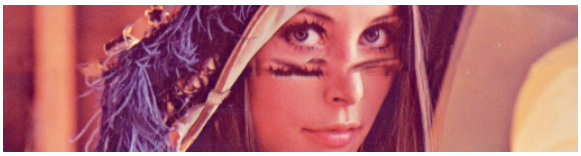
Figure 3.24: Results comparison of fast methods for the image of the cardinal bird. S-DCT extrapolation from whole surroundings of the corrupted area (b) mixed textures together. Method implemented in GIMP (c) did not connect the left bird wing very well. Our original method with simple segmentation (d, f) connected wings of the bird better, but it was not able to maintain soft edges in the right part of the image. The proposed method with fuzzy segmentation (e, g) significantly improved the main drawback of the original method. Table 3.1 provides a comparison of objective error measurements and timings for all results.



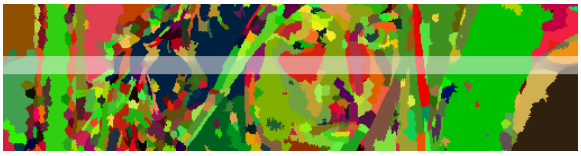
(a) Original image of Lenna with marked corrupted area



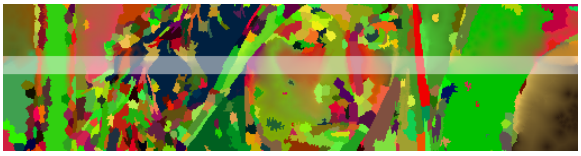
(b) Direct S-DHYT extrapolation from whole surrounding of the corrupted area



(c) The result of method proposed in [16] implemented in graphic software GIMP [25]

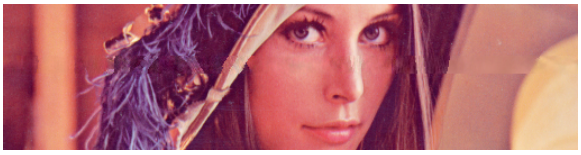


(d) Simple segmentation with highlighted repaired area



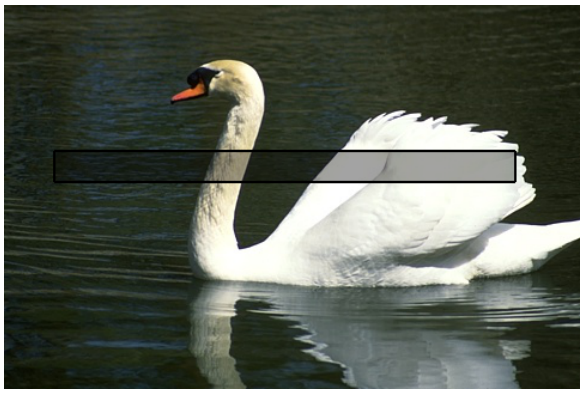
(e) Fuzzy segmentation with highlighted repaired area

(f) Result of our original method [68] using the simple segmentation from (d) and S-DHYT



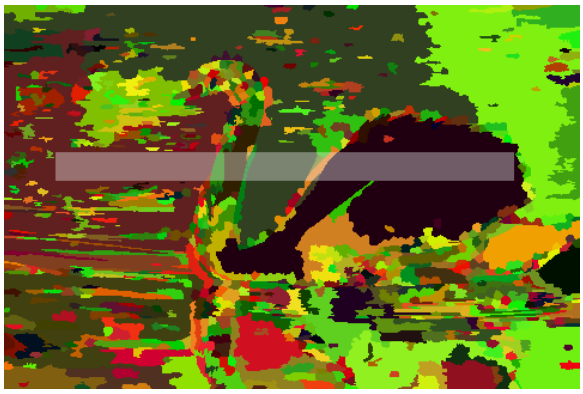
(g) Result of proposed method using the fuzzy segmentation from (e) and S-DHYT

Figure 3.25: Results comparison for the image of Lenna. S-DHYT extrapolation from whole surroundings of the corrupted area (b) mixed textures together. GIMP method (b) copied many artifacts into the face of Lenna. Our original method with simple segmentation (d, f) was not able to maintain soft edges and gradients and it resulted into many artifacts. The proposed method with fuzzy segmentation (e, g) radically reduced the artifacts and it achieved almost perfect result. Table 3.1 provides a comparison of objective error measurements and timings for all results.

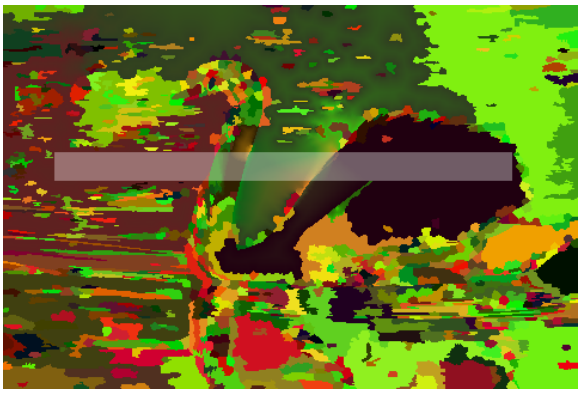


(a) Original image of swan with marked corrupted area

(b) The result of method proposed in [16] implemented in graphic software GIMP [25]



(c) Simple segmentation with highlighted repaired area



(d) Fuzzy segmentation with highlighted repaired area

(e) Result of our original method [68] using the simple segmentation from (c) and S-DHYT

(f) Result of proposed method using the fuzzy segmentation from (d) and S-DHYT

Figure 3.26: Results comparison for the image with swan. GIMP method (a) copied many artifacts into the body of the swan but it was able to nicely copy a texture of water. In our approach the pattern of water texture was separated into many segments with homogeneous color and therefore our original method with simple segmentation (c, e) was not able to extrapolate the pattern of water texture. On the other hand the concealment of the body of swan has very good visual appearance. Additionally error measurements in table 3.1 says that it is even the best result from all three methods. The proposed method with fuzzy segmentation (d, f) combined the texture of water from few segments what resulted into more realistic water texture, but it blurred a bit edge between swan body and water.

(a) Original image with raft from Kodak set [33] with marked corrupted area

(b) The result of method proposed in [16] implemented in graphic software GIMP [25]

(c) Our original method with simple segmentation and S-DCT extrapolation

Figure 3.27: Results comparison of the method proposed by Daisy et al. [16] and our original method for the image with raft. Zoomed part of the image shows that our method provides better visual impression compared to Daisy et al. method [16] that copied into the result several unfitting parts from the surroundings of the corrupted area.

Table 3.2: Error measurements and timings for the sophisticated methods in Fig. 3.28, 3.29, 3.30 and 3.31.

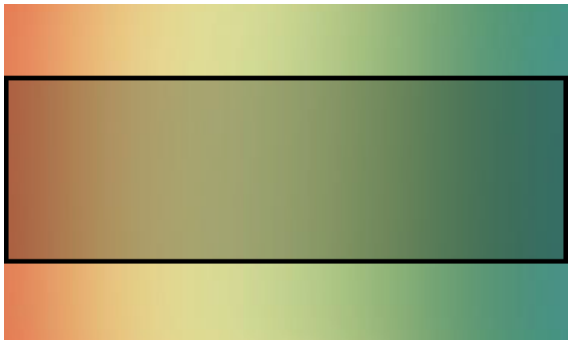
	Concealment Method	Quality Measures (whole image)				Time [min]
		MSE	PSNR	VQM	SSIM	
Gradient	Image melding	**2.30**	**44.51**	**0.33**	0.9963	11:36
	GSR	**0.97**	**48.28**	**0.25**	**0.9921**	**16:06**
	Our method	1.91	45.32	0.31	**0.9965**	**0:13**
Cardinal	Image melding	**3.28**	**42.97**	**0.41**	**0.9873**	2:22
	GSR	6.52	39.99	**0.89**	0.9839	**6:30**
	Our method	**10.57**	**37.89**	0.73	**0.9780**	**0:09**
Lenna	Image melding	**20.03**	**35.11**	**1.50**	**0.9749**	6:42
	GSR	**29.00**	**33.51**	**1.74**	0.9689	**11:18**
	Our method	26.40	33.91	1.55	**0.9633**	**0:16**
Raft	Image melding	29.75	33.40	2.65	**0.9755**	10:14
	GSR	**26.54**	**33.89**	**2.84**	0.9754	**11:06**
	Our method	**37.13**	**32.43**	**2.24**	**0.9698**	**0:28**

difference can be seen in the concealment of the Lenna face in Fig. 3.25(d-g). By using fuzzy segmentation we get better subjective results as you can see in the figure and also better objective results as you can see in the Table 3.1. In all four quality measures: Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), DCT-based video quality metric (VQM) and Structural similarity (SSIM), the method with fuzzy segmentation achieves better results. On the other hand the time required to conceal errors has increased, as texture approximation of larger regions takes longer.

Implementation of the fast patch-based method [15] with the addition of spatial blending technique [16] in the graphic software GIMP [25] provides quite good results for simple images (see the image of cardinal bird in Fig. 3.24(c)), but in the case of complex images (the image of Lenna in Fig. 3.25(c), swan in Fig. 3.26(b) and raft in Fig. 3.27(b)), we can see several artifacts in the result. The artifacts appear due to copying patches from surroundings of the corrupted area. Our method avoids such artifacts by using segmentation.

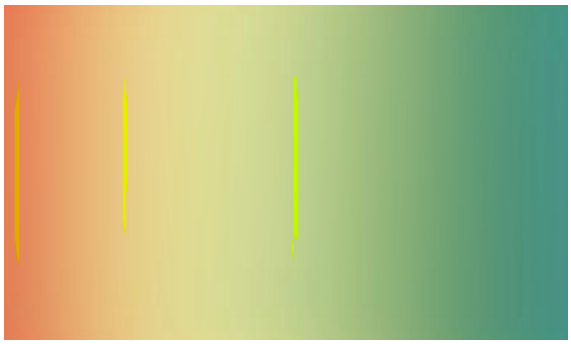
3.2.5.2.2 Comparison with Sophisticated Inpainting Methods

The main part of the experiments was conducted to compare the performance and restoration quality of our method with the state of the art methods: *Image Melding* [17] and *Group-based Sparse Representation for Image Restoration* (GSR) [84].



(a) Original gradient image with marked corrupted area

(b) Image Melding method [17]

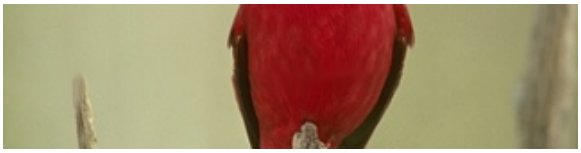


(c) GSR method [84]

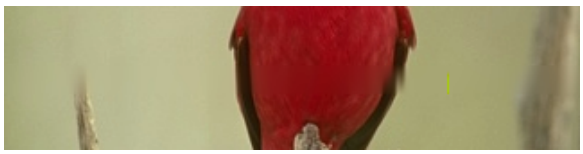
(d) Result of the proposed method using the fuzzy segmentation and S-DCT extrapolation

Figure 3.28: Results comparison for the gradient image. Image Melding method (b) visually provides very good result. The GSR method (c) resulted into some artifacts. The proposed method (d) has nearly the same visual quality as Image Melding and in quality measurements in table 3.2 it achieved even better results. On the other hand according to the quality measures the GSR method is better than our method except of the SSIM index comparison.

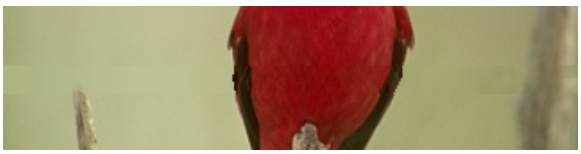
(a) Original image of bird cardinal with marked corrupted area



(b) Image Melding method [17]



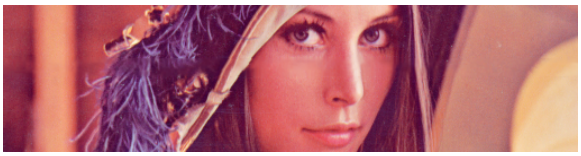
(c) GSR method [84]]



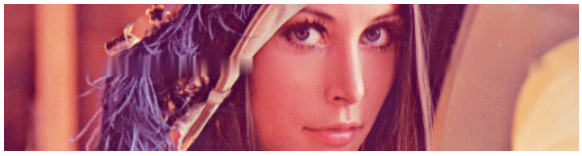
(d) Result of proposed method using the fuzzy segmentation and S-DCT extrapolation

Figure 3.29: Results comparison for the image of the cardinal bird. Image Melding (b) provides very good result. The GSR method (d) has problems with the right wing of the bird. Our method (d) nicely connected wings and in table 3.2 it can be seen that the time required to achieve comparable quality results is considerably shorter.

(a) Original image of Lenna with marked corrupted area



(b) Image Melding method [17]



(c) GSR method [84]

(d) Result of proposed method using the fuzzy segmentation and S-DHYT

Figure 3.30: Results comparison for the image of Lenna. Image Melding (b) has a very good result. GSR (c) blurred some parts of the concealment result. The proposed method with fuzzy segmentation (d) has visually better result compared to GSR method because there are not blurred areas. Table 3.2 provides a comparison of objective error measurements and timings for all results.

(a) Original Raft image with marked corrupted area

(b) Image Melding method

(c) GSR method

(d) Our original method with simple segmentation and S-DCT extrapolation

Figure 3.31: Results comparison for the Raft image. The zoomed wild water shows that our method provides the best visual impression within the shortest time for this image. Table 3.2 provides objective error measurements and timings.

In case of the gradient image (Fig. 3.28) the proposed method has nearly the same visual quality as Image Melding method and the objective error measurements comparison in Table 3.2 shows that our method is even better than the Image Melding method in quality and also in the time required to compute the result. The GSR method performs better in quality measurements, but it has a visually worse result because of the artifacts and also the runtime is more than 70× longer.

For the image of the cardinal bird (Fig. 3.29) the Image Melding method clearly provides the best result. The GSR method has problems with the right wing of the bird, but the quality measured by objective methods is still slightly better compared to our method. On the other hand, our method required only several seconds to conceal the corrupted region compared to the several minutes runtimes of Image Melding and GSR methods.

The face of Lenna (see Fig. 3.30) has many soft gradients but also fine features which are difficult to be segmented. With the proposed fuzzy segmentation we were able to avoid artifacts in the reconstructed face. We get better visual results compared to GSR which blurred the hat of Lenna and some other parts of the image as well. Image Melding achieves the best quality, but its runtime is not feasible in real-time.

The Raft image [33] contains both repetitive textures and unique objects (see Fig. 3.31). Being 20× faster, our method still provides comparable visual quality in the corrupted stripe. Even more, it reconstructs the water texture the best.

Chapter 4

Texture Classification of Arbitrarily Shaped Regions with an Application in Image Segment Merging

Image over-segmentation as a pre-processing step of image segmentation splits the input image into superpixels that are small compact regions with irregular shapes. The majority of existing methods for texture feature extraction are not suitable for arbitrarily shaped regions. Therefore, only color information can be used to classify and merge superpixels to create final image segmentation. We propose a novel method for extracting texture features of arbitrarily shaped image regions using orthogonal transforms. We introduce a mathematically correct method for unifying spectral dimensions that is necessary for accurate comparison and classification of spectra with different sizes. Our approach is based on the properties of certain orthogonal transforms when inserting zeros into the spectrum. We found out which orthogonal transforms possess this important property and also provide mathematical proofs for our claims. The proposed method is particularly suitable for classifying areas with periodic and quasiperiodic textures. However, by utilizing luminance and chrominance this method can be applied to and is suitable for any type of images.

In the following section we present a detailed analysis of the related work along with the description of advantages, disadvantages and applicability of the presented techniques divided into widely accepted categories: geometrical approach, model-based approach, statistical approach and signal processing approach. Section 4.2 first introduces the preliminary conditions for application of our proposed novel method for texture features extraction along with a detailed description of the important concepts such as basis set construction and spectral dimension unification. Experiments on the luminance components of textures and a proposal for construction of an optimal feature set for classification algorithms that are used to merge

superpixels into segments are also located in section 4.2. Evaluation of the proposed method in corrupted images and comparison of using various color spaces are located in the last section (4.3) of this chapter.

4.1 Related Work

Shapiro et al. [55] distinguish two principal approaches to texture analysis: structural and statistical. The *structural* approach defines textures as sets of primitives or elements, called texels. Texels are regularly or nearly regularly distributed. The *statistical* approach defines a texture as a quantitative measure of the intensities arrangement within a given area. While the structural approach works sufficiently well for regular texture patterns, the statistical approach, being simpler and more general, is more frequently used in practice.

Tuceryan and Jain [67] categorize texture analysis methods into four groups: geometrical, model-based, statistical and signal processing methods. Using their categorization we reviewed some representative methods to identify their advantages, disadvantages and their applicability to arbitrarily shaped regions.

4.1.1 Geometrical Method

Tuceryan and Jain [66] proposed one of the first texture analysis methods. It uses geometry-based description of texture structure which starts by finding texels – small image regions extractable through a simple process such as thresholding. The description of spatial relationship between neighboring texels is provided by a Voronoi tessellation of all texels. The shape properties of the polygons in Voronoi tessellation are computed and used to cluster the polygons into similarly textured regions. The algorithm is sensitive to small perturbations in the texel locations if the texel placement is regular [66]. Therefore it is not often used in practice.

4.1.2 Model-Based Methods

Model-based texture analysis methods create a model of an image that can be used both to describe and to synthesize a texture [67]. The model should capture the substantial perceived properties of the texture. Tuceryan and Jain [67] include Markov random fields and fractal geometry in model-based methods. These methods are rather used for image modeling than for texture classification.

4.1.3 Statistical Methods

Statistical texture analysis methods deal with the spatial distribution of gray levels (or colors) within a texture [55, 67, 60].

Local binary patterns (LBP) introduced by Ojala et al. [39] represent a simple methodology which extracts features from a local binary partition. An eight digit binary number is computed for each pixel p in the image. It is obtained by thresholding eight neighboring pixels of p with the threshold set to the intensity of p. Once the whole image is analyzed, the histogram of the obtained eight bit numbers represents the texture of the image. This method is fast and therefore also used in many applications where time is a crucial factor. Its further advantage is that it can be used to classify arbitrarily shaped regions. On the other hand, it extracts only local features, thus it neither captures wider surroundings nor produces the best results.

Co-occurrence matrices were introduced by Haralick et al. [27]. An element of co-occurrence matrix in i-th row and j-th column specifies how many times the value i (usually a gray level value) of the image co-occurs with the value j in some defined spatial relationship, e.g. for a pair of horizontally neighboring pixels the value of the right one is j and the value of the left one is i. The spatial displacement is defined by a displacement vector (r, c), where r is the displacement in rows (downward) and c is the displacement in columns (to the right). Therefore, multiple co-occurrence matrices can be generated for a single image.

Co-occurrence matrices can be used to classify arbitrarily shaped regions while being able to capture the wider surroundings. However, it is up to the user to specify what displacement vectors should be used to analyze possibly larger texture patterns.

4.1.4 Signal Processing Methods

Spatial domain filters like Roberts, Sobel and the Laplacian filter are used to find edges or their directions in the input image. The idea behind these filters is to approximate the gradient of an image through discrete differentiation. Edge density per unit area or density of edges with specified directions are properties that can be used for very simple classification of textures [24, 67, 44, 60]. These filters could be used to classify arbitrarily shaped regions, however, e.g. Roberts filter suffers greatly from sensitivity to noise [18].

Fast Fourier transform (FFT), Discrete cosine transform (DCT), Discrete Walsh-Hadamard transform (WHT) and Discrete Hartley transform (DHYT) are examples of frequency domain filters. Fourier transform represents frequency components of the time function by a

sum of sine waves of different frequencies. The frequency components spectrum is the signal representation in the frequency domain.

Monadjemi [38] proposed the Directional Walsh-Hadamard transform (DWHT) method which uses oriented Hadamard based features to represent the directionality of a texture. In his approach, the Hadamard matrix stays constant but the image function is, as Monadjemi says, *rotated* by $\alpha = (0°, 45°, 90°, 135°)$. However, it is not a standard geometrical rotation. E.g. rows of the $A_{45°}$ image are created by taking the pixels that sit in a diagonal (45°) direction in the image $A_{0°}$. Therefore, rows of $A_{45°}$ contain the information at the defined directions (angles). The extraction of row sequence information takes out the DWHT feature sets that are used to classify texture. This method does not examine individual spectral components, but only mean energy of directional sub-bands and therefore it is not accurate.

Image segmentation method proposed by Pun and Zhu [46] splits the input image into $M \times N$ blocks of small size. DCT is applied on each block to calculate its energy feature that is defined as the sum of a specified number of largest coefficients excluding the upper-left one. The similarity distance between two blocks is then defined as the absolute value of difference between their energy features. The main drawback of this method is that it splits the image into square blocks. Therefore, boundaries of the final segments consist of line segments and do not follow salience edges of the image. Additionally the sum of few largest DCT coefficients is not very meaningful texture feature.

Pun et al. [45] replaced regular blocks by small regions created by a watershed transformation of the input image. The regions have irregular shapes and the DCT has to be implemented on rectangle. Therefore, they proposed to find the maximum rectangle in a region for substituting the texture of the whole region. Such substitutions often cause that certain texture features are omitted, leading to accuracy loss. Another issue is that the rectangles do not have same dimensions and therefore it is impossible to directly compare their spectral coefficients. Pun et al. decided to divide DCT matrices into 9 parts by splitting each matrix into 3 rows and 3 columns with even distributions. Then, they sum absolute values of all elements in every part to get a nine dimensional feature vector of the matrix. However, matching of the feature vectors obtained by averaging parts of DCT spectrum with larger and smaller dimensions is mathematically incorrect.

4.2 Proposed Method for Feature Extraction

The basic assumptions of our proposed method for feature extraction are:

1. Analyzed areas have arbitrary shapes, generally non-rectangular
2. The areas have a texture structure inside. The textures tend to have mostly high frequency spectra, in small areas these can also be considered periodic or at least quasiperiodic.
3. A real and sensible possibility to inscribe a rectangle into the area is limited.

4.2.1 Basic Assumptions and Method Description

The proposed method solves the problem of partitioning a 2D (after generalization also n-dimensional) discrete image into multiple logically associated classes with the goal of establishing proper boundaries among various image elements. Furthermore it groups segments forming the elements into classes with shared properties. In order to establish a mathematically tractable result, the entire process is partitioned into multiple stages to address all relevant issues.

Similarly, as Pun et al. [45], the first step of the proposed method is to pre-process the image using an efficient image over-segmentation technique such as SLIC [5] or gSLIC [48]. In terms of the over-segmentation parameters, the segment sizes should have as little variance as possible to decrease the computational demand in the latter stages. This is however not a strong condition, as the method is capable of processing also segments with varying sizes.

Arbitrarily shaped segments represent a 2D image cutout containing internal texture structure. Pun et al. [45] use the maximum rectangle inscribed in the various segments to obtain the texture sample from within the segment in order to optimize processing efficiency and implement fast transform algorithms. For the same reasons and to avoid any possible information loss that might occur, the arbitrarily shaped segments are in our method padded with zeros in both dimensions to form the smallest circumscribing rectangle with both dimensions being power of two. More details about this process and selection of suitable basis functions we have already described in section 3.2.4.1, where we applied orthogonal transforms to final segments. Here we apply the transforms to superpixels in order to classify them and merge into segments. Each superpixel is transformed using discrete Walsh-Hadamard Transform (WHT) or Separable Hartley Transform (CAS) [77] producing sets of YUV segment spectra (similar to use of DCT in work of Singh [57] or Pun and Zhu [46]). In following section 4.2.2 we compare extrapolation capability of five different orthogonal transforms. Different transforms were purposefully chosen in order to create a basis for comparison of the proposed method's efficiency and reliability.

The next phase consists (in case of orthogonal transforms) of normalizing the spectra to the block size, where the block size is the desired spectral dimension corresponding to the original segment size. If the segment size condition during the segmentation process is not respected, the spectra must be modified to reflect this reality. If the spectrum's total number of rows or columns is greater then the desired value, dimensional reduction is used, if the total number of rows or columns is smaller than the desired value, dimensional expansion takes place. Dimensional reduction however necessarily removes part of the spectral information, while dimensional expansions retains it. Proofs of spectral row and column reduction and extension for WHT and CAS is located in section 4.2.3.

K-means based classification requires vectors, therefore the obtained spectra must be transformed into vectors using any of the number of available techniques. In order to respect the frequency order, zig-zag method [81] was chosen in our implementation. Complications only arise in case of CAS spectra, which are naturally ordered in a manner that would not produce a sequence of continually increasing component frequencies after applying zig-zag. Therefore spectral reordering of CAS spectral coefficients must take place first [41]. Zig-zag is then applied to obtain a vectorized form of spectral coefficients where the DC coefficient is the first value and highest component frequencies are the last values contained in the vector.

Vectorized spectra of all segments are input into the k-means classification. Multiple classification metrics can be used, however the best results are obtained with the Correlation, Cosine and Squared Euclidean metrics, that are Mean Squared Error (MSE) metrics, while the City Block metric, that is Mean Absolute Error (MAE) metric, fails to produce precise results [8].

4.2.2 Selection of Orthogonal Transforms

In order to select orthogonal transforms with best possible extrapolation results we extrapolated areas with various textures by five different orthogonal transforms. Results are shown in Fig.4.2. For experiments the synthetically composed textures from the Brodatz [14] and Simoncelli [43] databases were used. To split images composed of textures into small textured segments we used synthetic over-segmentation mask that can be found in Fig. 4.1. Based on obtained results we decided to use successive Separable 2D Hartley Transform (S-DCCT) and successive discrete Walsh-Hadamard transform (S-WHT) in our proposed method.

An important complication arises when the feature set is represented by magnitudes of the spectral components - all of the areas must use the same number of coefficients. From a mathematical perspective, specific position in the feature vector must correspond to the same basis function. Using the described method rectangular areas with different sizes can

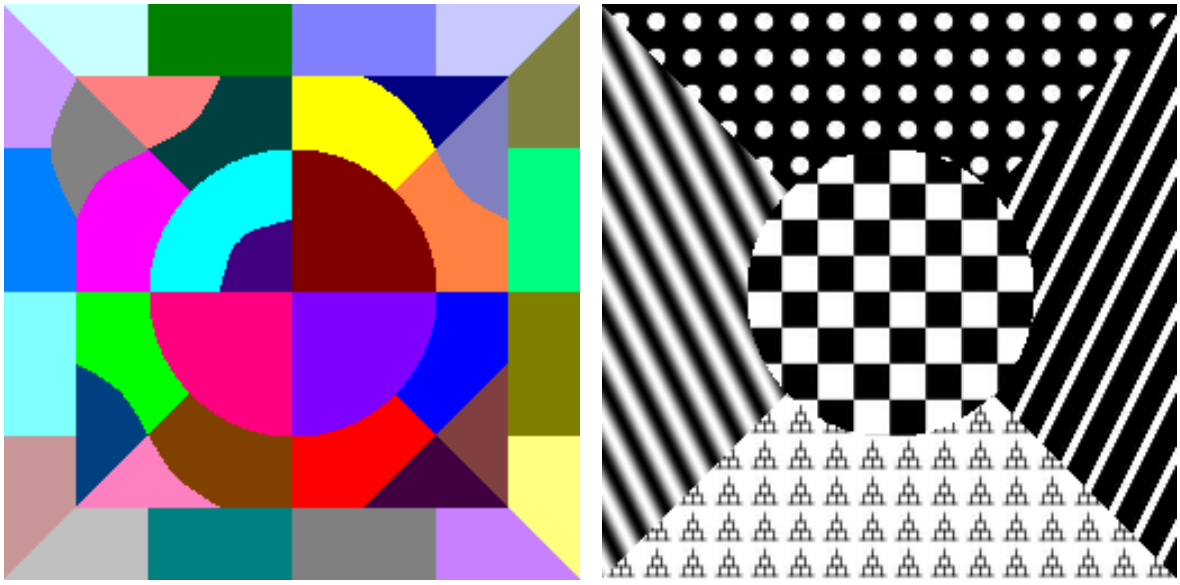


Figure 4.1: Over-segmentation mask (left) used to split image composed of periodic textures (right) into small textured segments. The same mask was also used for textured images in Fig. \ref{fig:Results:-First-two

be produced and therefore they must be unified. One of the possible solutions to this problem is exploiting the properties inherent to certain transforms, such as the property of periodification in spatial domain by extending the spectrum with alternating zeros e.g. in Discrete Fourier transform (DFT) method. Not every orthogonal transform possesses this property, not even the most commonly used DCT II.

The next section contains a proof that the Hartley transform indeed possesses this property which in 2D corresponds to its separable realization and we will refer to it as cas-cas and abbreviate it as DCCT. For the realization with successive calculation and extrapolation we are using the acronym S-DCCT. Another transformation that possesses this important property is the discrete Walsh-Hadamard transform (WHT). However, it should be noted that each of these transforms requires a different method for spectral periodification.

4.2.3 Unification of the Spectral Dimensions

In this section we provide mathematical proofs of the property used for unification of spectral components for different sized areas.

4.2.3.1 Discrete Walsh-Hadamard Transform - Natural (Hadamard) Order of Basis Functions

Let

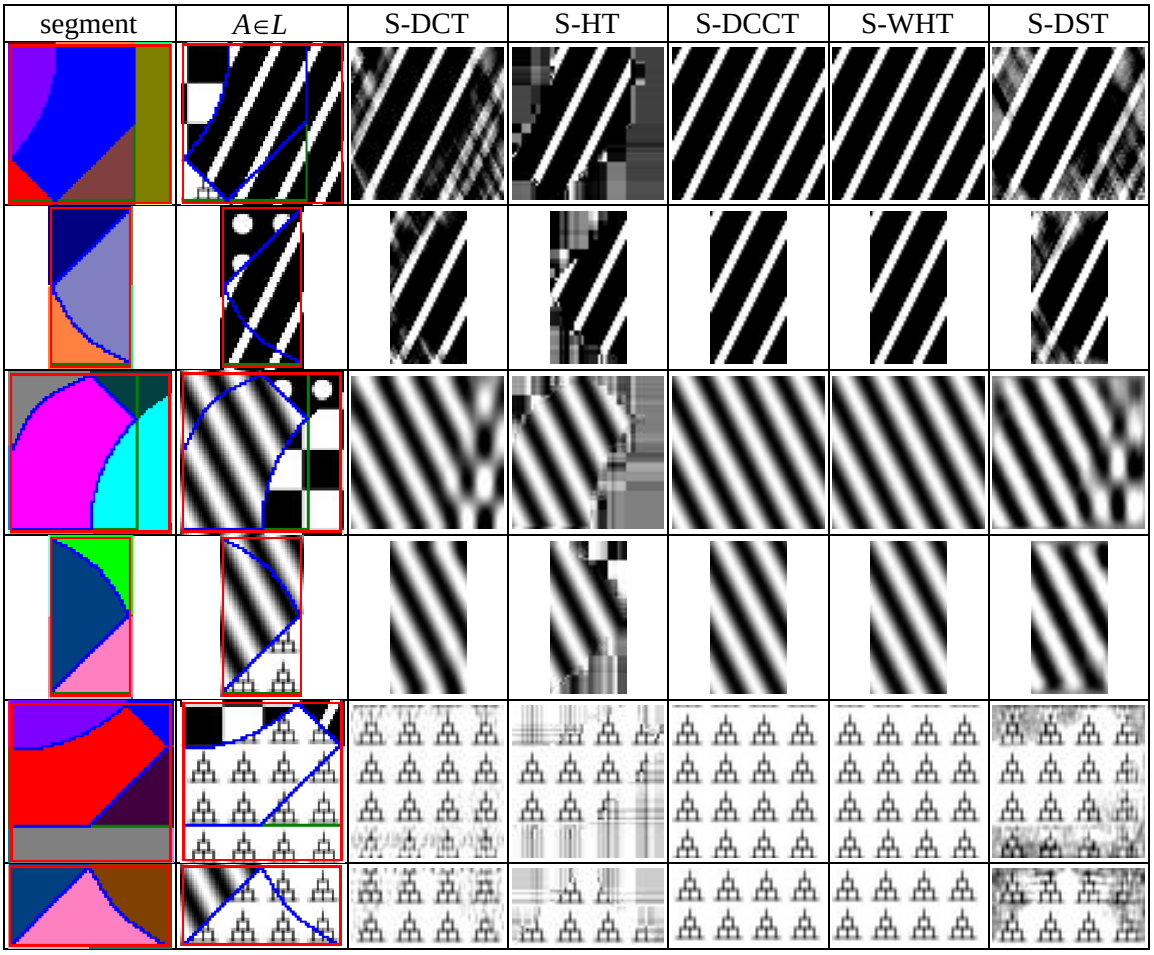


Figure 4.2: Extrapolation of six selected segments from Fig. 4.1 using different basis sets: successive Discrete cosine transform (S-DCT), successive Haar transform (S-HT), successive Separable 2D Hartley Transform (S-DCCT), successive discrete Walsh-Hadamard transform (S-WHT) and successive Discrete sine transform (S-DST)

$$\mathbf{x} = (x_0 x_1 x_2 ... x_{M-1})^T \tag{4.1}$$

then:

$$\mathbf{X} = \frac{1}{M}\mathbf{H}\mathbf{x} = (X_0 X_1 X_2 ... X_{M-1})^T \tag{4.2}$$

Let

$$\mathbf{y} = \begin{pmatrix} \mathbf{x} \\ \mathbf{x} \end{pmatrix} \tag{4.3}$$

then:

$$\mathbf{Y} = \frac{1}{2M}\begin{pmatrix} \mathbf{H} & \mathbf{H} \\ \mathbf{H} & -\mathbf{H} \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{X} \\ 0 \end{pmatrix} \tag{4.4}$$

4.2.3.2 Discrete Walsh-Hadamard Transform - Sequential Order of Basis Functions

By simple reordering of basis functions we get:

$$\mathbf{Y} = (X_0 00 X_1 X_2 00 X_3 X_4 ... 00 X_{M-1})^T \tag{4.5}$$

4.2.3.3 Discrete Hartley Transform - Natural Order of Basis Functions

Discrete Hartley transform of 1D sequence is defined:

$$X_k = \frac{1}{M}\sum_{n=0}^{M-1} x_n \left[cos\left(\frac{2\pi kn}{M}\right) + sin\left(\frac{2\pi kn}{M}\right)\right];\ k = 0, 1, ..., M-1 \tag{4.6}$$

By using *cas* function defined as:

$$cas\alpha = cos\alpha + sin\alpha \tag{4.7}$$

we get:

$$X_k = \frac{1}{M}\sum_{n=0}^{M-1} x_n \left[cas\left(\frac{2\pi kn}{M}\right)\right];\ k = 0,\ 1,\ ...,\ M-1 \tag{4.8}$$

Now we again define **y** as in Equation 4.3 and we get its Discrete Hartley transform

$$Y_k = \frac{1}{2M}\sum_{n=0}^{2M-1} y_n \left[cas\left(\frac{2\pi kn}{2M}\right)\right];\ k = 0,\ 1,\ ...,\ 2M-1 \tag{4.9}$$

We split the spectrum to even and odd components. Even components are

$$Y_{2l} = \frac{1}{2M}\sum_{n=0}^{2M-1} y_n \left[cas\left(\frac{2\pi 2ln}{2M}\right)\right];\ l = 0,\ 1,\ ...,\ M-1 \tag{4.10}$$

Then we perform the following sequence of modifications

$$Y_{2l} = \frac{1}{2M}\left[\sum_{n=0}^{M-1} x_n cas\left(\frac{2\pi ln}{M}\right) + \sum_{n=M}^{2M-1} x_{n-M} cas\left(\frac{2\pi ln}{M}\right)\right] \tag{4.11}$$

$$Y_{2l} = \frac{1}{2M}\left[MX_l + \sum_{n=0}^{M-1} x_n cas\left(\frac{2\pi l(n+M)}{M}\right)\right] \tag{4.12}$$

$$Y_{2l} = \frac{1}{2M}\left[MX_l + \sum_{n=0}^{M-1} x_n cas\left(\frac{2\pi ln}{M} + 2\pi l\right)\right] \tag{4.13}$$

Then

$$Y_{2l} = \frac{1}{2M}(MX_l + MX_l) = X_l \tag{4.14}$$

Odd components are

$$Y_{(2l+1)} = \frac{1}{2M}\sum_{n=0}^{2M-1} y_n \left[cas\left(\frac{2\pi(2l+1)n}{2M}\right)\right];\ l = 0,\ 1,\ ...,\ M-1 \tag{4.15}$$

We make the following sequence of modifications

$$Y_{(2l+1)} = \frac{1}{2M}\left[\sum_{n=0}^{M-1} x_n cas\left(\frac{\pi(2l+1)n}{M}\right) + \sum_{n=M}^{2M-1} x_{n-M} cas\left(\frac{\pi(2l+1)n}{M}\right)\right] \tag{4.16}$$

$$Y_{(2l+1)} = \frac{1}{2M}\left[\sum_{n=0}^{M-1} x_n cas\left(\frac{\pi(2l+1)n}{M}\right) + \sum_{n=0}^{M-1} x_n cas\left(\frac{\pi(2l+1)(n+M)}{M}\right)\right] \tag{4.17}$$

$$cas\left(\frac{\pi(2l+1)(n+M)}{M}\right) = -cas\left(\frac{\pi(2l+1)n}{M}\right) \tag{4.18}$$

$$Y_{(2l+1)} = \frac{1}{2M}\left[\sum_{n=0}^{M-1} x_n cas\left(\frac{\pi(2l+1)n}{M}\right) - \sum_{n=0}^{M-1} x_n cas\left(\frac{\pi(2l+1)n}{M}\right)\right];\ l = 0,\ 1,\ ...,\ M-1 \tag{4.19}$$

Then

$$Y_{(2l+1)} = \frac{1}{2M} \cdot 0 = 0 \tag{4.20}$$

4.2.3.4 Discrete Hartley Transform - Sequential Order of Basis Functions

Simple reordering of basis functions yields:

$$\mathbf{Y} = (X_0 0 0 X_1 X_2 0 0 X_3 X_4 ... 0 0 X_{M-1})^T \tag{4.21}$$

Discrete cosine transform II (DCT), discrete sinus transform (DST), discrete Haar transform (HT) and many other prominent transforms can not be periodified in the described manner. It is equally mathematically incorrect to unify the spectral dimension by averaging magnitudes of the DCT spectrum, as done in method proposed by Pun et al. [45]. If the need to reduce the feature vector dimensions arises, it is necessary to respect the properties described in this chapter. The same applies for increasing the dimension.

4.2.4 Experiments on the Luminance Components of Textures

For purposes of the experiments we created images by combining textures from Brodatz [14] and Simoncelli [43] databases. A total of 30 images with two, three, four and five grayscale

Table 4.1: Results for periodic textures.

number of classes			2	3	4	5
average accuracy [%]	DCT 8x8 sqeuclidean		**71,35**	**66,76**	**61,53**	**60,27**
	DCT 16x16 sqeuclidean		91,98	89,28	86,58	81,17
	S-DCCT	sqeuclidean	**99,91**	**99,73**	97,39	97,03
		correlation	**99,91**	99,55	99,19	**99,01**
		cosine	**99,91**	99,46	96,39	96,13
		city block	99,82	**99,73**	**99,64**	**99,01**
	S-WHT	sqeuclidean	99,73	99,46	98,11	94,59
		correlation	**99,91**	**99,73**	98,38	97,84
		cosine	**99,91**	99,37	97,93	95,41
		city block	99,64	99,46	97,21	94,41

Table 4.2: Results for quasiperiodic textures.

number of classes			2	3	4	5
average accuracy [%]	DCT 8x8 sqeuclidean		**73,51**	71,17	62,25	61,26
	DCT 16x16 sqeuclidean		87,48	85,86	82,07	71,26
	S-DCCT	sqeuclidean	98,11	96,49	91,26	89,91
		correlation	99,01	**98,74**	**95,23**	**93,51**
		cosine	**99,73**	98,65	93,69	91,35
		city block	89,19	**64,14**	**57,75**	**57.48**
	S-WHT	sqeuclidean	98,02	96,22	91,08	87,75
		correlation	96,22	95,14	94,05	89,28
		cosine	96,13	95,32	94,68	88,92
		city block	89,55	66,58	62,61	56.13

textures were created. The feature vector then contains only the magnitudes of the AC spectral coefficients vectorized by the zig-zag method. The spectral dimensions were further unified to 32x32. The DC coefficient was not used in order to demonstrate how important the AC coefficients are for the spectral description of the textures. The feature vector created in this manner is independent from the mean luminance of the analyzed area. We then compared the results with other techniques that use inscribed squares and magnitudes of the 2D DCT II spectral coefficients for classification. Synthetically over-segmented images use the same mask with 37 different regions (Fig. 4.1). The k-means algorithm was used in the classification process with different metrics. The results of each classification's success are located in Table 4.1 for periodic and in Table 4.2 for quasi-periodic structures. Some of the classification results can be found in Fig. 4.3.

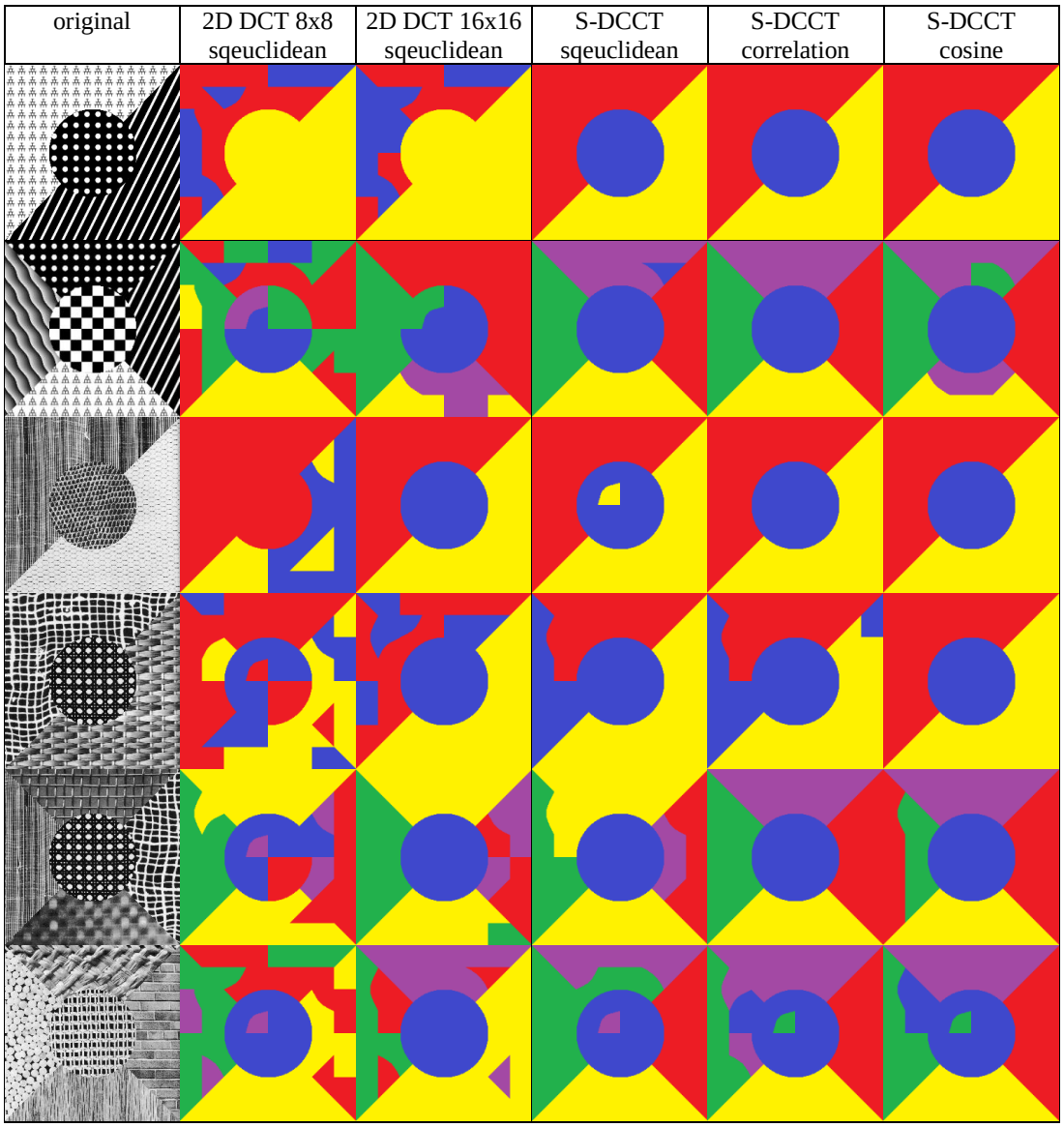


Figure 4.3: Classification results. For input original textured images (left column - first two rows only periodic textures, the rest also quasiperiodic textures) we used different methods to classify segments (cutout by the over-segmentation mask from Fig. 4.1) into specified number of groups. First two methods 2D DCT 8x8 and 2D DCT 16x16 use only inscribed squares with dimensions 8x8 and 16x16, respectively, into segments to find feature vectors. These two methods does not provide good results (see Tables 4.1 and 4.2 for complete average results) because they extract only part of the segment texture to get the feature vector and that yields to inaccurate results. The last three columns show results for successive DCCT transform (S-DCCT) method that is able to extract texture features from whole arbitrarily shaped segment and therefore it does not loose information compared to methods that inscribe squares into segments. To classify feature vectors obtained by S-DCCT transform of segments we used k-means clustering with different metrics: Square Euclidean (abbreviated as sqeuclidean), Correlation, Cosine and City Block metric. See Tables 4.1 and 4.2 for complete average results of S-DCCT and successive discrete Walsh-Hadamard transform (S-WHT)).

4.2.5 Amendment of Color Information into the Feature Vectors

In this section we propose a method for construction of an optimal feature set for classification algorithms that are used to merge superpixels into segments. Common color image segmentation techniques merge superpixels using the clustering algorithms working with an input in a form of features typically obtained only on the basis of the superpixel mean value (average color) in one of the color spaces. Arbitrarily shaped superpixels are usually problematic when considering the texture. It is often difficult to find an inscribed rectangle capable of representative texture shape properties capture. We solved this problem by the process proposed in the previous sections that describe a successive transformation to the space of chosen orthogonal basis. The feature vector obtained this way contains primarily information about texture. In order to ensure higher reliability of the classification process it is important to also consider the color information of superpixels in the form of three additional features.

We used the color space YUV, specifically the *AC* spectrum coefficients of the luminance (Y), mean luminance value (*DC* coefficient) and two additional values in the form of mean chrominance components (U, V) included with equal weight. *DC* coefficients containing color information remain in the feature vector even if its dimension is reduced by removing the higher frequency *AC* components scanned in a zig-zag fashion:

$$FV = (\mathrm{Y}_{DC}, \mathrm{U}_{DC}, \mathrm{V}_{DC}, AC_1, AC_2, AC_3, ..., AC_{max}) \tag{4.22}$$

The experimental results were obtained by applying the proposed method to images oversegmented by SLIC algorithm proposed by Achanta et al. [5].

The main part of the experiments we performed on the bird cardinal image from the BSD500 [6] database (see Fig. 4.4 (a)) and the results can be seen in Fig. 4.5. The image was oversegmented into 381 superpixels and the proposed algorithm was tested on a unified spectral block size 16x16 obtained by the proposed procedure. An image whose segmentation into superpixels corresponds exclusively to its color information was chosen purposefully. Superpixel merging requires also a classification approach based on the texture. The bird's feathers can in this context be considered as a quasiperiodic texture. The results clearly prove that while the color based classification (only *DC* coefficient) prefers the object as a whole and in case of some metrics also blends it to the background, the classification utilizing also spectral *AC* luminance information and an identical number of target classes produces a more realistic representation of the object's structure and even separates it better from the background. To ensure higher objectivity of the proposed method we did not use the condition for merging

(a) (b) (c)

Figure 4.4: Process of image segmentation: (a) Original image of bird cardinal from the BSD500 [6] database; (b) Over-segmented image using fast SLIC algorithm [5]; (c) Human classification of SLIC superpixels into 7 classes.

only neighboring superpixels. The complete vector consisted of 256 spectral coefficient from the luminance (Y) component and 2 *DC* coefficients of the chrominance components. As the presented results demonstrate, it is unnecessary to use all available *AC* spectral coefficients of the luminance to obtain best results. The most visually appealing results were achieved using only 10 spectral coefficients of the Y components and 2 *DC* coefficients of the U and V components using the k-means with correlation metric.

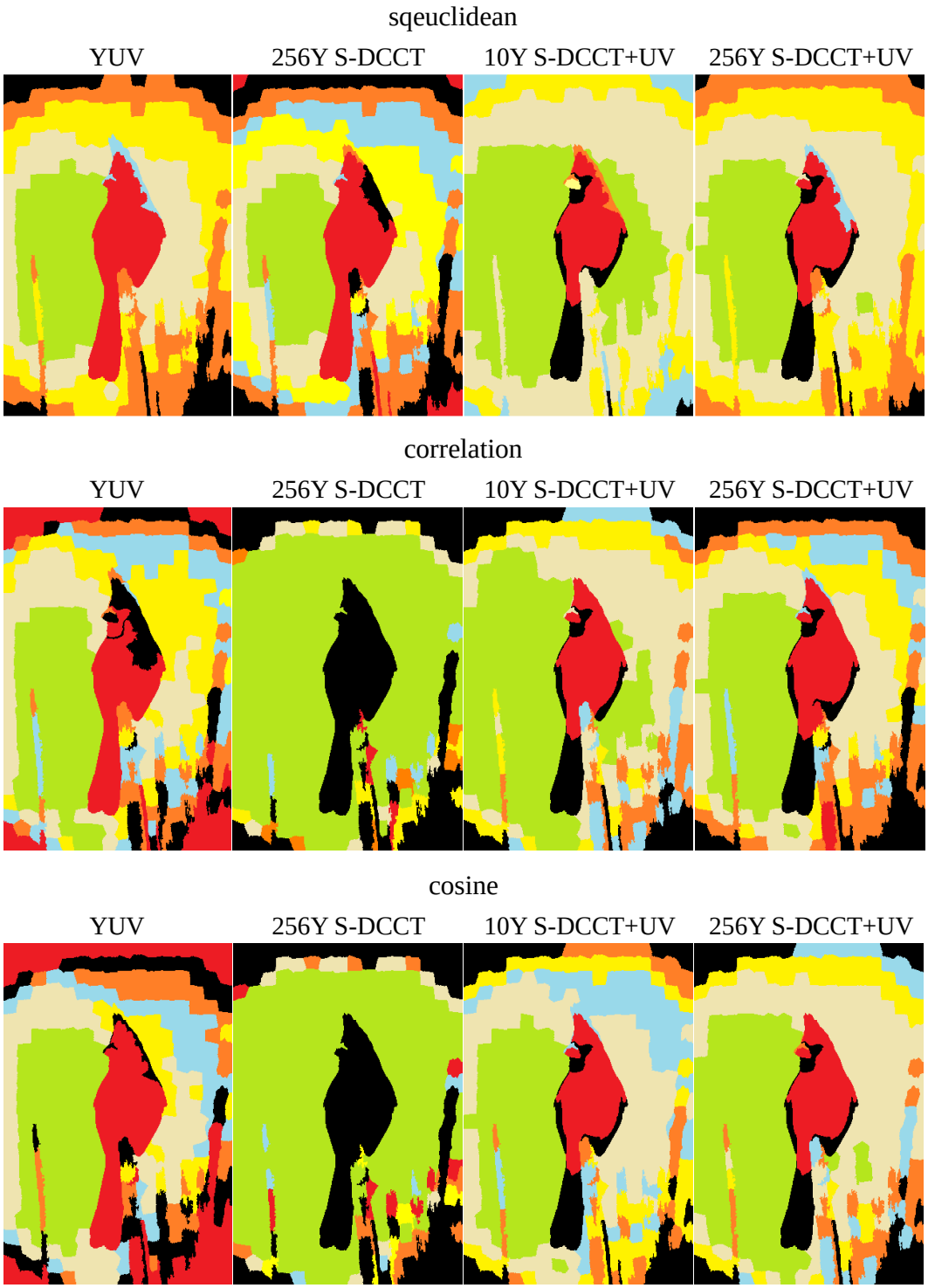
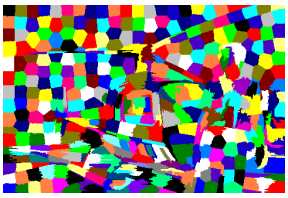


Figure 4.5: Classification of superpixels from over-segmented bird cardinal image from Fig. 4.4 using various feature vectors and metrics. All results were obtained by k-means clustering of image superpixels into 7 classes. Results in the left column were produced by utilizing only three mean color values of every superpixel, i.e. feature vector of every superpixel contained only one luminance (Y) and two chrominance (U and V) values. In second column, in contrary, color values were not used and the feature vector contained only spectral coefficients from S-DCCT matrix. The best results were achieved when both YUV color and spectral coefficients were included in feature vector for k-means classification. Additionally, the usage of only 10 most important spectral coefficients instead of all 256 provides similar or even better results.

We performed several tests of the proposed method with different images. Here we show results from testing of images with helicopter and giraffe (Fig. 4.6 and 4.7). From the results we can say that by using spectral coefficients the classification reaches better quality.

(a) Original image with helicopter



(b) Over-segmentation by SLIC algorithm [5]

(c) YUV

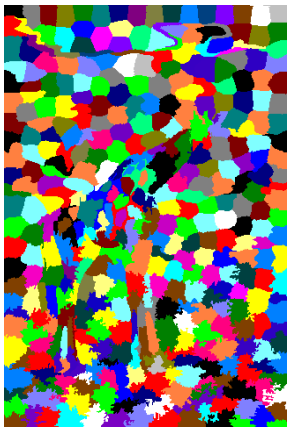
(d) 256Y S-DCCT

(e) 10Y S-DCCT + UV

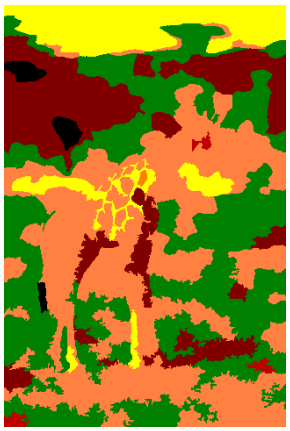
(f) 256Y S-DCCT + UV

Figure 4.6: Classification of superpixels from over-segmented image with helicopter (a, b) using various feature vectors. All results were obtained by k-means clustering of superpixels into 5 classes with City Block metric and the spectral dimensions were unified to 16x16. By using all 256 spectral coefficients (d, f) the helicopter was clearly separated from the background regardless the use of the chrominance (U and V) values. Without spectral coefficients (c) or with just 10 spectral coefficients (e) it was not possible.

(a) Original image with giraffe



(b) Over-segmentation by SLIC algorithm [5]



(c) YUV

(d) 256Y S-DCCT + UV

Figure 4.7: Classification of superpixels from over-segmented image with giraffe (a, b) using feature vectors with and without spectral coefficients. All results were obtained by k-means clustering of superpixels into 5 classes with Square Euclidean metric and the spectral dimensions were unified to 16x16. In case of classification without spectral coefficients (c) the giraffe is mixed with background. In case of classification with spectral coefficients (d) the result is much better.

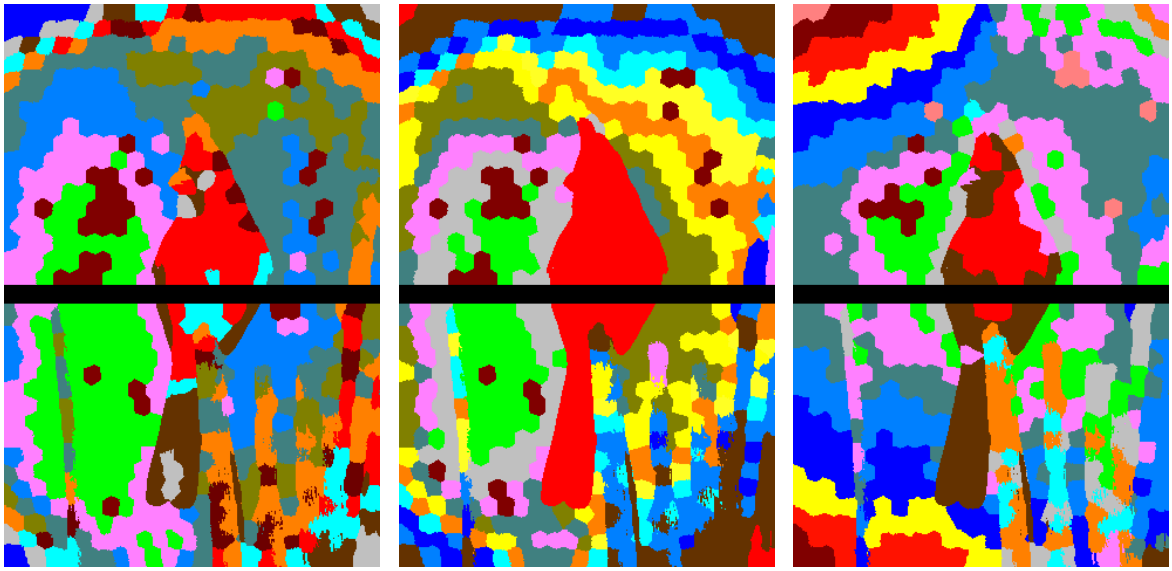
(a) (b)

Figure 4.8: Original image of bird cardinal with corrupted stripe in the middle marked by black color (a) and its over-segmentation by SLIC algorithm [5] (b)

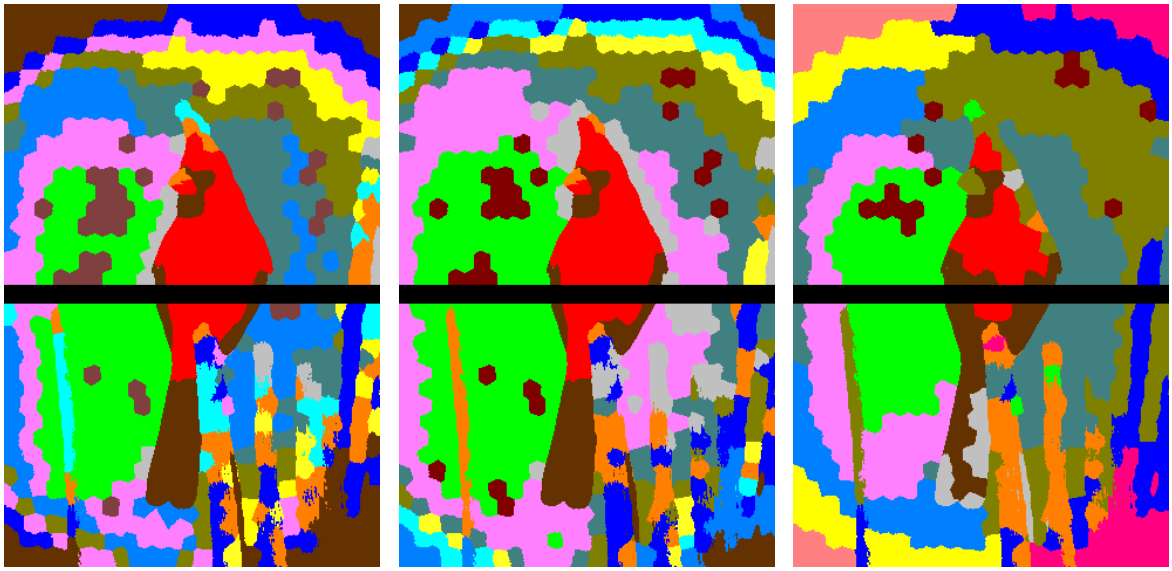
4.3 Evaluation of the Proposed Method in Corrupted Images and Comparison of Using Various Color Spaces

We made several experiments with classification of superpixels in corrupted images by using different color spaces and we compare the results when the spectral information is and is not used. Classification results on image with cardinal bird (Fig. 4.8 and 4.9) and image with Lenna (Fig. 4.10) show that mainly in case of L*a*b* color space the adding of spectral information is a big improvement. In our error concealment method (described in Chapter 3.2) we used only L*a*b* color values to decide whether to merge superpixels and these results lead us to use also spectral information. However, extracting of texture features can be time consuming, but we could still consider to use it at least for the image error concealment where the speed of the algorithm is not so crucial as in the video error concealment.

Tests in YUV and HSL color spaces (Fig. 4.9 and 4.11) also achieve better quality with using spectral information.



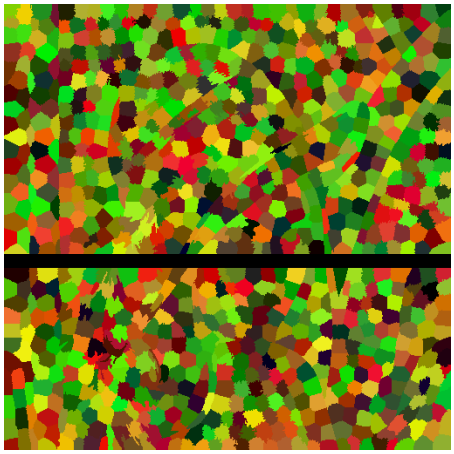
(a) YUV (b) L*a*b* (c) HSL



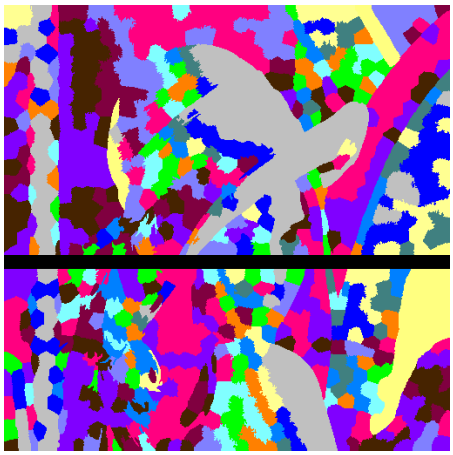
(d) 1024Y S-DCCT + UV (e) 1024L* S-DCCT + a*b* (f) 1024L S-DCCT + HS

Figure 4.9: Classification of superpixels from the over-segmented image of the bird cardinal from Fig. 4.8 using three different color spaces and the feature vectors with or without spectral information. All results were obtained by k-means clustering of superpixels into 15 classes with correlation metric and the spectral dimensions were unified to 32x32. Results in the first row (a-c) were produced by utilizing only three mean color values of every superpixel, i.e. feature vector of every superpixel contained only one luminance (Y or L(*)) and two chrominance (U and V or a* and b* or H and S) values. In second row the feature vector contained also spectral coefficients from S-DCCT matrix. The best results were achieved when both color and spectral coefficients were included in the feature vectors. Added spectral information to the YUV values (d) helped to properly segment right wing of the bird. In case of the classification based only on the L*a*b* values (b) the wings of the bird were not segmented at all. By adding the spectral information into the feature vector (e) the wings were nicely segmented. In HSL color space there was not such success (c, f), but at least the spectral information helped to separate the bird and the background.

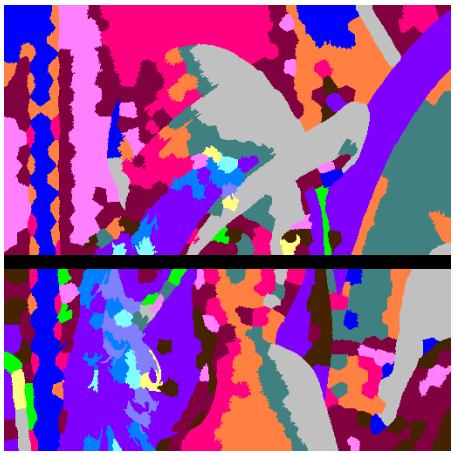
(a) Original image of Lenna with marked corrupted area



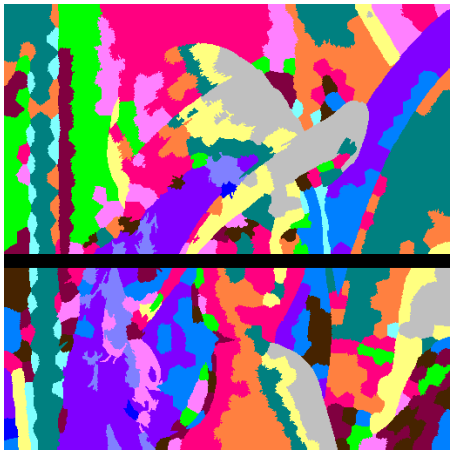
(b) Over-segmentation of (a) by SLIC algorithm [5]



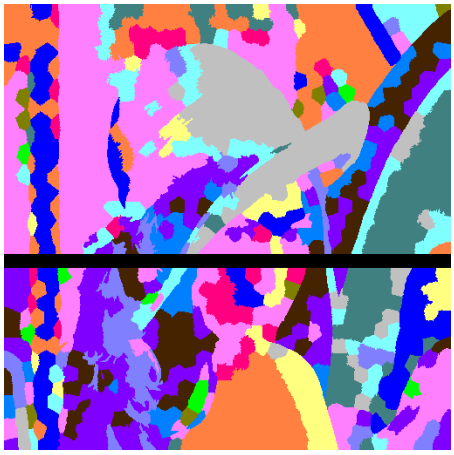
(c) L*a*b*



(d) 1024L* S-DCCT + a*b*



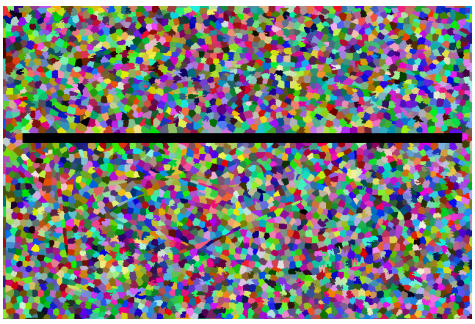
(e) 1024Y S-DCCT + UV



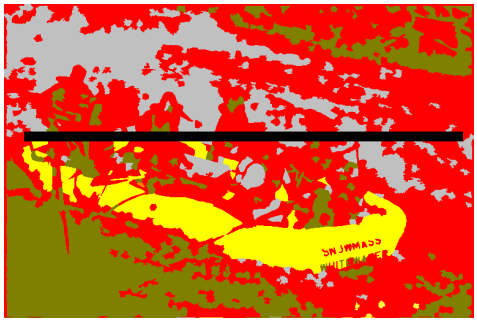
(f) 1024L S-DCCT + HS

Figure 4.10: Results of the classification of superpixels from the over-segmented image of Lenna (a, b). All results were obtained by k-means clustering of superpixels into 15 classes with correlation metric and the spectral dimensions were unified to 32x32. Classification result without spectral information in L*a*b* color space (c) is a big mess while the classification with spectral information in the same color space (d) provides a very good result. In this result the mouth of Lenna is segmented the best from all results.

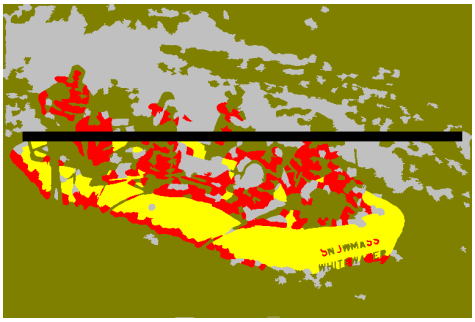
(a) Original image with raft and marked corrupted area



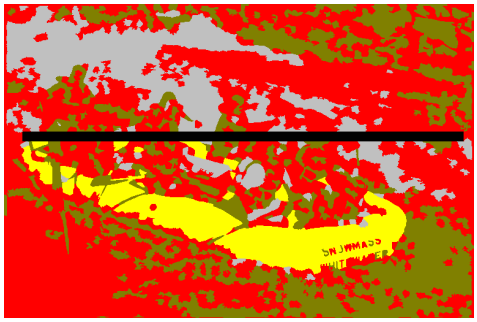
(b) Over-segmentation of (a) by SLIC algorithm [5]



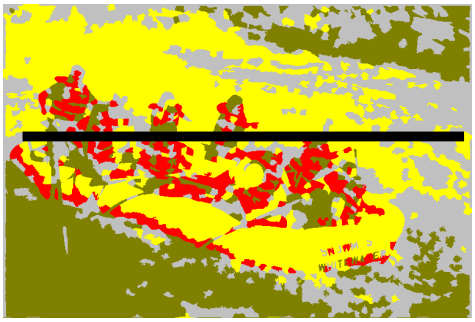
(c) YUV



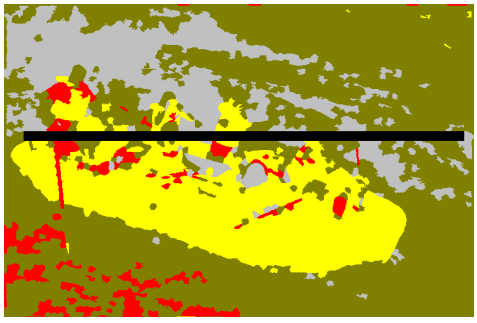
(d) 256Y S-DCCT + UV



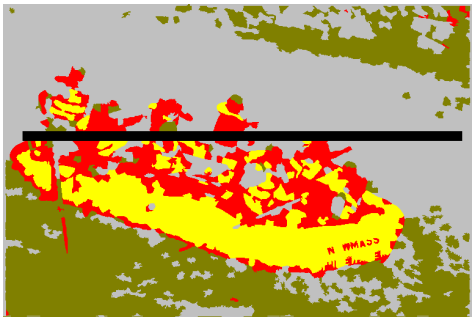
(e) L*a*b*



(f) 256L* S-DCCT + a*b*



(g) HSL



(h) 256L S-DCCT + HS

Figure 4.11: Classification of superpixels from the over-segmented image with raft (a, b) using three different color spaces and the feature vectors with or without spectral information. All results were obtained by k-means clustering of superpixels into 5 classes with Square Euclidean metric and the spectral dimensions were unified to 16x16. Results in the left row (c, e, g) were produced by utilizing only three mean color values of every superpixel. In the right column (d, f, h) the feature vector contained also spectral coefficients from S-DCCT matrix. In all three cases the added spectral information helped to separate the raft and people from surrounding water.

Chapter 5

Conclusions and Future Work

We proposed a new efficient and highly scalable method for image error concealment that outperforms the fast methods in quality and produces results with comparable quality as the sophisticated methods with run times of several orders of magnitude shorter.

The proposed algorithm for image error concealment utilizes a new approach to selective texture extrapolation into the corrupted area based on a completed segmentation of the image. Our key contribution is the extension of the shape error concealment techniques for single objects to segmented images with each corrupted segment being a small object to conceal. Connectivity of neighboring segments and of segment pieces cut by the lost area into several pieces was one of the main challenges that we solved.

A precise segmentation of textures is very important, but not always possible. Discrete segmentation fails for example for unsharp edges or gradients and therefore, we proposed to use the fuzzy segmentation instead. With the fuzzy segmentation the soft edges and gradients are maintained in the resulting concealed image.

Experimental results clearly demonstrate that by utilizing the segmentation we achieved a big qualitative improvement of the error concealment capability of our texture aware method over the frequency selective extrapolation [32, 54, 34] that extrapolates the texture from a square window around the corrupted area.

Fast patch-based methods [15, 16], that copy patches from the surroundings of the lost area very often create many artifacts. Our method avoids such artifacts thanks to the segmentation step.

The sophisticated patch [17] and group-based [84] methods are in general able to provide better results (although we found some exceptions), but their run-times are not feasible in

real time applications such as video streaming. Our method is much more faster, but still it should be speeded up to be suitable not just for error concealment of images and offline videos but also for concealment of streamed videos and animations.

There are some suggestions how to improve the performance of our method and to achieve interactive frame rates. The most time consuming part of our algorithm is the texture approximation by a successive orthogonal transform. The first improvement would be to implement it in parallel, because currently we run the transform for each of the three dimensions of the color space sequentially.

In our experiments we used successive Discrete cosine transform (S-DCT) and successive separable 2D Hartley Transform with sequentially ordered basis functions (S-DCCT) that are implemented in the SAPC program [70]. In this implementation the S-DCT has twice as long computational time compared to S-DCCT, because it computes the S-DCT transform through the fast Fourier transform algorithm of roughly twice the length. By substituting the algorithm with a faster one it would be possible to reach the run times of S-DCCT or even faster.

The third suggestion how to speed up our algorithm is to split large segments into smaller parts before approximating their textures, because texture approximation of smaller areas performed in parallel takes shorter. Splitting the segments should not be discrete but it should allow overlaps to avoid border artifacts.

Except for the speed up improvements there are also improvements possible in usability and quality. The input of our method is not only the corrupted image, but there are also some input parameters that should be defined by the user. Further work could also include a detailed testing to find the best possible input parameters in order to make a robust default preset. A better solution could be to define and implement automatic adaptation of the input parameters according to the image properties. That would be a nice usability enhancement.

For the quality enhancement we have three suggestions:

1. As it was already mentioned the approximation and extrapolation of the texture is performed separately for each channel of the color space. Firstly we tried to apply the transformation to RGB color channels, but the results were not satisfactory. After we switched to the YCbCr color space the achieved results improved. It would be interesting to test also other color spaces, e.g. CIE $L^*a^*b^*$ or YUV.

2. In case of the corrupted video frames it would be possible to use also information from previous uncorrupted or already concealed frames in order to achieve better quality results. It could help mainly for frames with larger corrupted parts. For the segmentation

it would be possible to utilize supervoxels instead of superpixels. The merging process of the supervoxels would be very similar to the merging of the superpixels. But the following step of the shape error concealment is not so easy then. In the 2D case we first find the segment boundaries and then we approximate them by curves in order to find tangent vectors in border points next to the lost area. With a tangent in each of the two border endpoints we can construct the concealment curve to smoothly join the endpoints. In the 3D case we need to find borders of segments as well, but we need to approximate each segment border by a patch going across the uncorrupted (previous) frames and not just by a curve for each frame separately. It is necessary in order to predict the change of the segment shape in corrupted (current) frame. The prediction of the shape change can be done by the extrapolation of the patch in the time domain into the current frame. The extrapolation result strongly depends on the quality of the matching of the segment shapes from each frame. The main difficulty is high elasticity of the shape changes in time. During our research we tried to develop a shape registration algorithm and we also tried to use the bUnwarpJ [7] library for the shape registration, but there is still a place for improvements. An important aspect of this concept is that in the last step, being the texture approximation and extrapolation, we can use some of the orthogonal transforms in a similar way as in 2D case. Here we would just apply it on 3D texture.

3. As the last quality improvement we suggest to merge superpixels into the segments based not just on their color, but also based on their textures. For that we proposed a novel spectral method for constructing the feature vector of superpixels that considers their inner texture and color as well. Conclusions for the proposed method are listed below.

In the second part of the thesis we introduced a new spectral method to construct the feature vector of arbitrarily shaped areas by considering both their inner texture and color. Our method is based on successive selective approximation from a set of orthogonal basis functions. Not every set of basis functions is suitable for realistic texture extrapolation. Therefore, we made several experiments to chose appropriate orthogonal basis function sets. Because the over-segmented areas do not have the same dimensions, it is necessary to first unify the spectral dimensions before the classification process. This leads to the problem of determining which basis sets fulfill the Fourier signal periodization conditions for inserting zeros between spectral coefficients. The requirement of this condition is based on the assumption that in a sufficiently small area its texture becomes periodic or quasiperiodic. We proved that such a property is exhibited not only by the Discrete Fourier Transform (DFT), but also by the separable Hartley transform and also the fastest transform used the in technical practice based on the Hadamard basis.

However, proper basis set selection and spectral unification alone do not solve all issues associated with the problem of proper feature extraction based on texture. We also proposed to extend the feature set commonly used for color images by typical information about the color. We examined the proposed method's efficiency in three color spaces: YUV, L*a*b* and HSL. By adding spectral information into the feature vector we always achieved better results and in the case of the CIE $L^*a^*b^*$ color space the difference was the most visible. Our experimental results confirm that the proposed method can be successfully used to give solutions for multiple problems such as texture classification and image segment merging.

Chapter 6

Projects and publications

Publications

AFC Publikované príspevky na zahraničných vedeckých konferenciách (Papers published in proceedings of international scientific conferences)

- Martin Ilčík (40%), Ivana Ilčíková (30%), Andrej Ferko (20%), Michael Wimmer (10%). 20 Years of the Central European Seminar on Computer Graphics. Accepted to the *Eurographics 2016: The 37th Annual Conference of the European Association for Computer Graphics, May 9 to May 13, Lisbon, Portugal*.
- Ivana Uhlíková (40%), Wanda Benešová (20%), Jaroslav Polec (20%), Tibor Csóka (20%). Texture Aware Image Error Concealment. In *Proceedings of EUROCON 2015: International Conference on Computer as a Tool, September 2015, Salamanca, Spain*, p. 88–93. Piscataway. ISBN 978-1-4799-8569-2.

AFD Publikované príspevky na domácich vedeckých konferenciách (Papers published in proceedings of home scientific conferences)

- Ivana Ilčíková (70%), Wanda Benešová (10%), Jaroslav Polec (10%), Tibor Csóka (10%). Texture Aware Video and Image Error Concealment. Accepted to the *ELITECH 2016: 18th Conference of PhD. Students, June 8, Bratislava*.
- Tibor Csóka (80%), Jaroslav Polec (10%), Ivana Ilčíková (10%). Physical and Data Link Layer Characteristics Capture in the Low Rate Wireless Sensor Network. Accepted to the *ELITECH 2016: 18th Conference of PhD. Students, June 8, Bratislava*.

- Ivana Ilčíková (70%), Wanda Benešová (10%), Jaroslav Polec (10%), Tibor Csóka (10%). Texture Aware Image Error Concealment with Fuzzy Segmentation. Accepted to the *IWSSIP 2016: 23rd International Conference on Systems, Signals and Image Processing, IEEE, May 23 to May 25, Bratislava*.

- Tibor Csóka (70%), Jaroslav Polec (10%), Ivana Ilčíková (10%), Ján Doboš (10%). Binary error models for Wireless Sensor Networks. Accepted to the *IWSSIP 2016: 23rd International Conference on Systems, Signals and Image Processing, IEEE, May 23 to May 25, Bratislava*.

- Ivana Ilčíková (70%), Jaroslav Polec (10%), Wanda Benešová (10%), Tibor Csóka (10%). Fuzzy Segmentation for Texture Aware Image Error Concealment. In *APLIMAT: 15th Conference on Applied Mathematics: 1st Edition, Bratislava: Vydavateľstvo STU, 2016*. p. 541-556. ISBN 978-80-227-4531-4.

AFH Abstrakty príspevkov z domácich vedeckých konferencií (Abstracts published in proceedings of home scientific conferences)

- Ivana Ilčíková (70%), Jaroslav Polec (10%), Wanda Benešová (10%), Tibor Csóka (10%). Fuzzy Segmentation for Texture Aware Image Error Concealment. In *APLIMAT: 15th Conference on Applied Mathematics: 1st Edition, Bratislava: Vydavateľstvo STU, 2016*. p. 116-117. ISBN 978-80-227-4530-7.

- Ivana Uhlíková. Webový video prehrávač s podporou 3D. In *Proceedings of Student Scientific Conference of Faculty of Mathematics, Physics and Informatics of Comenius University in Bratislava, 2012*, p. 313. ISBN 978-80-8147-001-0.

AFK Postery zo zahraničných konferencií (Poster papers published in proceedings of international scientific conferences)

- David Běhal (34%), Jana Dadová (33%), and Ivana Uhlíková (33%). 3D Extension of WEB. In *WSCG 2011 - Poster Papers Proceedings, 19th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzeň*, p. 37-40. ISBN 978-80-86943-81-7.

GII Rôzne publikácie a dokumenty (Various publications)

- Ivana Ilčíková. Maskovanie obrazových chýb zachovávajúce textúry. *Rigorous thesis*. 2015.

- Ivana Uhlíková. Webový video prehrávač s podporou 3D. *Diploma thesis*. 2012.
- Ivana Uhlíková. Open Source aplikácie vo výučbe. *Final thesis of the complementary pedagogical study*. 2012.
- Ivana Uhlíková. 3D WEB, *Bachelor thesis*. 2010.

Submitted paper into the journal

ADC Vedecké práce v zahraničných karentovaných časopisoch (Scientific papers in Current Contents Connect® Journals)

- Jaroslav Polec (20%), Wanda Benešová (20%), Radoslav Vargic (20%), Ivana Ilčíková (20%), Tibor Csóka (20%). Texture Features Extraction Using an Orthogonal Transform of Arbitrarily Shaped Image Regions. Submitted to the *Special Section: Color in Texture and Material Recognition of SPIE Journal of Electronic Imaging, IS&T & SPIE, ISSN: 1017-9909, November/December 2016*.

Research projects and travel grants

- Participating in the project *Cross-Layer Optimization of Wireless Systems Throughput* (VEGA 1/0789/15).
- Cooperating on the project *Visual object class recognition in video sequences using a linkage of information derived by a semantic local segmentation of visual saliency* (VEGA 1/0625/14).
- Ernst Mach Grant, Action Austria-Slovakia - 3 months scholarship (September to November 2015) for research stay in the Institute of Computer Graphics and Algorithms of Vienna University of Technology sponsored by OeAD (Austrian Agency for International Cooperation in Education and Research) in cooperation with SAIA, n. o. (Slovak Academic Information Agency).
- Travel sponsorship from Literárny fond (Literary Fund) for the Eurographics conference in Lisabon. May 2016.

Awards

- Winner of Computer Graphics Section of Student Scientific Conference of Faculty of Mathematics, Physics and Informatics of Comenius University in Bratislava, 2012

Tutoring

- Adam Šabík: Interaktívna vizuálna analýza stromov odvodení pre CGA shape gramatiky, *Bachelor thesis*. 2015.
- Boris Ledecký: Technologická prezentácia WebGL vizualizáciou vybraných fyzikálnych javov, *Bachelor thesis*. 2014.

Bibliography

[1] ITU-R Recommendation BT.1683: 2004, Objective perceptual video quality measurement techniques for standard definition digital broadcast television in the presence of a full reference.

[2] ITU-T Recommendation J.144: 2004, Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference.

[3] Msu video quality measurement tool. `http://www.compression.ru/video/quality_measure/video_measurement_tool_en.html`, accessed April 2016.

[4] ANSI T1.801.03: 2003, American National Standard for Telecommunications – Digital transport of one-way video signals – Parameters for objective performance assessment, 2003.

[5] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[6] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour Detection and Hierarchical Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011.

[7] I. Arganda-Carreras. bUnwarpJ. 2015. http://imagej.net/BUnwarpJ, accessed April 2016.

[8] İ. Avcıbaş, B. Sankur, and K. Sayood. Statistical evaluation of image quality measures. *Journal of Electronic imaging*, 11:206, 2002.

[9] M. Barni. *Document and Image Compression*. Signal Processing and Communications. CRC Press, 2006. ISBN 9780849335563.

[10] G. Beliakov. Shape preserving approximation using least squares splines. *Approx. Theory and its Appl.*, 16(4):80–98, 2000.

[11] W. Benešová and M. Kottman. Fast Superpixel Segmentation Using Morphological Processing. In *Proc. of Int. Conf. on Machine Vision and Machine Learning (MVML), 2014*. Avestia Publishing, 2014.

[12] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image Inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH'00, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[13] R. Birkus. Accelerated gSLIC for Superpixel Generation used in Object Segmentation. In *Proc. of CESCG '15*, Apr. 2015.

[14] P. Brodatz. *Textures: a photographic album for artists and designers*. Dover pictorial archives. Dover Publications, 1966.

[15] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 13(9):1200–1212, Sept 2004.

[16] M. Daisy, D. Tschumperlé, and O. Lézoray. A Fast Spatial Patch Blending Algorithm for Artefact Reduction in Pattern-based Image Inpainting. In *SIGGRAPH Asia 2013 Technical Briefs*, SA '13, pages 8:1–8:4, New York, NY, USA, 2013. ACM.

[17] S. Darabi, E. Shechtman, C. Barnes, D. B Goldman, and P. Sen. Image Melding: Combining Inconsistent Images using Patch-based Synthesis. *ACM Trans. on Graphics*, 31(4):82:1–82:10, 2012.

[18] L. S. Davis. A Survey of Edge Detection Techniques. In *Computer Graphics and Image Processing*, volume 4, pages 248–270, 1975.

[19] R. Dosselmann and D. X. Yang. A Formal Assessment of the Structural Similarity Index. Technical report, Technical Report TR-CS 2008-2, University of Regina, 2008.

[20] F. Drucker and J. MacCormick. Fast superpixels for video analysis. In *Workshop on Motion and Video Computing*, pages 1–8, Dec. 2009.

[21] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038, 1999.

[22] P. Felzenszwalb and D. Huttenlocher. Efficient Graph-Based Image Segmentation. *Int. J. C. Vision*, 59(2):167–181, Sept. 2004.

[23] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *12th IEEE Int. Conf. on Comp. Vision*, pages 670–677, 2009.

[24] J. Gibson. *The perception of the visual world*. Houghton Mifflin, 1950.

[25] GIMP. http://www.gimp.org/, accessed April 2016.

[26] I. Hamouchene, S. Aouat, and H. Lacheheb. Texture segmentation and matching using LBP operator and GLCM matrix. In *Intelligent systems for science and information*, pages 389–407. Springer International Publishing, 2014.

[27] R. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610–621, November 1973.

[28] ISO. Information technology - coding of audio-visual objects - part 2: Visual. ISO/IEC 14496-2:2004. ISO, 2004.

[29] A. K. Jain. *Fundamentals of Digital Image Processing*. New Jersey, United States of America: Prentice Hall, 1989. ISBN 0-13-336165-9.

[30] H. Jiang, J. Wang, Z. Yuan, T. Liu, N. Zheng, and S. Li. Automatic salient object segmentation based on context and shape prior. *BMVC*, 6:7, 2011.

[31] A. Kaup and T. Aach. Coding of segmented images using shape-independent basis functions. *Image Processing, IEEE Transactions on*, 7(7):937–947, 1998.

[32] A. Kaup, K. Meisinger, and T. Aach. Frequency selective signal extrapolation with applications to error concealment in image communication. *AEUE - International Journal of Electronics and Communications*, 59:147–156, 2005.

[33] Kodak. Lossless true color image suite. http://r0k.us/graphics/kodak/, accessed April 2016.

[34] J. Koloda, J. Seiler, A. Kaup, V. Sánchez, and A. Peinado. Frequency Selective Extrapolation with Residual Filtering for Image Error Concealment. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 1976–1980, Florence, Italy, May 2014.

[35] J. Koloda, J. Østergaard, S. H. Jensen, V. Sánchez, and A. M. Peinado. Sequential error concealment for video/images by sparse linear prediction. *IEEE Transactions on Multimedia*, pages 957–969, June 2013.

[36] X. Li and J. Cai. Robust Transmission of JPEG2000 Encoded Images Over Packet Loss Channels. In *2007 IEEE International Conference on Multimedia and Expo*, pages 947–950, July 2007.

[37] A. Mansfield, M. Prasad, C. Rother, T. Sharp, P. Kohli, and L. V. Gool. Transforming Image Completion. In *British Machine Vision Conference (BMVC)*, August 2011.

[38] A. Monadjemi. *Towards Efficient Texture Classification and Abnormality Detection*. PhD thesis, Department of Computer Science, University of Bristol, 2004.

[39] T. Ojala, M. Pietikainen, and D. Harwood. A comparative study of texture measures with classification based on feature distribution. *Pattern Recognition*, 29(1):51–59, 1996.

[40] P. Polatsek and W. Benesova. Bottom up saliency model generation using superpixels. In *Proc. of the 31st Spring Conf. on Comp. Graphics, Smolenice*, 2015.

[41] J. Polec, J. P. J., and R. Vargic. New ordering sequences and bases Functions of Discrete Fourier and Discrete Hartley Transforms for Transforms Coders. *Journal on Communications*, XLV:75–76, 1994.

[42] J. Polec, T. Karlubíková, and A. Březina. Hybrid Orthogonal Approximation of Non-Square Areas. In *Comp. Vision and Graphics*, volume 32 of *Computat. Imaging and Vision*, pages 752–757. Springer, 2006.

[43] J. Portilla and E. Simoncelli. Representation and synthesis of visual texture. http://www.cns.nyu.edu/ eero/texture/, accessed April 2016.

[44] W. K. Pratt. *Digital Image Processing (2nd Edition)*. John Wiley & Sons, Inc., New York, NY, USA, 1991.

[45] C. M. Pun, N. Y. An, and M. Cheng. A Region-Based Image Segmentation by Watershed Partition and DCT Energy Compaction. In *Computer Graphics, Imaging and Visualization (CGIV), 2011 Eighth International Conference on*, pages 131–135, Aug 2011.

[46] C.-M. Pun and H.-M. Zhu. Textural Image Segmentation Using Discrete Cosine Transform. In *Proceedings of the 3rd International Conference on Communications and Information Technology*, CIT'09, pages 54–58, Stevens Point, Wisconsin, USA, 2009. World Scientific and Engineering Academy and Society (WSEAS).

[47] T. R. Reed and J. M. H. du Buf. A Review of Recent Texture Segmentation and Feature Extraction Techniques. *CVGIP: Image Underst.*, 57(3):359–372, May 1993.

[48] C. Y. Ren and I. Reid. gSLIC: a real-time implementation of SLIC superpixel segmentation. Technical report, Oxford University, 2011.

[49] X. Ren and J. Malik. Learning a classification model for segmentation. In *9th IEEE Int. Conf. on Comp. Vision*, volume 1, pages 10–17, Oct. 2003.

[50] I. E. Richardson. *The H.264 Advanced Video Compression Standard*. John Wiley & Sons, Ltd, 2010.

[51] I. E. G. Richardson. *H.264 and MPEG-4 Video Compression*. John Wiley & Sons, Ltd, 2003.

[52] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*. Academic Press, Inc., Orlando, FL, USA, 2nd edition, 1982.

[53] G. Schuster, X. Li, and A. Katsaggelos. Shape error concealment using Hermite splines. *IEEE Transactions on Image Processing*, 13(6):808–820, June 2004.

[54] J. Seiler and A. Kaup. Complex-Valued Frequency Selective Extrapolation for Fast Image and Video Signal Extrapolation. *IEEE Signal Processing Letters*, 17(11):949–952, Nov. 2010.

[55] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[56] S. Shirani, F. Kossentini, and R. Ward. An adaptive markov random field based error concealment method for video communication in error prone environment. *Proceedings of ICIP*, 6:3117–3120, 1999.

[57] P. K. Singh. Unsupervised segmentation of medical images using DCT coefficients. In *Proceedings of the Pan-Sydney area workshop on Visual information processing*, pages 75–81. Australian Computer Society, Inc., 2004.

[58] L. D. Soares. *Error Resilience for Object-based Video Coding*. PhD thesis, Universidade Técnica De Lisboa, 2004.

[59] L. D. Soares and F. Pereira. Spatial shape error concealment for object-based image and video coding. *IEEE Transactions on Image Processing*, 13(4):586–599, Apr. 2004.

[60] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision*. Springer US, 1993.

[61] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum. Image Completion with Structure Propagation. *ACM Trans. Graph.*, 24(3):861–868, 2005.

[62] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[63] Y. Takishima, M. Wada, and H. Murakami. Reversible variable length codes. *Communications, IEEE Transactions on*, 43(234):158–162, Feb 1995.

[64] D. Tschumperle and R. Deriche. Vector-valued image regularization with PDEs: A common framework for different applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(4):506–517, 2005.

[65] E. Tsiligianni, L. Kondi, and A. Katsaggelos. Shape Error Concealment Based on a Shape-Preserving Boundary Approximation. *IEEE Transactions on Image Processing*, 21(8):3573–3585, Aug. 2012.

[66] M. Tuceryan and A. K. Jain. Texture segmentation using Voronoi polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):211–216, Feb 1990.

[67] M. Tuceryan and A. K. Jain. Handbook of Pattern Recognition and Computer Vision (2nd Edition). chapter Texture Analysis, pages 207–248. World Scientific Publishing Co, Inc., 1998.

[68] I. Uhlíková, W. Benešová, J. Polec, and T. Csóka. Texture Aware Image Error Concealment. In *Proceedings of Eurocon '15*, pages 88–93. IEEE, Sept. 2015.

[69] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. VideoTrace: Rapid Interactive Scene Modelling from Video. In *ACM SIGGRAPH 2007 Papers*, New York, NY, 2007. ACM.

[70] R. Vanya. SAPC. 2003. http://sapc.host.sk/, accessed April 2016.

[71] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and Supervoxels in an Energy Optimization Framework. In *Comp. Vision - ECCV 2010*, volume 6315, pages 211–224. Springer, 2010.

[72] B. Wang, Z. Wang, Y. Liao, and X. Lin. HVS-based structural similarity for image quality assessment. In *2008 9th International Conference on Signal Processing*, pages 1194–1197, Oct 2008.

[73] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *IEEE Int. Conf. on Comp. Vision*, pages 1323–1330, Nov. 2011.

[74] Y. Wang, Q.-F. Zhu, and L. Shaw. Maximally smooth image recovery in transform coding. *IEEE Transactions on Communications*, 41(10):1544–1551, Oct. 1993.

[75] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[76] A. Watson. Toward a perceptual video quality metric. *Human Vision, Visual Processing, and Digital Display VIII*, pages 139–147, 1998.

[77] A. B. Watson and A. Poirson. Separable two-dimensional discrete Hartley transform. *JOSA A*, 3(12):2001–2004, 1986.

[78] S. Winkler. *Digital video quality vision model and metrics*. Chichester: John Wiley and Sons Ltd., 2005. ISBN 0-470-02404-6.

[79] J. Woods. *Multidimensional Signal, Image, and Video Processing and Coding*. Elsevier Science, Second edition, 2012.

[80] F. Xiao. DCT-based video quality evaluation. 2000. http://compression.ru/video/quality_measure/vqm.pdf, accessed April 2016.

[81] M. Yang and N. Bourbakis. High-bitrate multimedia information hiding for digital image/video under lossy compression. *Journal of Electronic Imaging*, 16(1):13008–13008, 2007.

[82] W. Yubing. Survey of Objective Video Quality Measurements. 2006. ftp://ftp.cs.wpi.edu/pub/techreports/pdf/06-02.pdf, accessed April 2016.

[83] G. Zhai, J. Cai, W. Lin, X. Yang, and W. Zhang. Image error-concealment via block-based bilateral filtering. *IEEE International Conference on Multimedia and Expo*, pages 621–624, June 2008.

[84] J. Zhang, D. Zhao, and W. Gao. Group-Based Sparse Representation for Image Restoration. *IEEE Transactions on Image Processing*, 23(8):3336–3351, Aug. 2014.

[85] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *IEEE Conf. on Comp. vision and pattern recognition*, pages 1838–1845, 2012.

Appendix

A Compact Disc (CD) containing the created application together with instructions how to use it is attached to the thesis. In the CD there is also the written part of the thesis in PDF format.