Towards a Synthetic World

DISSERTATION

zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

vorgelegt von

Christopher Georg Haccius

Saarbrücken, 2016

Tag des öffentlichen Kolloquiums:	21. Dezember 2016
Gutachter der Dissertation:	Prof. Dr.-Ing. Thorsten Herfet Prof. Dr. Adrian Hilton Prof. Dr.-Ing. Philipp Slusallek
Prüfungskommissionsvorsitzender:	Prof. Dr. Joachim Weickert
Akademischer Mitarbeiter:	Dr. Mathias Humbert
Dekan der Fachrichtung:	Prof. Dr. rer. nat. Frank-Olaf Schreyer

Kurzzusammenfassung

Visuelle Informationen müssen verschiedenen Anforderungen genügen. Sie sollen zum einen realistisch aussehen und zum anderen einfach modifizierbar sein. Diese beiden Anforderungen werden in verschiedenen Darstellungsformen visueller Informationen bedient: So sind computergenerierte Bilder leicht modifizierbar, aber in der Regel weniger realistisch als Kamera-Aufnahmen. Hingegen sind Kamera-Aufnahmen photorealistisch, aber kaum oder nur schwer modifizierbar. Diese Arbeit betrachtet das Zusammenspiel von generierten und aufgenommenen Daten aus verschiedenen Blickwinkeln, mit dem Ziel, die Vorteile beider Datenstrukturen zu kombinieren. Dafür stellen wir ein Datenformat vor, welches die Zusammenführung verschiedener Datenquellen erlaubt, und von uns entwickelte algorithmische Ansätze werden präsentiert, um in aufgenommenen Daten eine mit computergenerierten Daten vergleichbare Wissensbasis zu schaffen. Mit dieser Grundlage eröffnen wir für die Verarbeitung der Bilddaten vielfältige Möglichkeiten, von denen hier einige vorgestellt werden. Ein wichtiger Beirag ist unsere Analyse der subjektiv empfundenen Qualität von Fehlern, wie sie in Virtual Reality- und Augmented Reality-Szenarien vorkommen, und der von uns entwickelte Ansatz, diese Qualität algorithmisch zu bestimmen.

Short Abstract

Visual information needs to fulfill various requirements. On the one hand such information should be realistic, while on the other hand it should be easily modifiable. Both of these requirements are met by different kinds of visual data: Computer generated imagery can be modified, but is generally less realistic than captured content, whereas captured data is photo-realistic, but merely modifiable. This thesis approaches the interplay of generated and captured data in different ways, with the goal of joining the advantages of both kinds. Towards this goal we introduce a representation format which allows the merge of content coming from different data sources. We have advanced algorithmic approaches that enhance captured data by information comparable to generated data. On the basis of our novel representation various processing steps can be implemented, of which some are presented here. An important contribution is our analysis of subjectively perceived quality of augmented and virtual reality scenarios, and an approach to determine this quality algorithmically.

Zusammenfassung

Visuelle Darstellungen sind seit jeher durch zwei verschiedene Anforderungen motiviert. Auf der einen Seite versucht man, Beobachtungen der realen Welt darzustellen und für die Nachwelt zu erhalten. Auf der anderen Seite nutzen Künstler visuelle Darstellungen, um sich kreativ auszudrücken. Beide Zielsetzungen für visuelle Inhalte finden sich auch in der digitalen Welt. Mit digitalen Bild- und Videokameras können direkte Abbildungen der realen Welt festgehalten werden. Gleichzeitig schaffen Künstler mit Computer-Generated Imagery, durch 3D Computergrafik synthetisierte Bilder, atemberaubende, aber wenig realistische Welten. In den letzten Jahren haben technische und algorithmische Entwicklungen diese beiden sehr verschiedenen Arten visueller Erzeugnisse näher zusammen gebracht. Mit Computational Photography begann man, Ideen der Computergrafik auch auf reale Aufnahmen anzuwenden.

Hochentwickelte Rendering-Methoden erzeugen, basierend auf synthetischen Daten, Bilder, die von realen Szenen kaum zu unterscheiden sind. Trotz dieser Errungenschaften ist der Weg zur lückenlosen Integration von aufgenommenen und generierten Daten noch weit.

Ein primäres Ziel beim Zusammenführen beider Datenquellen ist die Flexibilität computer-generierter Inhalte mit dem Realismus von Aufnahmen zu verbinden, beziehungsweise die Flexibilität synthetischer Inhalte in aufgenommenen Szenen verwendbar zu machen. In der vorliegenden Arbeit werden verschiedene Methoden und Schritte eingeführt, diskutiert und verbessert, die bei der Synthetisierung realer Aufnahmen nötig sind. Neuartige Geräte zur Datenaufnahme erweitern visuelle Informationen nicht nur um eine weitere räumliche Dimension, sondern können auch diverse Materialeigenschaften wie bidirektionale Reflektanzverteilungsfunktionen, die für die Synthetisierung von Lichtverhältnissen unerlässlich sind, erfassen. Im Rahmen dieser Arbeit haben wir eine Darstellungsform entwickelt, die sowohl aufgenommene als auch synthetische Daten sowie eine Vielzahl an möglichen Zwischenstufen zulässt. Damit wird eine wichtige Grundlage für die lückenlose Zusammenführung beider Welten geschaffen. Algorithmische Ansätze, die diese neue Darstellung mit über die aufgenommenen Daten hinaus gehenden Informationen füllen, werden

vorgestellt und verbessert.

Während das Ziel der neuen Szenenrepräsentation die lückenlose Zusammenführung aufgenommener und generierter Daten ist, werden diese Fortschritte auf Kosten des Speicherplatzes erreicht, denn die von uns vorgestellte Darstellungsform speichert zumeist Rohdaten oder baut diese mit Metadaten weiter aus. Dabei bietet die Kombination aus echten und synthetischen Daten über Modell-basierte Kodierung ein einfaches Verfahren, um die Datenrate zu reduzieren. Dazu haben wir ein hybrides, Modell-basiertes Kodierverfahren entwickelt, das die Datenrate eines Videos signifikant reduzieren kann. Allerdings treten bei diesem Verfahren neuartige Probleme auf, die sich unter anderem in unzureichenden Fehlermetriken für synthetische oder kombinierte Inhalte finden lassen.

Diesem Problem tragen wir mit einer neuen Bilddatenbank Rechnung, die Fehler in der Szenenkomposition enthält und damit als Grundlage für die Entwicklung und Erprobung geeigneter Fehlermetriken dienen kann. Auf Basis dieser Bilddatenbank haben wir eine Fehlermetrik entwickelt, die bereits existierende Metriken um die Möglichkeit erweitert, Szenenkompositionsfehler zu erkennen und zu berücksichtigen.

Die meisten Vorteile der Bestrebungen, aufgenommene Daten wie synthetische Welten darzustellen, finden sich sicherlich in den vielfältigen Möglichkeiten der Nachbearbeitung. Verschiedene unterschiedliche Bearbeitungsmöglichkeiten werden in dieser Arbeit vorgestellt, von denen alle durch den vorgestellten Wandel in der Videoproduktion signifikant vereinfacht, einige erst ermöglicht werden. Beispielhaft dafür sind synthetische Kameraeffekte: Während eine echte Kamera in ihren Aufnahmemöglichkeiten stets an die Gesetze der Physik gebunden bleibt, kann eine virtuelle Kamera problemlos mehrere und frei geformte Fokusebenen oder unrealistische Bewegungsunschärfen umsetzen.

In dieser Arbeit werden verschiedene Forschungserfolge vorgestellt, die von uns entlang der kompletten Videoproduktionskette erzielt wurden und gemeinsam die Videoproduktion einen kleinen Schritt in Richtung synthetischer Welten - “Towards a Synthetic World” - weiterbringen.

Abstract

Visual media is stemming from two very different origins: One origin is the goal to visually depict real world observations, the other origin is the goal to create visually appealing pieces of art. These two origins are today represented by digital photography or videos as visualizations of real world objects and computer generated imagery which is often artistic. In recent years these two worlds have started to merge. Computational photography transfers ideas from computer graphics to captured data, and sophisticated rendering approaches recreate realistic visualizations from synthetic content. However, there is still a long way to go to achieve seamless transitions between captured and generated data.

When looking at captured and generated data an important observation is that in order to merge the best of both worlds it is desirable to transfer the flexibility of computer generated content to captured data, given the possibility of photo-realistic rendering. This thesis therefore introduces, discusses and enhances several steps towards a synthetic world. Novel data acquisition methods enable spatial information capture that is not constrained to visible light but captures other material properties as well. We have developed a representation which enables the combination of captured and generated data, and thus provides an important basis towards the seamless combination of both, captured and synthetic, worlds. Algorithmic approaches are presented and enhanced that feed the novel representation with information exceeding the data from capturing devices. In this domain we have enhanced superpixel segmentation for multidimensional data coming from novel data acquisition devices.

The introduced representation focuses primarily on the seamless integration of both previously separated worlds. This integration comes at the cost of storage, as data is mostly uncompressed or even expanded for better access. At the same time the combination of captured data with synthetically created objects can be used to reduce storage requirements very efficiently. We introduce and present a hybrid model based coding scheme which significantly reduces the data rate of video input. However, the model based coding scheme

faces a difficulty of assigning a frame quality score to a frame that consists of combined captured and synthetically generated content.

We address this problem of quality assessment for synthetic contents or contents combined from real and captured data with a novel image database designed for the development and testing of quality assessment metrics. This database is subjectively evaluated. Additionally, we have designed a metric which extends existing metrics by the ability to detect and consider scene composition errors.

The benefits of the ambition to bring captured data into the synthetic world are mainly visible in post-processing. A variety of processing steps are presented that are tremendously facilitated through a synthetic representation of captured content. A number of processing steps only become possible through novel acquisition and representation, for example rendering of physically impossible camera effects. Where before camera effects have been constrained by physics, virtual cameras can effortlessly implement several and warped depths of field or implausible motion blurs.

All along the processing chain of virtual content production we have made achievements, presented in this thesis, that bring video production a small step towards a synthetic world.

Acknowledgment

Working towards and writing this PhD thesis has defined a significant chapter of my life. The Telecommunications Lab at which I was allowed to work towards this goal has become a second home, colleagues and supervisors have extended my family. My sincere gratitude goes towards Prof. Dr.-Ing. Thorsten Herfet, who has been a fruitful source of academic ideas and patient partner for discussions, and who moreover appreciated and participated in extra-curricular activities like our lab band and sport events like annual company runs. For him the German word "Doktorvater" is a lot more appropriate than the English translation "doctoral thesis supervisor". My gratitude for scientific advice extends to the other reviewers of this thesis: I had the pleasure to get to know Prof. Dr. Adrian Hilton in the SCENE project where his thoughtful technical lead made the project a huge success. Prof. Dr. Philipp Slusallek kindly accepted me as the manager of research and operations at the Intel VCI which he directs together with Prof. Herfet, allowing me to implement and practice skills beyond academia. To both of them I am much obliged for taking the time and effort of reviewing my work.

In the environment of the Telecommunications Lab I like to mention my current colleagues, Yongtao Shuai, Nasimi Eldarov, Tobias Lange, Kelvin Chelli, Harini Priyadarshini Hariharan, Andreas Schmidt and Frank Waßmuth who have been helpful dialog partners in both, academic and private matters, and who have made my life at the Telecommunications Lab very enjoyable. Two very important persons at the Lab are Diane Chlupka and Zakaria Keshta, who have invested plenty of time to solve technical and administrative issues before they turned into problems. Previous members of the Lab and long time colleagues worth mentioning here are Manuel Gorius, whose love for music has stimulated the foundation of our Lab Band, and Michael Karl, with whom I had the pleasure to teach several weekend lectures.

Music plays an important role in my life, and I am grateful for those people who enabled me practicing music throughout my time at Saarland University. The Telecommunications Lab Band consists of the aforementioned Thorsten Herfet and Manuel Gorius, as well as Juliane Riedl, Christoph Ehre and Ma-

lika Picart. On university level Prof. Helmut Freitag as a musical director is responsible for the Choir and Orchestra of Saarland University - in both ensembles I have felt very comfortable.

At least as important as music and sports were the quiet moments spent at the Catholic student community on Campus. The two most important persons there have been Dr. Johannes Kreier who managed to create moments of silence and thought even during most hectic times, and Edeltraud Brändle whose commitment made the KHG a warm and welcoming place.

Additional persons are closely connected to my life at Saarland University, and I like to thankfully mention Jörg Schad, Raphael Reischuk, Andreas Schwarte and Hanjo Viets who have been housemates, sports exercising partners and most of all good friends.

Even though this thesis is based in an academic world, there are several people who are most valuable to me and to the present thesis. First and most important I like to cordially thank my lovely fiancé and soon-to-be wife Marlene Mohr for encouraging me to pursue and complete a PhD, to appreciate my work and to stick with me, even though I repeatedly postponed the promise to finish my work and move closer to her. I am very much obliged to my parents, Stefanie and Michael Haccius, who have enabled and motivated me to stay in academia as long as I wanted to search for further knowledge, and to my sisters Johanna and Carina who have encouraged my academic progress and are simply the best family I can wish for. This "best family I can wish for" will soon be extended by wonderful parents-in-law and three sisters-in-law, whose interest in my research and discussions I appreciate tremendously.

Numerous people have accompanied my PhD phase and have - knowingly and unknowingly - contributed to this thesis. I am very grateful for all the enjoyable encounters, meetings and discussions I have had during the previous years; this gratitude extends much further than the names contained in these two pages of acknowledgments.

Related Publications

[P1] C. Haccius, T. Herfet, "Computer Vision Performance and Image Quality Metrics: A Reciprocal Relation", International Conference on Image and Signal Processing (ISPR), Vienna, Austria, December 2016

[P2] C. Haccius, T. Herfet, "A Visual Reality Metric for Synthetic Contents", International Conference on Signal Processing (ICOSP), Limerick, Ireland, December 2016

[P3] C. Haccius, T. Herfet, "An Image Database for Design and Evaluation of Visual Quality Metrics in Synthetic Scenarios", International Conference on Image Analysis and Recognition (ICIAR), Pvoa de Varzim, Portugal, July 2016

[P4] C. Haccius, H. Hariharan, T. Herfet, J. Jachalsky, W. Putzke-Röming and T. Hach, "Infrared-Aided Superpixel Segmentation", International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM), Phoenix, USA, February 2015

[P5] C. Haccius, T. Herfet, "Shape Adaptive Sum of Euclidean Distances for Superpixel Clustering" (Short Paper), Conference on Visual Media Production (CVMP), London, Great Britain, November 2014

[P6] C. Haccius, S. Khangura and T. Herfet, "Enhancing MPEG for Model Based Coding", Networked and Electronic Media (NEM) Summit 2013, Nantes, France, October 2013

[P7] C. Haccius, T. Herfet, V. Matvienko, P. Eisert, I. Feldmann, A. Hilton, J. Guillemaut, M. Klaudiny, J. Jachalsky, and S. Rogmans, "A novel Scene Representation for Digital Media", Networked and Electronic Media (NEM) Summit 2013, Nantes, France, October 2013

[P8] C. Haccius, T. Herfet, V. Matvienko, "Representing and Presenting Digital 3D Scene Content, PROLIGHT Workshop, Nantes, France, October 2013

[P9] T. Herfet, C. Haccius, V. Matvienko and S. Fort, "A Novel Representation for Digital Scenes" (Poster), Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Anaheim, USA, August 2013

[P10] C. Haccius and T. Herfet, "Model Based Coding Revisited", Picture Coding Symposium (PCS), Krakow, Poland, May 2012

[P11] C. Haccius, "Developing a hybrid model based coding scheme for still images", Master thesis, Saarland University, Saarbrücken, Germany, 2013

Contents

Table of Contents xiii

List of Figures xvi

List of Tables xxi

List of Listings xxii

1 Introduction 1
- 1.1 Motivation and Outline . . . 3
- 1.2 Novel Capture Devices . . . 5
  - 1.2.1 3D from Multi-Focal Systems . . . 6
  - 1.2.2 3D from Depth Sensors . . . 8
  - 1.2.3 The Motion Scene Camera . . . 9

2 Merging Real and Synthetic Worlds 11
- 2.1 The Scene Representation Architecture . . . 12
  - 2.1.1 Base Layer . . . 12
  - 2.1.2 Scene Layer . . . 14
  - 2.1.3 Director's Layer . . . 16
  - 2.1.4 Implementation . . . 17
- 2.2 Adding Semantics to Scenes . . . 21
  - 2.2.1 Spatially and Temporally Consistent Acels . . . 22

3 Data Reduction 41
- 3.1 Model Simplification . . . 44
  - 3.1.1 Vertex Clustering . . . 44
  - 3.1.2 Vertex Decimation . . . 45
  - 3.1.3 Resampling . . . 47
  - 3.1.4 Mesh Approximation . . . 48
  - 3.1.5 Model Error . . . 50
  - 3.1.6 Texture Quality . . . 52

3.2 Model Re-Use . . . 56
3.2.1 The H.264 and H.265 Video Coding Standards . . . 57
3.2.2 Hybrid Model Based Video Coding . . . 62

4 Flexibility Gain **69**
4.1 Flexibility in Scene Composition . . . 69
4.1.1 Adding Synthetic Content . . . 70
4.1.2 Object Recoloring . . . 71
4.1.3 Object Retexturing . . . 71
4.1.4 Clothing Exchange . . . 72
4.1.5 Scene Relighting . . . 73
4.1.6 Pose Modification . . . 73
4.2 Rendering Mixed Scenes . . . 74
4.2.1 Free Viewpoint Rendering . . . 76
4.2.2 Depth of Field . . . 77
4.2.3 Motion Blur . . . 82
4.2.4 Making the impossible possible . . . 85

5 Image Quality Metrics **87**
5.1 Datasets . . . 89
5.1.1 Datasets with classical Errors . . . 89
5.1.2 Dataset with synthetic Errors . . . 95
5.2 Subjective Metrics . . . 102
5.2.1 Single Stimulus . . . 103
5.2.2 Double Stimulus . . . 104
5.2.3 Forced Choice . . . 105
5.2.4 Similarity Judgment . . . 105
5.2.5 Quality Ruler . . . 106
5.2.6 Quality Scale . . . 107
5.2.7 Timings and Durations . . . 108
5.2.8 Evaluating the Synthetic Image Database . . . 110
5.3 No-Reference Metrics . . . 118
5.3.1 Image Blur Detection . . . 119
5.3.2 Coding Artifact Detection . . . 120
5.3.3 Image Noise Detection . . . 120
5.4 Partial-Reference Metrics . . . 121
5.5 Full-Reference Metrics . . . 122
5.5.1 Qualtiy Metrics based on Image Statistics . . . 123
5.5.2 Quality Metrics based on the Human Visual System . . . 133
5.5.3 Quality Metrics based on the Human Cognitive System . 139
5.5.4 Results of SC-VQM . . . 147

6 Conclusion and Future Work **151**

Bibliography **156**

Glossary **167**

Index **171**

List of Figures

1.1 Prehistoric Examples of Art presenting abstract Patterns and Real World Objects . . . 2
1.2 Contemporary Examples of Art presenting abstract Computer Graphical Objects and Photographically Captured Real World Objects . . . 3
1.3 Triangulation of a Scene with Two Cameras . . . 6
1.4 Multifocal Camera Setups which can be used for 3D Motion Capture . . . 7
1.5 Depth Sensing Devices using the ToF technique . . . 9
1.6 The Motion Scene Camera extending an ARRI Alexa Camera by a Depth Sensor, built by ARRI as part of the SCENE Project 10

2.1 Layout of the Scene Representation Architecture . . . 13
2.2 Registering Several Images to Form a Panorama . . . 15
2.3 Implementation Structure of the SRA Backend . . . 18
2.4 SLIC Segmentations for Different Numbers of Superpixels k and Changing Weights Between Spatial and Color Distance l . . . 31
2.5 Subjective Motivation to Employ Infrared Information for Image Segmentation . . . 32
2.6 Different Parametrizations of the μ-Law Compressor . . . 35
2.7 Dataset showing Talent in front of Dark and White Background, consisting of Color, Infrared, Depth and Ground Truth Segmentation . . . 37
2.8 Comparison of Oversegmentation to Boundary Recall for Segmentation Based only on Color Information and Segmentation Based on Color and Infrared Information . . . 38
2.9 Comparison of Compactness to Boundary Recall for Segmentation Based only on Color Information and Segmentation Based on Color and Infrared Information . . . 39

3.1 Components of a Computer Modelled Car . . . 42

3.2 High-level Structure of Model Based Codec 44
3.3 Relation of Mesh Size and Mesh Quality 45
3.4 Mesh Simplification Approaches 46
3.5 Different kinds of Mesh Vertices 47
3.6 Regularization of Mesh by Resampling 48
3.7 Mesh Approximation using Geometric Primitives 49
3.8 Mesh and Texture Example 53
3.9 Distribution of Surface Normals of the Stanford Bunny and of a Performance Capture Actor 54
3.10 Residual Information per Model Error $\bar{e}$ at different Texture Quality Levels 55
3.11 Structure of a Model Based Image Coder 57
3.12 Basic Structure of a H.264 Encoder 60
3.13 Basic Structure of a H.265 Encoder 61
3.14 List Structure of Decoded Picture Buffer 61
3.15 List Structure of Encoded Picture Buffer for Model Based Enhancement 63
3.16 Basic Structure of the Model Based H.264 Encoder 64
3.17 Frames of a Mazda Video Sequence Input to the Model Based AVC Enhancement 66
3.18 Model Input to the Model Based AVC Enhancement 66
3.19 Visualized Difference of IDR Frame predicted by Model Based AVC Enhancement 66

4.1 Captured and Generated Data Combined 70
4.2 Recoloring a Spatially and Temporally Consistent Object 71
4.3 Retexturing a Spatially and Temporally Consistent Object . . . 72
4.4 Exchanging a Talents Clothing in Post-production 73
4.5 Relighting a Scene 73
4.6 Temporally Consistent Modification of a Talents Pose 74
4.7 Free View Point Rendering of 3D Object 76
4.8 Illustration of the Depth of Field Effect 77
4.9 Camera Models 78
4.10 Parameters Contributing to the Cirlce of Confusion 79
4.11 Size of the Circle of Confusion with respect to Object Distance . 79
4.12 Inputs and Rendering for Synthetic Depth-of-Field 81
4.13 The Depth of Field Effect Allows to Look Behind Objects . . . 81
4.14 Relation between Exposure Time and Lens Opening 82
4.15 Motion Blur is Created from Several Instances of a Moving Object 84
4.16 Separation into Several Frames causes Loss of Information . . . 85
4.17 Examples for Synthetic Depth of Field 86

5.1 Example of failing PSNR for Image Quality Assessment 89
5.2 Images of the Kodak Lossless True Color Image Suite used for LIVE Database and TID 91
5.3 Blocking Artifacts created by JPEG Compression 92
5.4 Compression Artifacts created by JPEG2000 Compression . . . 93
5.5 Average White Gaussian Noise 94
5.6 2D Gaussian Kernel 95
5.7 Gaussian Blur 95
5.8 Images with Synthetic Content for Generation of Synthetic Errors 96
5.9 Scaled Car 98
5.10 Translated Car 99
5.11 Rotated Car 101
5.12 Structure of a Test Session for Subjective Quality Assessments . 103
5.13 Structure of a Single Stimulus Experiment 104
5.14 Structure of a Double Stimulus Experiment 105
5.15 Structure of a Forced Choice Experiment 106
5.16 Structure of a Similarity Judgment Experiment 106
5.17 Structure of a Quality Ruler Experiment 107
5.18 Quality Scales 109
5.19 Histogram of Raw Assessor Scores on Quasi-Continuous Scale . 109
5.20 Structure of Experiment to Obtain Subjective Evaluations . . . 111
5.21 User Statistics from the SID2015 Evaluation 114
5.22 Display Size vs. Viewing Distance 115
5.23 Ideal MOS for Different Error Types 116
5.24 PSNR vs. MOS for Classical and Novel Image Errors 117
5.25 Structure of the Synthetic Image Database 118
5.26 Depth and Segmentation for the Chess Scene 118
5.27 Structure of a No-Reference Image Quality Metric 119
5.28 Structure of a Partial-Reference Image Quality Metric 121
5.29 Comparison of Wavelet Coefficient Histograms 122
5.30 Structure of a Full-Reference Image Quality Metric 123
5.31 Calculating the Ratio of Averages from Individual Ratios 134
5.32 Schematic Diagram of the Human Eye 136
5.33 Architecture of the Sarnoff JND Model 137
5.34 Structure of SSIM Metric 139
5.35 Structure of HDR-VDP 2 140
5.36 Optical Illusions illustrating the Human Visual System and the Human Cognitive System 140
5.37 Structure of the Proposed SC-VQM 141
5.38 Example Image Set Illustrating the Implementation of SC-VQM 142

5.39 Mask and Environment of Erroneous Object 143
5.40 SIFT Matching between Test and Reference Image 143
5.41 Filled Background and Registered Object 144
5.42 SSIM Before and After Object Registration 145
5.43 Test Images for Calibration 146
5.44 Visualization of Free Parameters from Calibration 147

List of Tables

2.1 Boundary Recall of Superpixel Segmentations with 200 superpixels using only Color Information, Color and Infrared Information and Color, Infrared and Depth Information of the Talent standing in front of the Dark and White Background 40

3.1 Sizes of Components Contributing to Model-Based Encoded Content 56
3.2 Comparison of AVC reference software to Model Based AVC enhancement on a synthetic scene 64
3.3 Comparison of data requirements for video sequence encoded with AVC reference software and Model Based AVC Enhancement using only a model and using model and background information 67
3.4 Comparison of AVC reference software to Model Based AVC enhancement on a synthetic scene with scene changes after every second video frame 67

5.1 Comparison of LIVE to TID Database 90
5.2 Duration of Subjective Evaluation Experiments for n test images 110
5.3 Spearman - Correlation between MOS and existing metrics . . . 116
5.4 Kendall - Correlation between MOS and existing metrics 116
5.5 Spearman - Correlation between MOS and PSNR, SSIM, HDR-VDP 2 and SC-VQM 148
5.6 Kendall - Correlation between MOS and PSNR, SSIM, HDR-VDP 2 and SC-VQM 149

List of Listings

2.1 Including SRA Front-End Headers 20
2.2 Initializing Scene in SRA 20
2.3 Adding Meshes to the SRA 20
2.4 Retrieving a Mesh from the SRA 21
2.5 SLIC Superpixel Clustering - Algorithm Inputs 28
2.6 SLIC Superpixel Clustering - Initial Center Positioning 28
2.7 SLIC Superpixel Clustering - Distance Calculation 29
2.8 SLIC Superpixel Clustering - Pixel Assignment 29
2.9 SLIC Superpixel Clustering - Update Cluster Centers 30
5.1 Generating JPEG Compression Artifacts 92
5.2 Generating JPEG2000 Compression Artifacts 93
5.3 Modeling White Gaussian Noise 94
5.4 Modeling Gaussian Blur 95
5.5 Object Scaling Error 97
5.6 Object Translation Error 98
5.7 Object Rotation Error 100

Chapter 1

Introduction

Images and human beings are linked inseparably. According to paleontologists the modern human being is defined by the ability to think creatively, and creatively designed objects which have a purpose exceeding the demands for evolutionary fitness. Among the oldest currently known pieces of art are patterns carved into stone (findings from the Blombos-Cave in South Africa dating back to about 75.000 BC) and cave art (found in Spain, France, Indonesia and Africa, all dating back to roughly 40.000 BC).

These very first pieces of art known today already exhibit two fundamental differences. Some carvings and cave paintings present only abstract patterns or shapes (see Figure 1.1a), others illustrate real world objects such as animals or humans (as shown in Figure 1.1b).

Throughout history artists have tried to create look-alike replications of the world they perceived. Portraits were painted to capture the appearance of people, landscape paintings were persisting nature and historical moments were replicated in monumental illustrations. At the same time, artists also used their creative freedom to paint people the way they wanted them to be seen, added emotions to natural scenes or altered historical events to their liking.

The era of imaging changed with the advance of photography. While precursors of cameras have been used since the 11th century, means to capture and persist images were introduced in the 19th century only. During the 20th century the initially large and bulky cameras were reduced in size, leading to a wide distribution of candid cameras. All of a sudden every common person had the ability to persist people, landscapes and moments in images. However, the initial flexibility of artists to adjust images freely to their likings was reduced to rudimentary camera settings such as focal distance and exposure time.

At the turn of the millennium digital camera devices became widely available. While the early digital cameras were visibly deficient with respect to spa-

(a) Pattern engraved in ochre, found in the Blombos Cave, South Africa, and dating back to 75.000BC ©Chris Henshilwood

(b) Cave Painting of a Hyena, found in the Chauvet Cave, France, and dating back to 30.000BC ©Carla Hufstedler

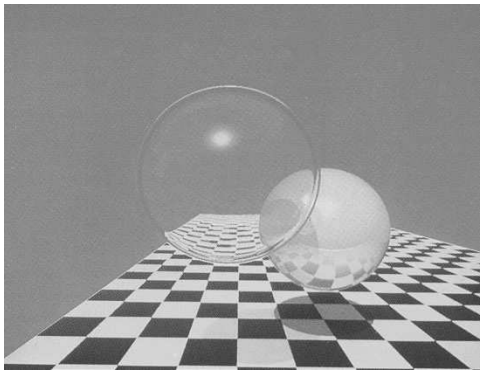
Figure 1.1: Prehistoric Examples of Art presenting abstract Patterns and Real World Objects

tial resolution compared to the previously used analog cameras, they advanced quickly and have meanwhile fully overcome these initial resolution problems.

Looking back at the desires of the first humans to create art, digital photography well covers the desires to represent objects and scenarios from the real world. The creation of abstract patterns, shapes and knowingly unreal content, which has been available and used by many artists (e.g. surreal artists like Salvador Dalí, Max Ernst or cubistic painters like Pablo Picasso, Georges Braque) could, however, not be implemented by camera systems.

In the digital age designs and graphics created on the computer best match the creative flexibility of artists. This domain is summarized in the term "computer generated imagery" (CGI). CGI evolved in the middle of the 20th century, when the first computers at the MIT were able to obtain user input and display visual information. In the following decades several inventions made CGI available to the broader public. Input devices like mice and sketchpads facilitate graphical input, increasingly potent processing units allow the computation of graphical content and constantly advancing display devices present computer graphical content to users. Examples of the first digital versions of art showing abstract designed shapes and replications of real world objects are shown in Figure 1.2.

Since the beginning of art there have been two worlds: representations of real objects and purely artistic creations. Today these worlds are represented by digitally captured and computer generated content, and both words are still largely separated.



(a) One of the first Images showing different Light Reflectancies computed with recursive Ray Tracing in 1979 [106]

(b) First Digital Photo created in 1975 by Russell Kirsch by Scanning a Photo of his 3-month old Son [15]

Figure 1.2: Contemporary Examples of Art presenting abstract Computer Graphical Objects and Photographically Captured Real World Objects

1.1 Motivation and Outline

Computer graphics and captured data have presented separate worlds for several decades. At the same time, the best of both worlds is desired by users. Users want to have the realism that can be achieved by captured, real data, and the flexibility and creative opportunities that come with computer graphical content. The desire to merge captured data with synthetic content to gain additional flexibility becomes apparent when looking at the production process of movie features. Traditionally, on set real data is captured. In a design studio computer graphical objects are created, and movie post-production merges the captured data with the synthetic content and special effects. If at this point in time a movie director notices that parts of a scene do not match his requirements, options are very limited. To a certain degree elements can be cut out of a scene, or color settings can be adjusted. However, in many cases, the director needs a different take, thus needs to go back to the set and shoot a scene again. This process is that time expensive that it is usually impossible to re-shoot takes after the shooting phase. In order to avoid such situations scenes are shot multiple times with different settings to provide sufficient footage to choose from, and if no suitable material is available content cannot be adjusted.

Bridging the gap between computer graphics and captured image or video content is an essential prerequisite to add the desired creative flexibility to real captured data and the desired realism to synthetic data. Traditional captured

data lacks two important characteristics which synthetic data has, which are 3D information and semantically meaningful objects. Adding such information to captured data therefore already presents a significant step towards a world uniting captured and synthetic content. Additional information can partly be generated algorithmically, but an important tool for additional data are data acquisition devices.

In the following Section of this Introduction some of the technical advances which enable acquisition of additional scene information are presented. These technical devices described here represent the data sources used in the other chapters of this work. Chapter 2 than addresses the fundamental problem of pooling real captured data and synthetic computer generated data. Core contributions here have been made with respect to the way a united world can be represented (see also Publications [P7, P9]) and to an approach adding semantics to captured data (see Publication [P4]).

Bringing real and synthetic data together has two fundamental goals. One the one hand it increases post-processing flexibility of the captured data, on the other hand a union of both worlds can be used to reduce the data rates required for visual information storage. Chapter 3 focuses on the latter one. The term Model Based Coding describes the use of synthetic model information for more efficient image and video coding. Important contributions here have been made for still images (see Publication [P10]) and with an enhancement of current state-of-the-art video codecs to benefit from synthetic model information (see Pubication [P6]). While the application scenarios of the presented model based coding approach is very limited and depends on several factors, the research conducted reveals major obstacles that need to be taken in order to create a robust and beneficial model based coding scheme. One of these obstacles is an error metric which conforms to the human perception in cases where synthetic and real contents are merged, which is addressed in Chapter 5 of this work.

Chapter 4 addresses the first of the two benefits mentioned above: increase in flexibility. Several technologies developed in the SCENE Project [24] built on the Scene Representation introduced in 2. These technologies and their application in a use case employing the developments introduced in the previous Chapters are introduced and explained.

Users commonly perceive synthetic visual information rendered in a certain way. How this rendering process is done is critical for the perceived realism of a scene. This holds even more for content stemming from both worlds; real and synthetic sources. Chapter 4.2 deals with the challenges arising when reproducing realistic camera effects for real and computer generated content (see Publication [P8]). In addition to realistic camera effects synthetic cameras introduce additional artistic freedom. Some of these new flexibilities arising

from synthetic camera models are explored and explained in this chapter.

Research described in Chapter 3 reveals that the perceived visual quality is a critical factor for the use of visual information. Such visual quality depends on the subjectively perceived quality users experience when viewing given information. Subjective quality assessment is, however, in many scenarios not possible. Chapter 5 explores available error metrics which employ statistical image features or are based on the human visual system. The existing metrics are extended by a metric considering the human cognitive system (see Publications [P1, P2]). For the creation of this metric existing image databases had to be extended by an image database exhibiting novel error features which may occur as soon as real and synthetic content are united in a single representation (see Publication [P2]).

1.2 Novel Capture Devices

Traditional imaging sensors and systems have focused on the information that is at the core interest for a 2D color representation: color and 2D spatial information. These image sensing systems needed to be designed under consideration of physical constraints. For cameras, a fundamental bottleneck is the image sensor (or previously the film) which requires a certain amount of light input. An image sensor consists of numerous densely packed light sensitive blocks which capture light of a predefined wavelength (usually either red, green or blue). The amount of light energy reaching one of those pixel sensors can be increased by either increasing the spatial extension of a collector (reducing the spatial resolution of an image), increasing the integration time (potentially introducing motion blur) or focusing more light with a larger lens opening onto the sensor (reducing the focal depth). More details about these camera parameters are provided in the context of realistic rendering of camera effects in Chapter 4.2 of this work.

Ongoing research has advanced sensor technology to a level where capturing sufficient light intensity is in many capture scenarios not a critical aspect any more. All of a sudden image and movie acquisition approaches have the ability to capture information exceeding the requirements of 2D images, thus adding data to the knowledge about a scene composition. Today several data acquisition systems are available which add information about a third spatial dimension, augment visible light by invisible electromagnetic waves of various length (infra-red, ultra-violet or thermal) and overall enrich information of captured images or image sequences.

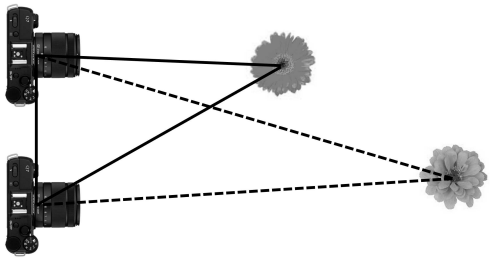


Figure 1.3: Triangulation of a Scene with Two Cameras

1.2.1 3D from Multi-Focal Systems

Multifocal Systems in general are setups capturing the 3D performance of one or multiple talents in a scene using a setup of several calibrated cameras. Depth information in such setups is obtained in a process named triangulation: with knowledge about the distance between two cameras a triangle with those cameras and a third scene point seen by both cameras can be computed. In this triangle, the distance between the point in the scene and both cameras can be computed (see Figure 1.3).

Initially multifocal setups required a large number of calibrated cameras and additional markers or tracking points in a scene. An early version of a marker based motion capture system was proposed by Munkelt et al. in 1998 [63]. According to an overview study conducted two years later by Moeslund and Granum available motion capture systems were based on a set of assumptions, which restrict the size of the workspace, camera and talent motion as well as environment and required markers in a setup [62].

Marker-based motion capture using multifocal systems was successfully employed for motion picture productions since 1998 (among others characters of the first three Star Wars Episodes were created with motion capture techniques). Compared to marker-based motion capture, markerless motion capture however offers additional freedom and flexibility while coming at the cost of higher complexity. One of the first markerless performance capture systems was introduced by De Aguiar et al. in 2008 [13]. De Aquiar et al. use a setup of 8 calibrated cameras to capture a talent from several directions over time. These multiple views are algorithmically combined into a 3D model, which performs the same actions as the real world antetype.

Calibrating multiple cameras for a performance capture setup is a time expensive problem and limits the flexibility of the setup as a whole. For motion picture capture, however, usually a high flexibility is required, as talents are captured in arbitrary environments and with unrestricted motion. This desire

(a) A Trifocal Camera Rig combining an ARRI Alexa Camera with two Indie GS2K Cameras, build by the Fraunhofer HHI as part of the 3FLEX Project [30]

(b) P2 HPX170 Pro 3D Stereo Camera produced by Panasonic

Figure 1.4: Multifocal Camera Setups which can be used for 3D Motion Capture

for higher flexibility can be satisfied with setups requiring fewer cameras. Wu et al. have enhanced the approach initialised by De Aguiar et al. to work with a stereo camera [109]. Stereo cameras have the advantage that they need to be calibrated only once and remain in a fixed setup for usage. Trifocal- and stereo-cameras (combining three or two lenses respectively in a fixed setup) are available to the market. Figure 1.4 shows a trifocal system build by the Fraunhofer HHI as part of the 3FLEX Project [30] and the P2 HPX170 3D Stereo Camera produced by Panasonic.

Light field cameras take the idea of capturing scene content by a multifocal system to the next level. Light field camera arrays were constructed among others by the Computer Graphics Laboratory at Stanford University, with 17×17 cameras assembled in a grid resulting in 289 views of the same scene content[1]. Systems using only a single device and splitting the information captured on a single CMOS sensor by multifocal lenses have reached the consumer market with the Lytro Illum[2]. While such camera systems capture significantly more data than the aforementioned stereo or trifocal systems, the conceptual idea of reconstructing and exploiting spatial information from multiple viewpoints remains the same.

[1]The Stanford Light Field Archive, `http://lightfield.stanford.edu/`, accessed 20-Apr-2016

[2]Lytro Illum, A new way to capture your stories, `https://illum.lytro.com`, accessed 20-Apr-2016

1.2.2 3D from Depth Sensors

An alternative approach to triangulation as described above is the so called time-of-flight measurement. Technologies to determine the distance of an object by measuring the time light takes to travel between object and measurement device are not new. Since the 1960's military adopted laser range finders in battle tanks, and today laser range finders are used in numerous fields of application like construction, sports or landscaping. To use time-of-flight (ToF) information for a per-pixel depth measurement and therefore create a depth image was first employed in 2000 with the release of the Z-Cam. Different from the original laser range finders, ToF-cameras like the Z-Cam use infrared light emitted by active infrared lights sources at the measuring device.

The game market has always been a driving factor in the development of novel technologies. Therefore it is not surprising that the XBox game console was enhanced by Microsoft with a depth sensor based on the ToF principle, the Kinect. Equipped with a color sensor, a ToF depth sensor and multiple cameras the Kinect allows motion and sound input to games played on the XBox. A Kinect module is displayed in Figure 1.5a.

Similar to the Microsoft Kinect Intel released a camera array consisting of a color sensor and a ToF depth sensor. This Intel RealSense (see Figure 1.5b) was designed to allow motion interaction, and image acquisition and processing beyond the traditional 2D color information using standard notebooks, tablets or smart phones.

ToF cameras can be employed for real time motion capture [27] as well as for 3D shape scanning [10]. While the spatial resolution in all three dimensions does not yet match the precision achieved with triangulation techniques, ToF comes with a large benefit of flexibility. No color images need to be calibrated, the only calibration step necessary is to align color and depth information coming from two sensors.

In addition to the depth information ToF cameras often capture an infrared image. This information is captured quasi as a by-product of the ToF measurement process, where infrared light is required for depth sensing. While the infrared information is not directly required for any visual output, all additional information about a scene can be used to deduce further interesting scene knowledge. Details about such a process are given in Chapter 2.2.

An interesting observation is that depth from triangulation and depth from time-of-flight both have advantages and shortcomings which compensate each other. A significant weakness of the triangulation approach are plane, unstructured surfaces, as a triangulation algorithm requires detectable image features. On plane, unstructured surfaces the ToF-technique, however, produces best results, as no noise inducing obstacles or structures distort the distance mea-

(a) The Microsoft Kinect sensor used for Motion and Sound Control of the XBox game console

(b) The Intel RealSense camera array which adds a third spatial dimension to notebooks, tablets or smartphones

Figure 1.5: Depth Sensing Devices using the ToF technique

surements. On the other hand, object boundaries and significant structures in images often do distort ToF measurements wherever they occur. At the same time, object boundaries and structures are most important for successful triangulation. Current technologies combine the features of both approaches by adding a ToF sensor to a stereo camera pair (see Figure 1.5b, the Intel RealSense camera array).

1.2.3 The Motion Scene Camera

Surveys of camera men have revealed that professional camera operators prefer not to do any - potentially error prone - calibration steps which distracts from their core work of shooting a movie. Any additional sensors, independent of color or depth sensors, should therefore be contained in a single system that can be operated like a single traditional camera. In addition to that, having a depth sensor spatially separated from a color sensor leads to areas which are seen by one of the cameras only, thus causing shadows: either depth pixels with no color information or color pixels with no depth information. The same holds for stereo- or tri-focal camera systems: triangulation is impossible for areas only visible to one camera.

The camera producer ARRI has solved these problems with the Motion Scene Camera [35]. Here a depth sensor is integrated into the same camera and views a scene through the same optical system as the color sensor. This is possible since color and depth sensor capture light of different wavelength: the color sensor is restricted to visible light from 400nm to 700nm while the depth sensor measures the traveling time of near infrared lightm between 700nm and 1400nm. A hot mirror filter can therefore be employed as a beam splitter in the optical system of the Motion Scene Camera. Color and depth information

Figure 1.6: The Motion Scene Camera extending an ARRI Alexa Camera by a Depth Sensor, built by ARRI as part of the SCENE Project [24]

acquired by the Motion Scene Camera is therefore not only temporally, but also spatially perfectly aligned and does not require any re-calibration before usage. Material captured by the Motion Scene Camera is used as footage for the algorithms presented in Chapter 4 and as input to the representation introduced in Chapter 2.

Chapter 2

Merging Real and Synthetic Worlds

Traditionally, visual data comes from a single source and is only used in a single domain as well. Real images or videos are recorded by a camera or other acquisition device. 3D models and animation data is usually computer generated. Previously, these two domains of data barely touched. Traditional motion pictures are a good example of real data that is used as pure captured data and played as such. Traditional video games serve as a good example for the opposite, data which is computer-generated, and remains computer generated until rendering.

In recent movie productions and video games a combination of both worlds is pursued. Movies use special effects which require computer generated content, video games use characters which are captured in the real world with cameras and cut-scenes in video games are produced like traditional movies. Nevertheless, both kinds of data still remain largely independent for the major part of a visual media production process.

Bringing both worlds together is a difficult process. Neither captured video nor computer generated content was meant to be merged with the other. Two of the most prominent obstacles when bringing captured and computer generated data together are that captured video lacks any spatial information to be integrated with computer generated data, and that captured data lacks the semantics necessary for a proper integration. The SCENE project [24] was designed to - among others - tackle these problems of merging real and synthetic data.

The Scene Representation Architecture was designed with the purpose of providing an environment to combine captured and computer generated content. The following section describes the conceptual layout and implementation of this architecture, which enables combined processing of data coming

from multiple sources [P8].

Having a format available which allows merging manifold data sources, however, is not sufficient. On the one hand, dimensional deficiencies of captured data need to be overcome. Novel acquisition hardware (as introduced in Section 1.2) paves the road to full 3D data acquisition which facilitates merging captured and computer generated data. Algorithms which create temporal and spatial consistencies and semantics (as they are inherently present in computer generated data) are relevant to allow the same post-processing steps for both, captured and generated data. Such algorithms are presented in Section 2.2.

2.1 The Scene Representation Architecture

The problem of merging captured and generated data is to a large extend the problem of finding an adequate architecture to represent both, real and synthetic data. In the SCENE project an architecture was designed, implemented and tested. This architecture is called the Scene Representation Architecture (short SRA) [P7].

Captured 2-dimensional data has a pixel as the smallest unit, and 3-dimensional implementations like to employ voxels as building blocks for the presented contend. When synthetic data is created, computer artists or algorithms usually place vertices in 2D or 3D space and connect these with lines or curves. For the newly created architecture therefore an elementary task was to define the smallest unit on which a merged representations of either captured or computer generated data can be based.

As a novel building block for scenes with merged contents the *acel* is defined, which abbreviates **a**tomic s**c**ene **el**ement. The definition and implementation of such acels will be given later in this chapter (see Section 2.1.1).

Figure 2.1 shows the layout and a rough idea of the Scene Representation Architecture. The architecture is conceptually divided into three layers, which are called *Base Layer*, *Scene Layer* and *Director's Layer*. These individual layers are described in the following sections.

2.1.1 Base Layer

The Base Layer contains the most basic information a scene is consists of. It can be thought of as the repository of scene elements, which can be combined in a multidimensional scene. These base elements are an unordered collection of assets, fully independent of each other. That means, that neither their coordinate systems, nor their color settings or their registration points need

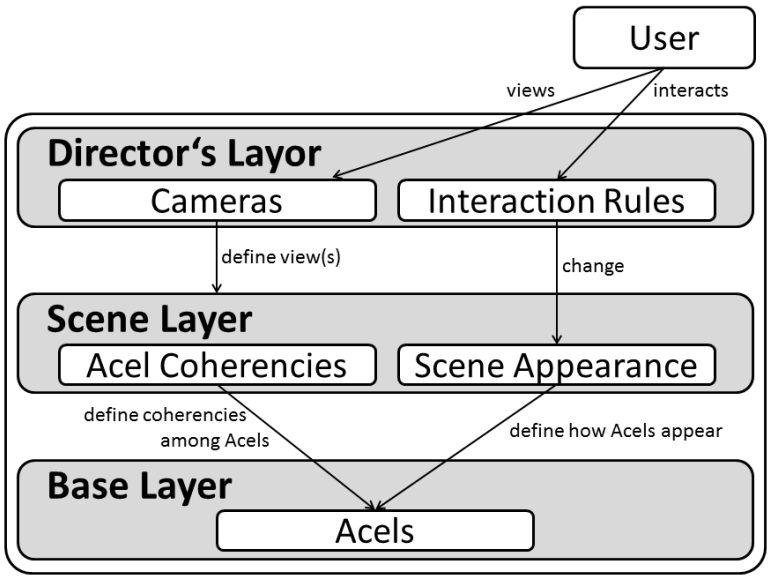


Figure 2.1: Layout of the Scene Representation Architecture

to be coordinated yet. These unordered and inconsistent scene assets are the above mentioned atomic scene elements, or acels. In terms of a motion picture production the base layer is the requisite, the collection of items which are used in a movie.

Acels

Acels are the smallest elements a scene can be composed of. These smallest elements can come from either captured data, computer generated data or any intermediate processing step. Therefore it is important to not restrict the architecture to one kind or the other. Acels can be either, they can be single pixels, or vertices, but they are also not constrained to either. An acel in general is understood to be the smallest unit, and the kind of this unit depends to a large extend on the source. Therefore, combinations of pixels like image patches, whole meshes, video frames or even whole videos or animations are allowed as acels as well.

The smaller acels are, the more flexible are the scenes composed of these acels. The inverse holds true for the amount of information associated with an acel. While acels are allowed to be single values representing only a color or a location in space or time, single-valued acels have little meaning. Larger acels which combine a color with a certain location at a given time, or even

collections of such points with a semantic meaning in a single acel are more valuable with respect to the information they contain, but less flexible.

2.1.2 Scene Layer

The Scene Layer is responsible for combining any of the assets given in the Base Layer into a coherent scene or parts of a scene. Employing once again the analogy to a motion picture production the Scene Layer corresponds to the stage or setting, in which requisites are placed. All requisites can be placed, moved and combined freely in this global setting, any kind of grouping can be done here. Individual pixels can be combined into frames, frames can be combined into videos, vertices and curves can be merged into meshes, or complete scenes can be combined from meshes and videos.

Registering different uncoordinated acels in a global coordinate system changes the appearance of an acel. Acels can be up- or downscaled to match the global coordinate system, they can be translated or rotated, their color balance and their temporal settings can be adjusted. A scene contains by default as many dimensions as the combination of all acels, and each acel needs to be registered with respect to all dimensions of the coordinate system.

Acel Coherencies

Groupings of acels are stored as coherencies of acels. Acels can be coherent with respect to any common dimension. Coherency is not necessarily a fixed value, but can be given as a probability, or strength of the group. A group of coherent acels can again be coherent to another acel or group of other acels. This creates a graph structure in a scene, similar to those proposed by Sowizral and Nadeu [65, 95]. The smallest elements or leaves of these graphs are always the acels stored in the base layer, whereas the root of the scene graph is given by the global coordinate system of the full scene.

Scene Appearance

As soon as several acels are grouped by coherencies into a larger coordinate system, the individual acel dimensions need to be registered with respect to the global coordinates. This changes the appearance of the individual acels. Consider for example multiple images of the same view which shall be stitched together to a panorama view (see Figure 2.2). As the individual images are taken at slightly different times, maybe with different cameras and with different angles towards light sources their color settings have changed. Registering those images in the same global coordinate system does not only require to set

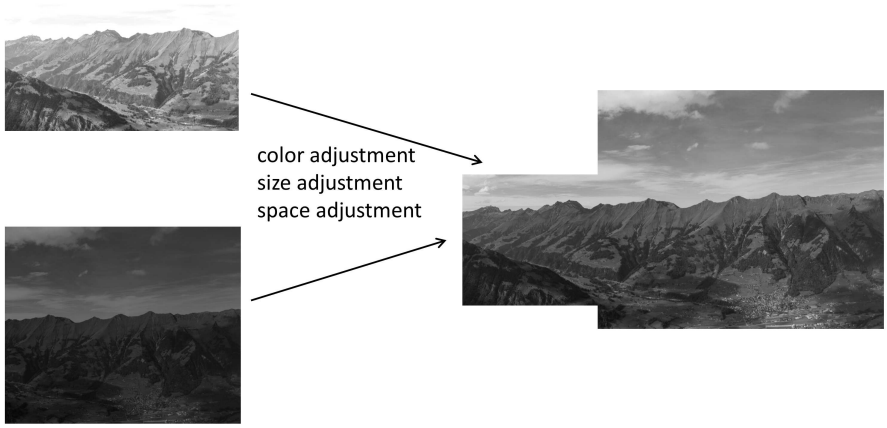


Figure 2.2: Registering Several Images to Form a Panorama

the spatial offset of those images, but also requires to adjust the color offset, thus changing the appearance of said acels.

Scene Dimensions

A scene is a composition of acels or groups of acels. When acels are joined, the group of coherent acels necessarily maintains all the dimensions of the contributing acels. Therefore the dimensions of a scene are the set union of individual acel dimensions. For K acels the set of dimensions of acel k with $0 < k \leq K$ is denoted by D_k. Then the scene composed of these acels has dimensions D_S with

$$D_S = \bigcup D_k \forall \{k | 0 < k \leq K\} \tag{2.1}$$

According to this joint sum, the dimensions of a scene are necessarily a superset of the dimensions of individual acels.

$$D_S \supseteq D_k \forall \{k | 0 < k \leq K\} \tag{2.2}$$

Exemplary, let X, Y, Z denote spatial dimensions, T a temporal dimension, and R, G, B color dimensions. If a mesh defined in dimensions $D_1 = \{X, Y, Z, R, G, B\}$ forms a coherent group with a video defined in dimensions $D_2 = \{X, Y, R, G, B, T\}$ then this coherent group of video and mesh needs to be defined over the set union of the individual acel dimensions, which is $D_S = \{X, Y, Z, R, G, B, T\}$. That means, for a coherent scene representation a time needs to be defined for the mesh, and a third spatial dimension needs to be set for the video.

All dimensions which are not initially present in the original acels need to be set, such that the acels can be fully registered in the global coordinate system. For the example given above a trivial assumption would be a fixed depth for the video, like an on-screen projection, and an infinite time for the mesh, thus making it available in all frames.

2.1.3 Director's Layer

The Director's Layer implements the decisions how a consumer or user is allowed to perceive a scene. It therefore directly corresponds to the role of a director in a motion picture production. In the same way that the director decides what parts of a setting shall be seen and how they shall be seen, the Director's Layer of the Scene Representation defines exactly these choices. With the novel representation also new options compared to traditional motion picture productions become available to the director. He can now not only decide what consumers will see, he may also define interactive content and specify allowed interaction.

Such interaction is open to all stages beyond the production. In the exemplary case of a production intended for broadcast interaction rules in the Director's Layer can allow the local broadcaster to update product placement and in-scene advertising to the local requirements. Further interaction rules may allow the final consumer of the production to change camera angles or color settings, limited to a range where the producer remains in control of the content.

Cameras

Cameras play an essential role with respect to the realistic perception of scene content, as they define the "eye" through which a consumer views a scene. A part of a chapter is dedicated to realistic rendering of scene content, see Chapter 4.2. Most important from the aspect of the Scene Representation Architecture is the use of cameras to perceive scene content. One or several virtual cameras can be placed in the scene composited in the Scene Layer (see 2.1.2). Applying the same camera characteristics to all elements viewed by the camera, independent of the elements source, e.g. real captured images or videos or computer generated content, is crucial for the realistic perception of the content.

Assuming a sufficiently high quality and completeness of the acel data there are no restrictions as the where a camera can be located in the virtual scene. For most real settings, however, camera placement remains constrained. Exemplary, if footage is obtained by a single camera like the Motion Scene

Camera (Section 1.2.3), occlusions occur in a scene. Such occlusions limit virtual camera placement to positions on or close to the optical axis of the real camera.

Interaction Rules

Having full 3D scenes with scene semantics enables different forms of interaction with the content. After media is produced in the SRA-Format it can be modified by any number of intermediates and the final consumer. Intermediates are, for example, broadcasters or service providers. While certain types of scene interaction and modification may be allowed or even desired, other forms of interaction need to be prohibited. A broadcaster could be allowed to place regional items in a scene or update commercial items for product placement, and the final consumer may be allowed to change the camera angle and exchange certain object types to his liking. However, neither broadcaster nor consumer shall usually be permitted to make more drastic changes to a scene.

The SRA therefore allows to specify interaction rules for each object in a scene, considering all dimensions of a scene. By default, interaction is not allowed and is only enabled by specifying an interaction rule for a specific object and a specific scene dimension.

2.1.4 Implementation

The SRA is implemented as a data structure and the necessary procedural interface to access it. The SRA is implemented in C++ and was tested under Windows and Ubuntu operating systems. Further more, it is provided as a Visual Studio and Eclipse project, thus integrating into the most common programming interfaces. The SRA is structured into two main parts, the back-end which provides a data structure to persist all necessary information and a front-end to access the back-end data structures.

The back-end provides all the necessary data structure to store the fundamental types of data relevant to a scene representation. Core to a scene is its structure in the SRA. The scene contains acels, which are again compositions of further acels, values and properties. Values can be any data or data array. Properties contain a descriptive name and further acels assigning values and properties in a tree structure. Such a structure is presented in Figure 2.3.

Any kind of persistency structure can be employed behind the back-end. Currently the Google Prototype Buffers are implemented to persist scene data. This is, however, not restrictive: any other persistency structure which supports structuring elements as visualized in Figure 2.3 is applicable. The SRA

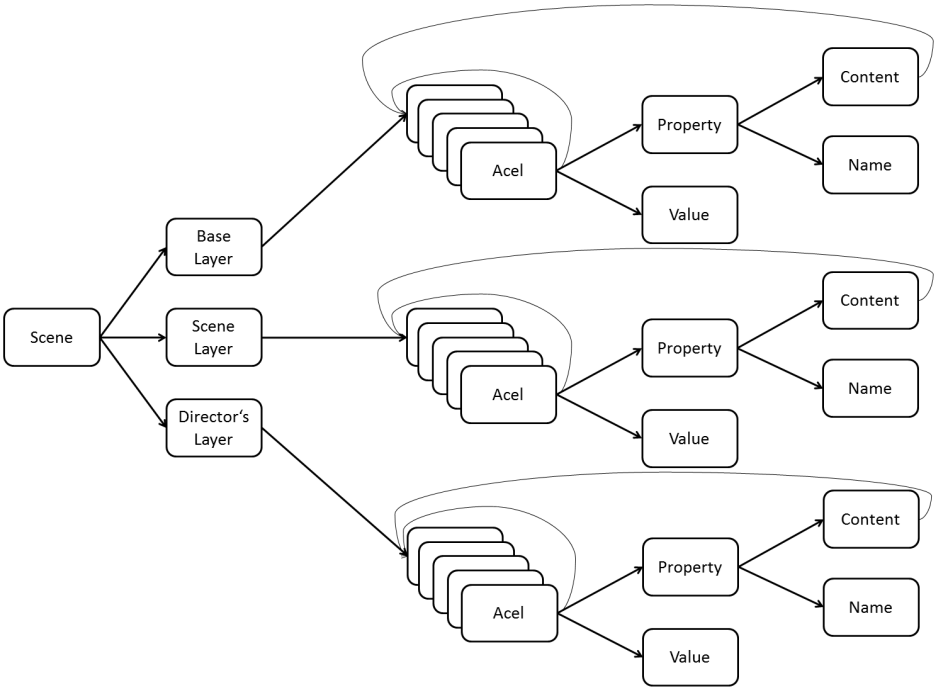


Figure 2.3: Implementation Structure of the SRA Backend

back-end is accessed through a header file, which can be included into applications directly working on the back-end and is employed by the SRA front-end.

The SRA front-end structures and facilitates the use of the SRA back-end. Since acels in the SRA are ID based, the front-end offers an ID management system. This ID management system allows users to add, access and remove content to and from the SRA independent of knowledge about available IDs. Furthermore, several intuitive functions to write and retrieve various common data types like images, videos, meshes or calibration matrices to and from the SRA are exposed.

The functionality of the SRA is demonstrated in different small use cases, showing how data can be written, accessed and visualized to and from the SRA. To work with the SRA front-end the *sra.h* header file needs to be included. The following small unit tests validate different use cases of the SRA.

Load and Write Camera Projections This unit test has two files, *LoadCameraProjectionTest* and *WriteCameraProjectionTest*. It showcases and explains how camera projection matrices can be stored in the SRA and retrieved from the SRA. A camera projection matrix is a 3×3 matrix of integer or double values. Two examples demonstrate a simple and a more complex use

of writing such a matrix to the SRA in a structured form.

Load and Write EXR Streams This unit test consists of the files *LoadEXRStreamTest* and *WriteEXRStreamTest*. It demonstrates the use of EXR streams in the SRA. The EXR format allows storing multi-channel maps, which becomes relevant when storing RGB plus depth, infrared or any additional channel information in a single frame. For EXR data the OpenCV library is required.

Load and Write Data Streams Compared to the above mentioned use case this example demonstrates the storage of arbitrary consecutive files in a stream. It consists of two files, *LoadStreamTest* and *WriteStreamTest*. Streams extend the SRA structure by a common format to represent consecutive data, e.g. frames in a video.

Load and Write Meshes This example is contained in the file *Per"-sis"-tance"-API-Test*. It demonstrates the storage of meshes and retrieving whole meshes or individual vertices from a mesh. Meshes are returned from the SRA as ASSIMP Meshes, thus requiring the ASSIMP Library.

Load and Write Maps Loading and writing maps, like images or depth maps, is demonstrated in this test. It is contained in *PersistenceAPI-Test* and showcases storage of whole maps, their retrieval and the access of individual channels or pixels. Image access requires the OpenCV library.

Load and Write File Paths Writing file paths to the SRA is a common way to include objects into a scene, without restructuring a file to the SRA. A demonstrating example is included in *PersistenceAPI-Test*, which shows how file paths can be inserted into and read from a scene.

Load and Write Binary Data Speed, size and portability are the major reasons for including binary data into a scene. Such data is not further structured, which means that if an image is stored as binary data individual pixels cannot be accessed, or if a mesh is stored as binary data its vertices cannot be referenced individually. One advantage of binary data is that it allows compression of contents with current data compression schemes, which is otherwise difficult to achieve in the SRA. A second advantage is that binary data provides an easy option to include novel and uncommen data types into a scene representation without the need to implement an addition to the SRA.

An example showcasing the insertion and extraction of binary data to and from a scene is shown in *PersistenceAPI-Test* as well.

Visual Validation Scenes can be visualized by different tools implemented in the SCENE project. Chapter 4.2 presents approaches how content of the SRA files can be visually validated using realistic rendering approaches.

Frontend Usage Example

The following example illustrated the use of the SRA front-end. In this example a user adds several meshes contained in an *obj*-file to a scene. This requires the ASSIMP library, and the user needs to add the SRA front-end and filetype specification:

Listing 2.1: Including SRA Front-End Headers

```
#include <scene-persistence/sra.h>
#include <scene-persistence/backend/filetypes.h>
```

The user can than create a scene or load an existing scene file. This is done automatically, depending on whether the specified scene name already exists or not:

Listing 2.2: Initializing Scene in SRA

```
string sceneName = ``myScene.sra'';
Scene myScene(sceneName);
```

All new scene objects can only be added to a parent acel. This ensures a valid tree structure and no acels which are orphaned in a scene. If no parent acel is specified or the specified acel does not exist, the new content is automatically added to the top level of the current scene. The code in Listing 2.3 adds the meshes stored in file *test.obj* to the parent acel with ID 10. The file type needs to be specified with the input, where filetypes are defined in the *filetypes.h* file included above.

The front-end function *addMesh* returns an integer number specifying the number of meshes contained in the input file. Additionally, a pointer to a list of IDs is returned, allowing the user to access the newly added meshes in the scene structure. This process of adding a mesh to a scene is given here:

Listing 2.3: Adding Meshes to the SRA

```
string meshName = ``test.obj'';
int parentID = 10;
```

```
int numOfMeshes = myScene.addMesh(meshName, meshType::OBJ,
    parentID, &ids);
```

The mesh, all faces and vertices, normal vectors and texture coordinates from the *obj* file are afterwards stored as individually structured acels in the scene file and can be individually accessed and processed. Meshes as a whole are returned as *aiMeshes*, the native structure of the ASSIMP library. To retrieve an *aiMesh* from a scene the ID of the mesh in the scene file needs to be passed to a function asking for the mesh. The following query illustrates such a request, returning the first of the meshes added to the scene file:

Listing 2.4: Retrieving a Mesh from the SRA

```
aiMesh *myMesh = myScene.getMesh(ids[0]);
```

While the SRA presented here is a suitable format for captured and computer generated data, further processing options severely depend on the knowledge about content in the scene. For illustration purpose consider a video, and three different approaches to store the video. First, a video can be stored in total as a binary bunch of data in a single acel. The SRA only allows access of the video as a whole, with very limited flexibility. Second, a video can be stored as a sequence of frame acels, each frame being persisted as binary data. In this scenario, the SRA enables access of individual frames, but smaller segments are not possible. It is, however, also possible to persist each frame as a matrix of individual color values, allowing individual access and processing of the smallest elements in video data.

More interesting than the individual pixel values are often semantically connected objects in a scene. What is inherently available in computer generated content - spatial and temporal knowledge about objects - needs to be algorithmically extracted from captured content. A core objective for the goal to create content for the SRA which allows seamless integration of captured and generated content is therefore the task of segmenting content into spatially and temporally consistent objects.

2.2 Adding Semantics to Scenes

While the SRA provides a necessary environment to combine computer generated and real captured data, to enable the same flexibility on captured data as inherently available for generated data still a considerable amount of preprocessing is required. This preprocessing is mostly concerned with adding semantics to content. While the term “semantics” is often used as a meaning of content that makes sense for human observers, here semantics is considered

less strict. In the context of adding flexibility to captured data it does not matter whether for example a car has the semantic meaning of a car attached to it, or whether it is just an object with a unique identifier. More important than the name is that the semantics remain consistent, spatially as well as temporally. Captured scene elements belonging to the same object need to be assigned to this object at any time and at any spatial extension. In this work semantics are understood as exactly this characteristic of being attributed to a spatially and temporally consistent object.

2.2.1 Spatially and Temporally Consistent Acels

Image segmentation is a fundamental preprocessing step for many computer vision related problems. Next to the afore mentioned problem of merging real and synthetic data the task of object recognition, object detection and object tracking pose challenging problems. Gonzalez et al. introduce image segmentation as one of the most difficult problems in digital image analysis, as success or failure of many succeeding processing steps depend on a correct segmentation. According to Gonzalez et al. image segmentation is primarily based on two basic characteristics of images, which are similarities and discontinuities [31]. Image segmentation tries to combine pixels with similar attributes to the same object and differentiates between objects at attribute discontinuities. Multiple algorithms for image segmentation are described in the literature, from simple thresholding techniques from the 1980s [23] over region growing methods developed in the 1990s [113, 98] to more sophisticated watershed segmentation approaches, for which significant advances were still made in the 2000s [90, 68].

Common to all segmentation approaches is the goal to achieve a segmentation which comes closest to a ground truth segmentation, a segmentation that humans would apply to the scene. Three measures are central to the evaluation of the quality of an algorithmic segmentation compared to a ground truth. These measures are boundary recall, oversegmentation and undersegmentation. In [P4] we define these measures as follows.

Boundary recall is commonly understood as the fraction between the number of boundary pixels that are correctly found by a segmentation approach and the total number of boundary pixels of a ground truth. Boundary accuracy can be specified less strictly if it suffices that a segmentation boundary pixel falls within a given region of the ground truth boundary. For a segmentation S and a ground truth T let $bp(S \wedge T)$ denote those boundary pixels which are overlapping in segmentation and ground truth, $bp(T \veebar S)$ denote the boundary pixels exclusively available in the ground truth and $bp(S \veebar T)$ analogously denote the boundary pixels exclusively found by the segmentation.

Let $bp(T)$ denote all boundary pixels of the ground truth and $bp(S)$ denote all the boundary pixels of the segmentation. Then boundary recall $br(S,T)$ of segmentation S for ground truth T in percent is calculated as

$$\begin{aligned} br(S,T) &= \frac{bp(S \wedge T)}{bp(T)} \cdot 100\% \\ &= \frac{bp(S \wedge T)}{bp(S \wedge T) + bp(T \veebar S)} \cdot 100\% \\ &= \frac{bp(S) - bp(S \veebar T)}{bp(T)} \cdot 100\% \end{aligned} \tag{2.3}$$

Oversegmentation is a measure indicating how many additional segments a segmentation algorithm creates to represent the segments of a ground truth. Let t be the number of segments of ground truth T, and $or(S \wedge T_i)$ be the number of segments of S overlapping with segment T_i for $1 \leq i \leq t$. The oversegmentation factor $os(S,T)$ for segmentation S and ground truth T is then computed as

$$os(S,T) = \frac{1}{t} \sum_{i=1}^{t} or(S \wedge T_i) \tag{2.4}$$

The undersegmentation measure is - analogue to the oversegmentation measure - an indicator how many segments a segmentation approach fails to produce compared to the ground truth. With the same nomenclature as used for oversegmentation and $or(T \wedge S_i)$ as the number of segments of T overlapping with segment S_i for $1 \leq i \leq s$ where s is the number of segments of segmentation S the undersegmentation factor $us(S,T)$ for segmentation S and ground truth T is given by

$$us(S,T) = \frac{1}{s} \sum_{i=1}^{s} or(T \wedge S_i) \tag{2.5}$$

Apart from these measures which compare an algorithmic segmentation to a ground truth there are two further metrics specifying the performance of a segmentation. These two further measures are compactness of segments and complexity of the segmentation algorithm. Especially for segmentations which oversegment objects a high level of compactness is desireable. A measure of compactness is introduced by Schick et al. who compare the individual segments to circles. If a segment has the same area-to-perimeter ratio (called isoperimetrical coefficient) as a circle it is said to have maximal compactness

[88]. For s segments of the segmentation the isoperimetrical coefficient for the i-th segment with $1 \leq i \leq s$ of segmentation S is defined as

$$Q_i = \frac{4\pi A_i}{L_i^2} \tag{2.6}$$

where A_i is the area and L_i is the perimeter of segment S_i. The average of all isoperimetrical coefficients for the segments of a segmentation S gives the compactness factor $co(S)$ for this segmentation:

$$co(S) = \frac{1}{s} \cdot \sum_{i=1}^{s} Q_i \tag{2.7}$$

Complexity, as mentioned above, is another important characteristic of a segmentation approach. A crude segmentation approach groups pixels by their similarity (or difference), which requires pairwise pixel comparisons. More sophisticated segmentation approaches may limit the searchspace for comparisons or compare items only to predefined seeds. For a complexity estimation the complexity of an approach comparing each pair of pixels is considered. The total number of comparisons is given by possible combinations of permutations. The general equation of combinations of permutations (regardless of order and without repetition) where n is the number of possible items and r is the set of items in one comparison simplifies with $r = 2$, giving a total number cp of required comparisons for segmentation S

$$cp(S) = \frac{n!}{(n-r)!(r!)} = \frac{1}{2} \cdot (n \cdot (n-1)) \tag{2.8}$$

To compare all pixels of an image with each other therefore the number of comparisons increases quadratically with the number of pixels in an image. As the complexity of a full comparison increases quadratically with the number of pixels, the complexity of a segmentation approach becomes critical especially when segmentations shall be used in real-time environments.

Optimization of a segmentation approach is a multidimensional problem, since a segmentation shall in general be optimized with respect to all, boundary recall, oversegmentation, undersegmentation, compactness of segments and complexity of the algorithm. At the same time many of the quality measures of a segmentation are related. The term relation is necessarily weak as segmentation quality measures depend on image content and cannot be determined generally. However, for many of the quality measures it can be stated that the larger one measure, the larger or respectively smaller another measure. In the following the $\sim$-sign denotes a relation of stated kind, where $x \sim y$ emplies

that the larger x, the larger y, and $x \sim \frac{1}{y}$ denotes relations where the larger x, the smaller y.

- $br(S,T) \sim os(S,T)$: Boundary recall is directly related to oversegmentation. By increasing the oversegmentation the propability increases that one of the additional boundaries of the new segments corresponds to a ground truth boundary. In the extreme case every pixel of a segmentation can be an individual segment, thus creating a perfect boundary recall at the cost of the highest possible oversegmentation. On the other hand, by using only a single segment as the whole segmentation, the lowest possible oversegmentation is achieved at the cost of a low boundary recall. Boundary recall and oversegmentation need to be optimized together with the goal to achieve the highest boundary recall ($br(S,T) = 1$) for the lowest possible oversegmentation factor ($os(S,T) = 1$).

- $br(S,T) \sim \frac{1}{us(S,T)}$: Boundary recall and undersegmentation are inversely related. A large undersegmentation factor means that many segments spill across ground truth boundaries, thus a large undersegmentation leads to low boundary recall. Both measures can be optimized together by simply assigning each pixel of a segmentation to an individual segment ($br(S,T) = 1$ and $us(S,T) = 1$). This crude approach, however, adds little value. A key concern therefore is to increase the segment size while maintaining a high level of boundary recall and a small undersegmentation factor.

- $br(S,T) \sim \frac{1}{co(S)}$: Perfect compactness can only be achieved if all segments of a segmentation correspond to circles. Since average objects in scenes differ from circles, boundary recall and compactness are inversely related. In the extreme case of perfect compactness boundary recall will tend towards zero if segment sizes are lare. Therefore a goal for segmentation approaches is to optimize both; find the best (highest) compactness for the highest possible boundary recall. Once again optimization needs to be considered on multiple dimensions, as a trivial optimal case for both compactness and boundary recall is a segmentation which considers each pixel as an individual segment.

- $br(S,T) \sim cp(S)$: Crude approaches with low complexity usually create segmentations with little boundary recall. For example, a segmentation approach which would split an image into uniform rectangles runs in constant runtime, but has little boundary recall. Increasing boundary recall comes at the cost of increasing complexity, thus optimizing segmentation

approaches strives for the best achievable boundary recall ($br(S,T) = 1$) at the lowest possible complexity.

- $os(S,T) \sim \frac{1}{us(S,T)}$: Oversegmentation and undersegmentation are by definition inversely related. Optimizing one of them individually is trivial. For all pixels as individual segments undersegmentation is optimal ($us(S,T) = 1$) and for the whole segmentation being a single segment oversegmentation is optimal ($os(S,T) = 1$). The optimum of both, undersegmentation and oversegmentation ($us(S,T) = us(S,T) = 1$), is only achieved for a segmentation perfectly matching the ground truth.

- $os(S,T) \sim co(S)$: The smaller the segment sizes the more compact they become. With the extreme of a single pixel being perfectly compact, a large oversegmentation is directly related to a high compactness value. Achieving a high compactness for a low oversegmentation is the goal for an optimized segmentation approach.

- $os(S,T) \sim \frac{1}{cp(S)}$: Complexity of a segmentation approach can be reduced by reducing the search space for comparisons - an idea which is employed by SLIC superpixel segmentation (see following Section 2.2.1). Reduced search space directly leads to smaller segments, as segments can only be created in the reduced search space. Therefore, the smaller the complexity, the potentially larger is the oversegmentation. Optimizations try to decrease the complexity of an algorithm while maintaining the oversegmentation, or decreasing oversegmentation for constant complexity of a segmentation approach.

- $us(S,T) \sim \frac{1}{co(S)}$: Similar to the previous comparison between oversegmentation and compactness, undersegmentation is inversely related to compactness. The reasoning remains the same: a larger number of segments tends to be more compact, and a larger number of segments decreases the oversegmentation. Optimization approaches try to achieve a high compactness for a low undersegmentation, which can best be achieved if segments boundaries conform to ground truth boundaries.

- $us(S,T) \sim cp(S)$: With the same argumentation as employed for the relation between oversegmentation and complexity it holds that undersegmentation is directly related to complexity. Undersegmentation corresponds to larger segment sizes, which in turn require larger search spaces for the segmentation. This increases the complexity of segmentation approaches.

Summarizing this comparison of different segmentation quality assessment methods it is worth to consider a segmentation which exactly matches the ground truth. For such a perfect segmentation the achieved boundary recall is $br(S,T) = 100\%$ and oversegmentation as well as undersegmentation are both perfect, $os(S,T) = us(S,T) = 1$. The compactness measure, however, is different from the perfect compactness of circle-shaped segments, and complexity is necessarily larger than for algorithms which restrict a segment size to a fraction of a ground truth segment. For the algorithmic solutions presented in the following compactness and complexity therefore present important factors for algorithmic decisions, but subordinate importance for the quality evaluation of the segmentation results.

Superpixel Segmentation

Combining neighboring pixels with similar attributes into groups of pixels is known as superpixel segmentation, and each of the groups of pixels is denoted as a superpixel. Known superpixel clustering approaches can be sorted into two groups, graph-based and gradient-ascent-based algorithms [2]. Graph-based approaches consider all pixels as nodes of a graph and assign different kinds of similarity measures as weights to the edges connecting said nodes. Gradient-ascent-based algorithms start from some initial distribution of pixels to clusters and iteratively enhance clusters until a given convergence criterion is met.

Of the current superpixel algorithms SLIC (abbreviating Simple Linear Iterative Clustering) superpixels define the state-of-the-art with respect to algorithm complexity, boundary recall and segment compactness [2]. As discussed above performance measures of segmentation approaches are interrelated. For SLIC superpixels the decreased algorithm complexity and enhanced boundary recall performance come at the cost of oversegmentation. By limiting the search space for pixels corresponding to one superpixel on a predefined grid, the superpixel size is limited to the size of one grid cell but at the same time the number of necessary comparisons between pixels is drastically reduced.

Several different implementations of the SLIC algorithm are proposed. The initial SLIC algorithm places cluster centers and iteratively refines the cluster points belonging to this center, thus belongs to the gradient-ascent-based algorithms. A derivation of this approach clusters pixels based on their geodesic distance to a cluster center, which is a graph-based approach of the clustering problem.

The SLIC algorithm introduced by Achanta et al. [2] is simple to implement and use, which makes it - combined with the characteristics of low complexity and high boundary recall - a perfect basis for further research on image seg-

mentation. The algorithm requires the rough number of desired superpixels as well as a factor influencing the compactness of the superpixels as user input. A number of cluster centers corresponding to the user-desired number of superpixels is placed in regular intervals over the area to be segmented.

In the next step, each pixel is assigned to the cluster which is closest to it with respect to a metric defined over both, color and space. Here, the search space is limited to grid cells which are derived from the desired number of superpixels, thus significantly reducing the number of comparisons and - corresponding to the comparisons - reducing the complexity of the algorithm compared to a full search over the whole image.

After all pixel correspondences have been calculated, the cluster centers are updated by the mean value of all pixels belonging to this cluster. These two steps of pixel assignment and center calculation are iteratively repeated until a convergence criterion is met. This algorithm is given as MATLAB code in Listings 2.5 to 2.9. The algorithm inputs are the image to be segmented, the desired number of superpixels and the factor influencing the compactness of the segments.

Listing 2.5: SLIC Superpixel Clustering - Algorithm Inputs

```
img;                          % the image to be segmented
k;                            % desired number of superpixels
l;                            % factor influencing compactness of
    segments
```

The initial centers are positioned according to the user-desired number of superpixels on a regularly spaced grid.

Listing 2.6: SLIC Superpixel Clustering - Initial Center Positioning

```
N=row*col;                    % number of image pixels
S=sqrt(N/k);                  % nominal spacing between grid
   elements
labels = -ones(row,col); % initialise pixel labels to -1
distances = inf(row,col);% initialise pixel distances to
   infinity

% position initial clusters
centers = double(image(round(S/2:S:r),round(S/2:S:c),:));
```

In several iterations, until the convergence criterion is met, a grid cell around each superpixel center is extracted, and the spatial and color distance for all pixels is calculated. Color and spatial distance are combined to a single distance using the user-given input factor. The higher the weight assigned

to the spatial distance, the more compact segments become. The higher the weight assigned to the color distance, the less compact the resulting segments.

Listing 2.7: SLIC Superpixel Clustering - Distance Calculation

```
while(true)     % iterate until convergance criterion is met
    for c = 1:k % iterate over all cluster centers

        % consider only grid cell around center
        left = max(round(centre(c,4)-S), 1);
        right = min(round(centre(c,4)+S), c);
        top = max(round(centre(c,5)-S), 1);
        bottom = min(round(centre(c,5)+S), r);
        subimg = double(image(top:bottom,left:right,:));

        % calculate spatial distance
        x = double (subimg(:,:,4)-centre(c,4));
        y = double (subimg(:,:,5)-centre(c,5));
        ds = double (x.^2 + y.^2);

        % calculate color distance
        for n = 1:3
          subimg(:,:,n)= (double(subimg(:,:,n))-double(
             centre(c,n))).^2;
        end
        dc = sum(double(subimg(:,:,1:3)),3);

        % combine spatial and color distance
        D = double (sqrt (dc + ds.*(1/S)^2));
```

Knowing the distance between a cluster center and all pixels in a grid cell, those pixels closest to the center can be assigned to the center.

Listing 2.8: SLIC Superpixel Clustering - Pixel Assignment

```
        % assigning pixels to closest center and update
           distances
        distance = dist(top:bottom,left:right);
        label = labels(top:bottom,left:right);
        refresh = D < distance;
        distance(refresh) = D(refresh);
        label(refresh) = c;
        distances(top:bottom,left:right) = distance;
        labels(top:bottom,left:right) = label;
    end
```

Finally the cluster centers are updated with the mean values of all pixels assigned to the cluster. These steps are iteratively repeated until a convergence criterion is met.

Listing 2.9: SLIC Superpixel Clustering - Update Cluster Centers

```
    % update cluster centers with mean values
    lab = cat(3,labels,labels,labels,labels,labels);
    for c = 1:k
        centers(c,:)= mean(reshape(img(lab==c),[length(img(
            lab==c))/5,5]));
    end
end
```

Figure 2.4 shows a segmentation created with the SLIC algorithm described above and different parameter choices. In the first row a medium compactness factor l is chosen and two extreme values for the desired number of superpixels k are used, in the second row a medium number of superpixels k is used to display the effect of extreme choices of compactness factor l. Figures 2.4a and 2.4b show segmentations for numbers of desired superpixels, $k = 100$ and $k = 400$ respectively, while the weighting between spatial and color distance remains constant at $l = 10$. In Figures 2.4c and 2.4d the number of superpixels is kept constant at $k = 200$, and the weighting factor is chosen as $l = 1$ for color weight and $l = 50$ for a focus on spatial weight. The reference image stems from the Kodak Lossless True Color Image Suite [22], which is fully introduced in Chapter 5.1.1.

Without a full analysis several observations about the performance of the segmentation can be made. Undersegmentation and loss of boundary recall for fine image structures are directly related, see Figure 2.4a. A higher number of superpixels can recall finer image structures leading to a better boundary recall, at the cost of oversegmentation, visible in Figure 2.4b. An increase in compactness yields more appealing segments, but also comes with a loss of recall for fine image features (compare Figure 2.4c and Figure 2.4d). A comparison of SLIC segmentation to other segmentation approaches confirming SLIC as an approach which defines the state-of-the-art with respect to both, boundary recall and complexity, is given by Achanta et al. in [2].

Multi-Dimensional Segmentation

Traditional approaches to image segmentation employ the three channels of color information in any desired and suitable color space. Video segmentation can additionally consider time as a parameter to create spatially and temporally consistent segments. A segmentation approach which has extended SLIC superpixels to the temporal domain is presented by Reso et al. in [81].

Current approaches for data acquisition, however, go beyond color and time information. The acquisition hardware introduced in Chapter 1.2 captures information exceeding color per frame, such as depth information and infrared

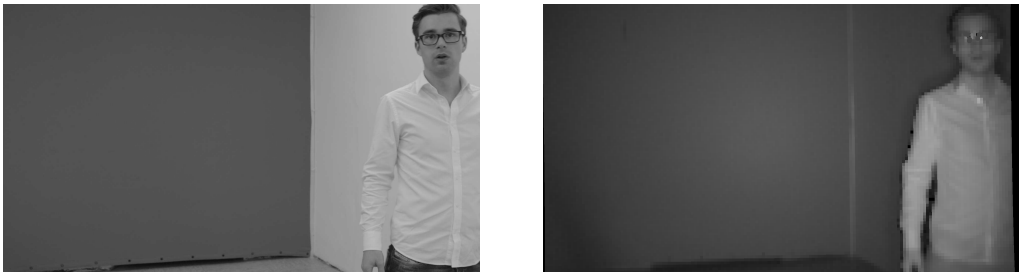
(a) SLIC Segmentation with $k = 100$ and $l = 10$

(b) SLIC Segmentation with $k = 400$ and $l = 10$

(c) SLIC Segmentation with $k = 200$ and $l = 1$

(d) SLIC Segmentation with $k = 200$ and $l = 50$

Figure 2.4: SLIC Segmentations for Different Numbers of Superpixels k and Changing Weights Between Spatial and Color Distance l



(a) Color Information in front of White Background

(b) Infrared Information in front of White Background

Figure 2.5: Subjective Motivation to Employ Infrared Information for Image Segmentation

radiation. In [P4] the idea to use such additional information channels for enhanced segmentation results is explored.

Due to its low complexity, good boundary recall and ease of implementation and use the SLIC superpixel algorithm introduced in the previous chapter is employed as a basis for research on multidimensional segmentation approaches. First, in [P4] we analyze whether additional channels also contain additional information which can be used for image segmentation, and how such information can be combined with the already employed color and spatial information. Subjectively, the quest for additional information is answered exemplary by the images shown in Figure 2.5 (see also Figure 2.7): the infrared channel adds considerably stronger contours to the talent then the color image, as the talent wears a white shirt in front of a white wall.

The standard SLIC implementation employs the Euclidean distance measure for color and spatial distance measurement, and combines both distances in a common distance with a weighting factor m. For three color channels c_1, c_2 and c_3 the color distance d_c is calculated as

$$d_c = \sqrt{(\Delta c_1)^2 + (\Delta c_2)^2 + (\Delta c_3)^2} \tag{2.9}$$

where Δc_i denotes the distance to the segment center for color channel $1 \leq i \leq 3$. Analog to the color distance the spatial distance d_s is defined as the Euclidean distance between spatial x and y dimensions

$$d_s = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{2.10}$$

As further information channels contribute with information similar to color information the nomenclature is adjusted with respect to Listing 2.6. We define $m = \frac{S^2}{l^2}$ to replace the weighting factor used by Achanta et al. [2],

in Listing 2.6 and in the previous examples. Through this redefinition the input weighting factor becomes a prefactor of the color distance (instead of the spatial distance) and the spatial normalization factor S is combined into one single factor. With this redefined m the two distance measures are combined to distance d as

$$d = \sqrt{m \cdot d_c^2 + d_s^2} \tag{2.11}$$

The weighting factor m remains as a user input and defines the weighting between focus on color proximity and local distance.

Further information channels can be introduced to the combined distance measure by calculating the individual distances on the channel, and merging those individual distances with further weighting factors into the combined distance measure. Exemplary, based on the acquisition hardware introduced in Chapter 1.2, the distances for infrared information d_i and depth information d_d based in infrared channel ir and depth channel de respectively are given as

$$\begin{aligned} d_i &= \sqrt{(\Delta ir)^2} \\ d_d &= \sqrt{(\Delta de)^2} \end{aligned} \tag{2.12}$$

Consequently, those channels are combined in the combined distance measure with additional weighting parameters n and o. As infrared and depth information may contain edge information of the scene, these three kinds of information are inherently different from the spatial information which is purely used for the compactness and spatial delimiter of the segments. Therefore we propose in [P4] to keep the spatial distance constant and weight the other channels to achieve optimal segmentation results. The combined distance measure enhanced by infrared and depth information d_e then becomes

$$d_e = \sqrt{d_s^2 + m \cdot d_c^2 + n \cdot d_i^2 + o \cdot d_d^2} \tag{2.13}$$

While a single parameter can be fixed in advance for satisfying segmentation results and the difficulty of setting a single parameter is low, this is not feasible for a growing number of parameters any more. With the introduction of further information channels therefore a dynamic weighting factor allocation is necessary.

The dynamic distribution of weighting parameters follows two core design criteria which are as follows.

1. Weights should be allocated to channels contributing edge information to a segmentation. The more object boundaries an information channel contains, the higher the weight allocated to this channel.

2. The relation between spatial information and all other information channels needs to be constant. This constant relation assures segments of a desired compactness.

Mathematically, the second design criterion is given by $m+n+o = c$, where c is a constant value corresponding to the weighting factor which is considered as an adequate weight between contours and spatial proximity in the literature [2].

The first design criterion is more difficult to specify. Allocation of weights should consider the amount of object boundary information a channel can contribute in the local environment of the segment, it needs to consider the dynamic range of the information and the dimensions contributing to the given information.

For the number of edges in a region the same grid cell around a superpixel center is considered that also represents the search space for pixel proximity. In this area edges are detected using the 2D Laplacian which is given by the second derivative in x- and y-direction of an image

$$\Delta^2 f(x,y) = 4 \cdot f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1) \quad (2.14)$$

which can be efficiently computed as the convolution $edges = image * k$ with convolution kernel k, where k is

$$k = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (2.15)$$

This operation leads to a matrix per channel indicating the number of edges found. These edge matrices can be converted to binary using a predefined threshold. In practice a threshold of $T = 0.05$ was found to yield good results for a channel with a dynamic range of 1. If several channels contribute to one kind of information (e.g. for color information) the combined edge information is 1 everywhere where at least one of the channels is 1, thus an OR combination of all channels contributing to the same information. By counting the edge information in the grid cell corresponding to the superpixel, knowledge about the distribution of contours over the different kinds of information is gained.

After this process, weighting factors m''', n''' and o''' are obtained for the color, infrared and depth information. These values are directly counted from the edge pixels found by 2D Laplacian filtering of the information channels. In a next step, the weighting factors are mapped to the weighting factors m'', n'' and o''. Weighting factors m'', n'' and o'' are not adjusted to the dynamic range

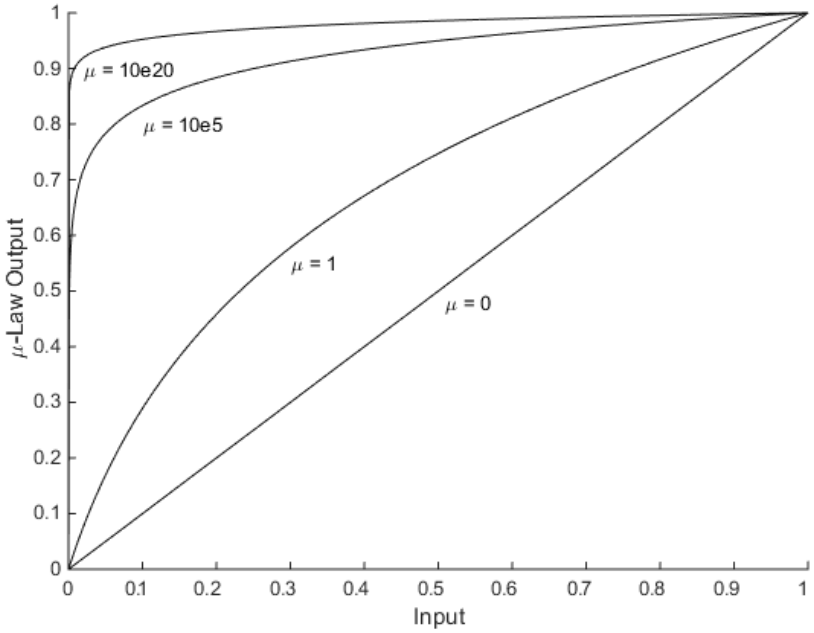


Figure 2.6: Different Parametrizations of the μ-Law Compressor

or number of dimensions, but represent a subjectively pleasing relation to the final weighting factors. The mapping function $g : x''' \mapsto x''$ for $x \in m, n, o$ was found by subjective tests.

The μ-law, which was originally designed for the digitization of audio signals, is used here as a basis function for subjective tests, as it can be parametrized to cover mappings from linear to constant assignments of function values. Figure 2.6 shows a set of functions for different parameters μ. Multiple segmentations created with different mapping functions were presented to a group of subjects. Out of 10 segmentations, each subject had to choose the segmentation most appealing to him or her. Results from all evaluations were averaged, leading to the following parametrization of the μ-function

$$g(x) = sgn(x) \cdot \frac{ln\,(1 + \mu\,|x|)}{ln\,(1 + \mu)} \tag{2.16}$$

with $\mu = \frac{2200}{S^2}$ where S corresponds to the grid cell size as introduced in Listing 2.6.

In the next step the weights m'', n'' and o'' are normalized by the dynamic range of the channel, an adjustment for multiple dimensions is made and all parameters are adjusted to sum up to the desired constant value c. The following scenario illustrates the importance of these normalization steps. Given

two information channels of which one is given in the range between 0 and 255 (representation with unsigned 8bit values), and the other one is represented with images from 0 to 1 (representation with double values). A step from black to white would result in a Δ of 255 for the first channel, but only in a Δ of 1 for the second channel.

Considering further more two information sources, one providing three channels, the other one only one. If both information sources have a dynamic range from 0 to 1. Looking again at the difference from black to white, the three-channel information presents a $\Delta = \sqrt{(1^2 + 1^2 + 1^2)} = \sqrt{3}$, while the single channel information has $\Delta = 1$. For a minimum channel value min_x and a maximum channel value max_x, and z_x dimensions per information the mapping $h : x'' \mapsto x'$ for $x \in m, n, o$ is given by

$$h(x) = \frac{x}{\sqrt{z_x}\,(max_x - min_x)} \tag{2.17}$$

Finally the weights are adjusted to sum up to a constant value c, motivated by the second design criterion outlined above. This mapping $i : x' \mapsto x$ for $x \in m, n, o$ is

$$i(x) = \frac{x \cdot c}{\sum_{x \in m,n,o} x'} \tag{2.18}$$

Equations 2.14 to 2.18 lead to a fully dynamic adjustment of weights among the different channels, reducing the required user input to the same number of parameters as required for the standard SLIC algorithm. This approach was tested on a dataset acquired with the Motion Scene Camera (see Chapter 1.2). The dataset explicitly shows a scenario which is difficult to segment correctly with a purely color based segmentation approach: a talent with a white shirt is moving from a dark background to a white background. Figure 2.7 gives frames of this dataset for the talent standing in front of the dark background and in front of the white background. Along with the color information depth and infrared frames as well as ground truth segmentations are shown. The ground truth segmentations were created manually and show 9 different segments, which are the dark background, white background, floor, hair, face, hands, shirt and pants of the talent.

In experiments on the dataset shown in Figure 2.7 with the original SLIC segmentation and the multichannel enhanced algorithm two parameters, the desired number of superpixels k and the relation between spatial proximity and contours, originally denoted by m, can be adjusted. Adjusting the desired number of superpixels directly influences the oversegmentation: the more superpixels are desired, the higher the oversegmentation factor. By adjusting the weighting between spatial proximity and edge adherence the compactness

(a) Color Information in front of Dark Background

(b) Color Information in front of White Background

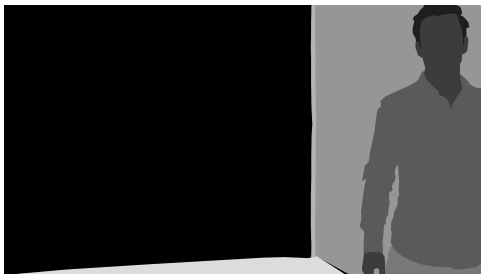
(c) Infrared Information in front of Dark Background

(d) Infrared Information in front of White Background

(e) Depth Information in front of Dark Background

(f) Depth Information in front of White Background

(g) Ground Truth Segmentation in front of Dark Background



(h) Ground Truth Segmentation in front of White Background

Figure 2.7: Dataset showing Talent in front of Dark and White Background, consisting of Color, Infrared, Depth and Ground Truth Segmentation

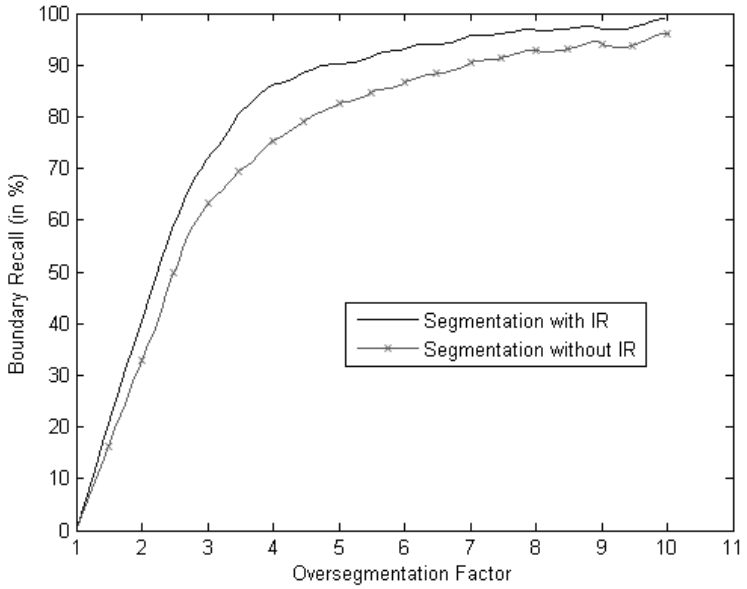


Figure 2.8: Comparison of Oversegmentation to Boundary Recall for Segmentation Based only on Color Information and Segmentation Based on Color and Infrared Information

of superpixels is influenced: the higher spatial proximity is weighted, the more compact superpixels become.

Figure 2.8 shows the boundary recall values achieved for different oversegmentation factors generated by adjusting the number of desired superpixels, for segmentation approaches using only color and color as well as infrared information. The general relation for both implementations between boundary recall and oversegmentation confirms the observation stated above, that a higher number of superpixels, thus a higher oversegmentation factor, is directly related to a higher boundary recall. It is clearly visible that the segmentation using both, infrared and color, exceeds the boundary recall of only color segmentation independent of the oversegmentation factor. The largest benefit of using the additional infrared channel is achieved for an oversegmentation factor of 4.3 with an enhanced boundary recall of over 11%. While this exact oversegmentation factor achieves the largest benefit for boundary recall, it does by no means indicate an optimal oversegmentation factor: finding an optimal position requires a multidimensional optimization also considering undersegmentation, compactness and complexity of the algorithm.

Adjusting the weighting factor between spatial proximity and edge adherence changes the compactness of the superpixel. Figure 2.9 gives the bound-

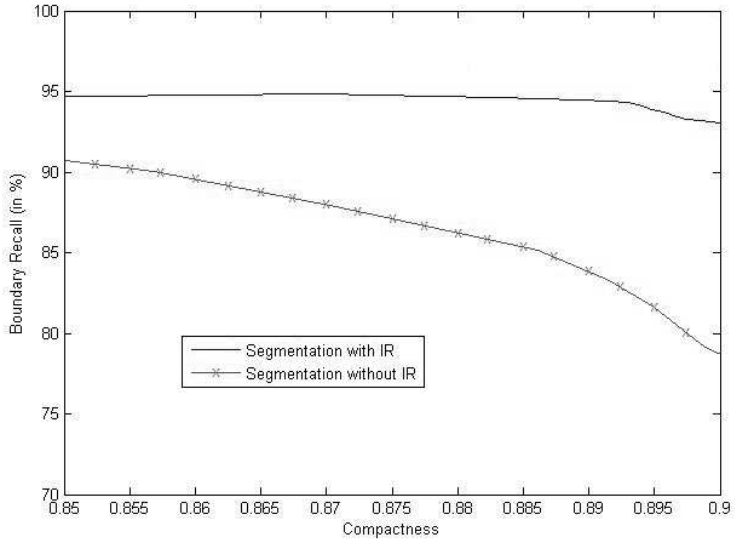


Figure 2.9: Comparison of Compactness to Boundary Recall for Segmentation Based only on Color Information and Segmentation Based on Color and Infrared Information

ary recall values achieved for different segment compactness values, where the compactness is set by the weighting factor. Again, the plot compares a segmentation using only color information to a segmentation employing color and infrared information. The inverse relation discussed above is visible here as well: with increasing compactness the boundary recall of the segmentation decreases, as spatial proximity is rated higher than conforming to fine image structures. The segmentation approach using the additional infrared information achieves higher boundary recall values than the segmentation without infrared information for arbitrary compactness values. For compactness levels approaching 0.9 the boundary recall improvement of additional infrared information is exceeding 10%. Again, this better performance does not indicate an optimal segmentation parameter, as for a subjectively appealing segmentation usually compactness is rated less important than boundary recall.

Table 2.1 compares the boundary recall which is achieved for a segmentation into 200 superpixels of the dataset given in Figure 2.7. The depth information adds additional object edge information to the segmentation, and both infrared as well as depth information maintain the object boundaries independent of the background color. Thus the method presented in [P4] contributes to object boundary detection and semantic segmentation using multi-channel information. While the current implementation only considers an enhancement

Table 2.1: Boundary Recall of Superpixel Segmentations with 200 superpixels using only Color Information, Color and Infrared Information and Color, Infrared and Depth Information of the Talent standing in front of the Dark and White Background

Background	Color Information	Color and Infrared Information	Color, Infrared and Depth Information
Dark Background	95%	95%	96%
White Background	89%	95%	96%

of the SLIC superpixel algorithm, all other graph-based metrics which employ a distance measure for edge weighting can be extended with the presented approach to multi-dimensional segmentation.

In the SRA superpixel labels have been implemented as an additional acel corresponding to each image acel. With multiple image acels and superpixel label acels being concatenated in a video stream superpixels can be individually accessed, tracked and modified.

Temporally Consistent 3D Acels

Spatial and temporal consistency is not limited to 2D space. Several approaches have been taken to create temporally consistent 3D meshes, which for example allow temporally correct texture mapping to a mesh. One approach developed by Klaudiny and Hilton uses the SRA for persistence of temporally consistent mesh structures. [43] Klaudiny and Hilton track surface patches from calibrated camera sources to determine the temporal evolution of patches. The SRA presents the first architecture which can persist information gained by algorithms detecting temporal consistency of 3D meshes.

Temporal consistency between different time frames is created in two consecutive steps. First, the texture from a proceeding frame is mapped to the texture of the current frame. Using cross-correlation texture patches are compared to find the best fit for a texture in a series of frames. Cross-correlation defines a distance metric for shifted versions of image patches, based on pixel intensities. This illustrates that pixel intensities and differences between pixels play an important role not only in 2D image segmentation but also for the creation of 3D consistent content. While integrating multi-dimensional information coming from additional infrared and depth sensing devices into the cross-correlation computation is straightforward with the approach introduced in the previous chapter, the value of having additional channel information for 3D surface patch matching has not been explored so far.

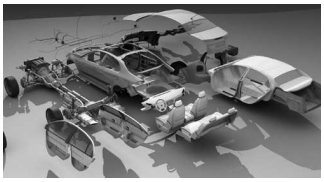
Chapter 3

Data Reduction

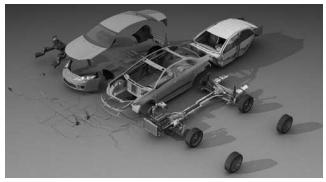
In the previous chapter it was observed that real and synthetic data are inherently different with respect to the semantic structure of the data. For real data a sensor usually observes a whole scene and captures as much of the available information as possible. Further processing can segment scene content and assign semantic meanings to individual parts or objects of captured content. The previous chapter has introduced approached to enhance the information of captured content. On the opposite, this chapter concentrates on information coming from the synthetic world, which - by its nature - is exactly the other way around as it exists inherently. When synthetic data is generated, design artists start with individual, small parts and combine these parts into larger models or scenes. Figure 3.1 exemplarily illustrates the different components of a computer modeled car[1]. By design this model already contains semantic knowledge about tires, engine, chassis, and all the other parts which are modeled individually. Furthermore, combinations of parts, e.g. the combination of all parts into a car, is again linked to the semantic knowledge about an object.

Introducing models, like the car model shown in Figure 3.1, in the context of data decrease might at first glance seem counterintuitive. File sizes for the given model range from $15MB$ (Maxon Cinema 4D Studio - file) to $105MB$ (Maya - file). When compared to the size of a JPEG coded image of the car (e.g. Figure 3.1e) at $30kB$ data decrease indeed seems absurd. However, certain scenarios enable the use of computer generated models for more efficient data compression. One option is to compress or simplify the model in order to reduce its storage requirements. A second option is to reuse the available model sufficiently often to amortize the additional storage requirements of a full model.

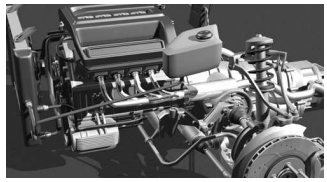
[1]Sebastian Dosch. Dosch 3D: Car Details V2. `http://www.doschdesign.com/products/3d/Car_Details_V2.html`. Accessed 20-Apr-2016



(a) Parts of a Car Model (1)

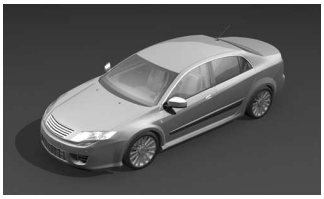


(b) Parts of a Car Model (2)



(c) Motor Block of a Car Model

(d) Car with transparent chassis



(e) Exterior View on full Car

Figure 3.1: Components of a computer modelled car ©Dosch Design[2]

Combinations of these options to achieve lower bit-rate compression for image and video content is known as Model Based Coding. The term Model Based Coding (MBC) was coined in the middle of the 20th century. Only two years after the telephone had been patented, in 1878 early concepts of videophones were patented as well. The first public videophone service was introduced in Berlin in 1936 and illustrated the need for efficient video data transmission. Motivated by psychological studies showing that human facial expressions can be described by less than 50 parameters [37] Pierce envisioned a codec that only needed to communicate facial parameter changes [74]. With the advance of computer graphics Parke developed a facial model resembling a human face including expressions in 1970 [71], and in the 1980s Forchheimer et al. were able to show that MBC indeed leads to reduced data rates compared to video encoding schemes used at that time [20].

As I have outlined in [P11] the State-of-the-Art in MBC is defined by three different approaches. One approach of MBC encodes scene content by fitting parameterized object data to the content. Due to the complexity of of fitting general shapes this branch of MBC is still mostly concerned with facial coding, and problems like initialization, texture registration and modeling remain [17, 40, 16, 94]. Often human aid is required in the context of object fitting: Yao for example employs an human assisted initialization step in order to fit a facial model to a real face in a video-telephony scenario [112].

Shape based encoding is the 2D approach to MBC. While most image and video coding schemes work on rectangular blocks (see e.g. JPEG encoding,

Chapter 5.1.1), shape based encoding considers shapes fitted to 2D scene content as coding blocks. Free shape encoding is included in the MPEG-4 Visual (part 2) standard[3], but is in reality hardly ever used due to the computational and algorithmic complexity this approach poses on the encoder compared to rectangular blocks.

Novel capturing devices allow to capture not only color but also depth information (see e.g. Chapter 1.2). A third approach makes use of such 3D acquisition methods to gather 3D scene data. MBC can use such acquired 3D information as a projection screen for captured texture [87]. For arbitrary scene content this method currently returns the best results but also lacks the feature of having freely parameterizable and reusable objects.

This work considers the first approach to MBC which strives for a 3D synthetic object representation of as much content of a real scene as possible. If the goal is to achieve exactly the same realism as contained in captured data with model based coding, a purely synthetic representation is not sufficient. In fact, synthetic data can only be used to predict the real content to a certain degree, and in addition to the synthetic information residual information needs to be encoded to create the same level of realism as observed in captured data. A model based encoder will therefore try to approximate scene content as good as possible by synthetic data, encode the parameter selection and texture, and observe the residual information remaining after scene approximation by synthetic data. The decoder needs models and their parameters, the textures for the synthetic content and the residual information to reconstruct captured data truthfully. This very high-level structure of a model-based codec is presented in Figure 3.2 [P10].

As stated before, the size of full 3D models is prohibitive in most cases if they should be used for data reduction. Therefore, models used for scene approximation either need to be simplified dramatically or reused to compensate for their additional cost. Both of these options are considered in the following sections: Section 3.1 details ideas developed with respect to model simplification, and Section 3.2 elaborates on research reusing model content for video material prediction.

The results of these approaches are twofold: They are promising, as significant success in data reduction can indeed be achieved. At the same time the results are also very illustrative, as success is limited to very artificial scenarios only. One of the obstacles that need to be taken to make MBC applicable for a broader range of use-cases is one of the motivations for research presented in Chapter 5.

[3]ITU-T Recommendation ISO/IEC 14496-2:2004 “Coding of Audio-Visual Objects, Part 2: Visual”

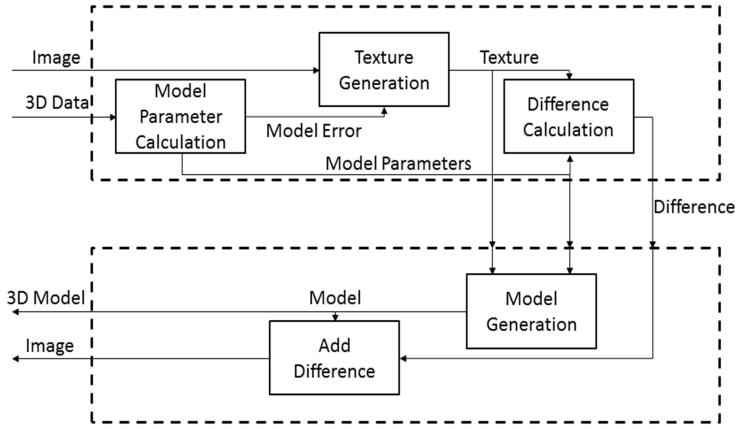


Figure 3.2: High-level Structure of Model Based Codec

3.1 Model Simplification

3D Models are represented by sets of vertices. These vertices can have multiple attributes like colors or normal vectors, but most important require a position in 3D space. By connecting these vertices with edges full 3D structures or meshes are created. The major approaches to mesh simplification are vertex clustering, decimation, resampling and mesh approximation. Common to all these approaches is that the visual quality of the mesh and the mesh size are directly related, as illustrated in Figure 3.3 using the well-known Stanford Bunny[4] as an example.

3.1.1 Vertex Clustering

Vertex clustering is the approach of simplifying mesh data by clustering "close" vertices into single points. "Close" is not necessarily restricted to the spatial domain but can include any other mesh knowledge incorporated in a distance metric. In 1993 already Rossignac and Borrel described an algorithmic approach which separates an arbitrary input model into a grid of cells. All vertices within a cell are then clustered to a single new vertex (see Figure 3.4a) [83]. While this approach is straightforward and the error limited by design to the grid size, edges and peaks of the original mesh often disappear in the simplification process.

Better results were achieved by Hoppe et al. with their approach of minimizing an energy function. This energy function is designed to keep the distance to the original mesh small, minimize the number of used vertices and

[4]Courtesy of the Stanford Computer Graphics Laboratory.

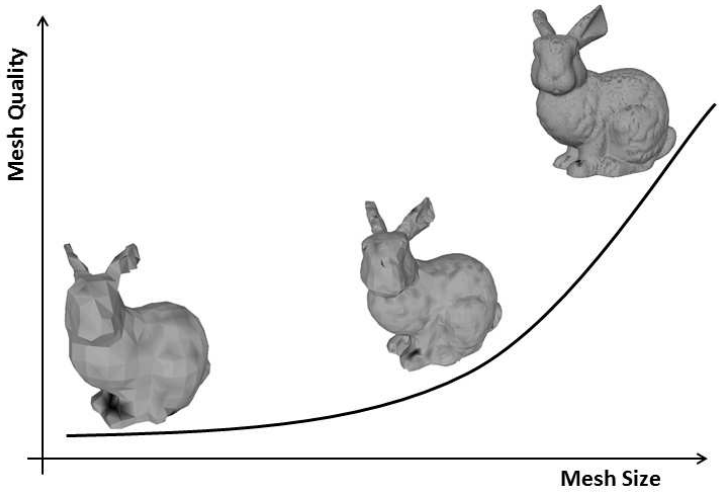


Figure 3.3: Relation of Mesh Size and Mesh Quality

penalize long edges. The edge penalization is necessary, as an energy function composed of only the first two characteristics might not have local minima and therefore result in unconstrained peaks [38].

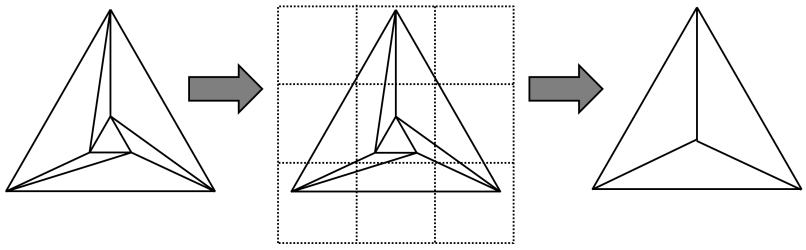
While Hoppe et al. define all simplification steps as concatenations of edge splits and edge collapses, Garland and Heckbert allow also contractions of non-edge vertices. This is achieved by quadratic error functions defined for each vertex of a mesh [28]. Quadratic error functions have been a method of choice afterward for mesh simplification using vertex clustering.

If the rendering size of a mesh is known, an algorithm can predetermine which vertices will be combined in the same pixel representation. This knowledge can than be used to combine said vertices in a single vertex, effectively reducing a mesh without compromising the rendering quality [50].

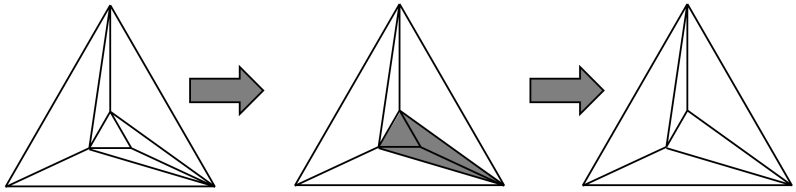
3.1.2 Vertex Decimation

While vertex clustering reduces the complexity of a mesh by combining vertices into new ones, vertex decimation is the idea of reducing the complexity of a mesh by extracting irrelevant vertices. Schroeder et al. describe an approach which iteratively removes vertices fulfilling certain removal criteria [89]. In each iteration the mesh is analyzed for the local vertex geometry and mesh topology and the removal criteria are evaluated for all vertices. After the vertices fulfilling the criteria are removed the resulting hole needs to be triangulated, and another iteration can be done.

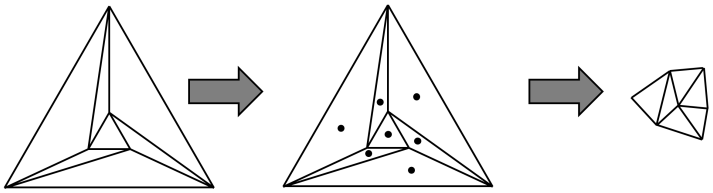
Vertices which are candidates for removal are all vertices which are either surrounded by a full or semi-cycle of triangles (see Fig. 3.5), so called complex



(a) Mesh Simplification by Vertex Clustering: Spatially Close Vertices are combined to a single Vertex [83]



(b) Mesh Simplification by Vertex Decimation: Vertices surrounded by a Full Cycle are removed [89]



(c) Mesh Simplification by Resampling: New Vertices are placed on existing Faces [32]

Figure 3.4: Mesh Simplification Approaches

(a) Simple Vertex surrounded by Full Cycle

(b) Boundary Vertex surrounded by Semi-Cycle

(c) Complex Vertex with Neighbors not forming a Cycle

Figure 3.5: Different kinds of Mesh Vertices [89]

vertices of which the neighboring vertices do not form a cycle remain part of a mesh. The main deletion criterion is the distance-to-plane or distance-to-boundary criterion. If a simple vertex is below a given thresholding distance d away from the average plane defined by its neighbors it is deleted, and if a boundary vertex is below this distance d from the boundary defined by two of its neighbors it is also removed (see Figure 3.4b).

3.1.3 Resampling

Both, vertex clustering and vertex decimation, depend on the original set of mesh vertices. Depending on the original mesh structure approaches based on a suboptimal structure may be limited with respect to the achievable simplification and quality. Resampling is an approch to the mesh simplification problem which creates a completely new and independent set of vertices. This novel set of vertices is distributed on the original mesh surface such that a new mesh after triangulation of these new vertices exhibits a topology similar to the original mesh. In addition to simplification, resampling has the major benefit of being able to regularize a mesh structure.

A classical approach of mesh complexity control by resampling a mesh is explained by Gotsman et al. By placing new vertices on existing faces the number of vertices can be adjusted. As large meshes have on the average twice as many faces as vertices [70], this method can increase the total number of vertices. Alternatively, by thresholding the size of faces and placing vertices only on faces above a certain size vertices can be decimated and the vertex count can be arbitrarily reduced [32].

With respect to regularization Turk presents an approach in which a given number of vertices is placed on the surface of a mesh, and these vertices are

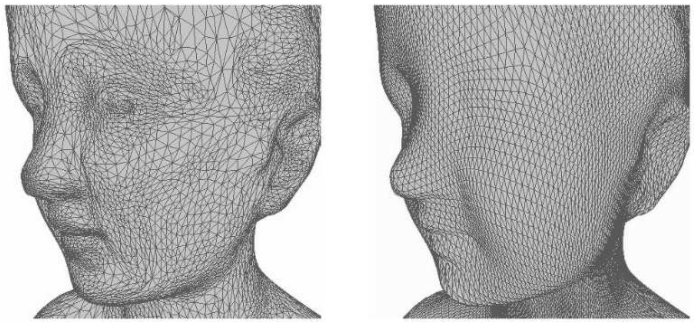


Figure 3.6: Regularization of Mesh by Resampling [32]

positioned by repulsion. All vertices try to maximize the distance to their neighbors, constrained to the surface of a given mesh. By this approach a regular structure is obtained with a uniform distribution of vertices [99]. An example of such a regularized mesh is shown in Figure 3.6.

3.1.4 Mesh Approximation

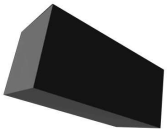
Similar to resampling mesh approximation defines a new mesh structure not associated with the initial mesh vertices. The core idea of mesh approximation is to represent a mesh by simple building blocks. We have introduced one approach which considers geometric primitives as fundamental building blocks in [P10]. The algorithmic procedure is based on primitive fitting to clustered vertex clouds. Primitives used are cuboids and ellipsoids, where cuboids are defined by six orthogonal rectangular faces, and ellipsoids as given by the general equation of ellipsoids

$$(\vec{x} - \vec{v})^T A^{-1} (\vec{x} - \vec{v}) = 1 \tag{3.1}$$

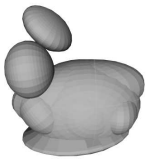
where $\vec{x}$ is the coordinate vector to any point on the surface of the ellipsoid, $\vec{v}$ is the vector to the center of the ellipsoid and A is a positive definite matrix with $A = Q\lambda Q^{-1}$. Both Q and λ are quadratic matrices. The columns of Q correspond to the ellipsoid axes, and $\lambda_{i,j} = radius_i$ for $i = j$ and $\lambda_{i,j} = 0$ else. Both, cuboids and ellipsoids can be fitted to point clouds in a least square sense. The approach we have presented in [P10] fits both geometric primitives to the underlying point clouds and calculates the normalized approximation error (see Section 3.1.5) for a mesh approximation by measuring the distance from each of the original mesh vertices to the closest new model surface. Whichever of the primitives returns the smaller normalized error is used to represent the point cloud in the approximation.

(a) Model of a Wooden Block

(b) Model of the Stanford Bunny



(c) Approximation of the Wooden Block 3.7a by Geometric Primitives



(d) Approximation of the Stanford Bunny 3.7b by Geometric Primitives

Figure 3.7: Mesh Approximation using Geometric Primitives [P10]

Examples of such a mesh approximation are shown in Figure 3.7 on the models of the Stanford Bunny[5] and a Wooden Block[6] dataset.

Bokeloh et al. follow a conceptually similar approach to approximate an original mesh by simple building blocks [3]. The core difference is that instead of using geometric primitives as building blocks, reoccurring parts of the original input mesh are detected and reused to approximate the full mesh structure. Next to mesh simplification this approach comes with several other advantages, like quality improvement of noisy data or propagation of edits to similar building blocks.

The algorithmic approach to find such building blocks has three main steps, which are feature detection, matching and geometric validation. A core idea of the feature detection step is slippage analysis which computes components possessing so called "slippable motions", motions which do not result in gaps when applied to a copy of the original component superimposed with the original. [29] Finding matches among the feature components applies base matching from the feature space and an iterative closest line detection (as presented in [85]). RANSAC iterations over the two named approaches reduce the possible

[5]Courtesy of the Stanford Computer Graphics Laboratory.

[6]Courtesy of the Heidelberg Graduate School of Mathematics and Computational Methods for the Sciences and the Interdisciplinary Center for Scientific Computing.

matches to the best match, which is than validated by comparing the mesh composed of individual parts to the original mesh [3].

3.1.5 Model Error

All of the above mentioned mesh simplification approaches describe only approximations of the original input mesh. This means that by simplifying a mesh errors are introduced. The smaller the error the better the quality of the model. Under the assumption that the original model is perfect, all of its vertices potentially present features of this model. By measuring the distance of the original feature points to the simplified mesh surface a model error can be calculated which represents the overall observed model quality. We have presented such an approach in [P10].

In a first step the difference between each vertex of the input model and the approximation mesh surface is calculated. In 3D space this distance is either the closest distance to a vertex, to an edge or to a face of the approximating mesh. A full search over all vertices, edges and faces would definitely return the smallest distance, but this search space can be dramatically reduced. For each vertex of the original mesh the distance to the approximating mesh is found by the following procedure.

First, the closest vertex from the set of all vertices of the approximating mesh is calculated. The point-to-point Euclidean distance between points $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ in 3D space is given by

$$d(p_1, p_2) = |p_2 - p_1| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{3.2}$$

However, after the first distance calculation a third vertex p_3 can only be closer to p_1 than p_2 if $max(|x_1 - x_3|, |y_1 - y_3|, |z_1 - z_3|) \leq d(p_1, p_2)$, which already reduces the number of necessary distance calculations significantly. The shortest distance calculated in this process is denoted by d_{min}.

Second, the shortest distance between the edges of the approximating mesh and the considered vertex are calculated. The Euclidean distance between a line through two points $p_2 = (x_2, y_2, z_2)$ and $p_3 = (x_3, y_3, z_3)$ and a point $p_1 = (x_1, y_1, z_1)$ in 3D space is given by

$$d(p_1, p_2, p_3) = \frac{|(p_3 - p_2) \times (p_2 - p_1)|}{|p_3 - p_2|} \tag{3.3}$$

For the mesh-to-vertex distance this distance is however only relevant, if the line segment closest to the vertex is part of the mesh. Knowing that the closest point-to-line distance is orthogonal to the line going through the points

p_2 and p_3 the Pythagorean theorem can be applied to formulate this restriction. $d(p_1, p_2, p_3)$ gives a new minimal distance d_{min} if and only if $(d(p_1, p_2, p_3))^2 + |p_3 + (p_2 - p_3) \cdot t|^2 = |p_3 - p_1|^2$ for $0 \leq t \leq 1$.

Similar to the point-to-point distance calculation the number of necessary point-to-line distance calculations can be reduced to the edges connecting vertices p_2 and p_3 with

$$\begin{aligned} &min(x_2, x_3) < (x_1 + d_{min}) \wedge \\ &min(y_2, y_3) < (y_1 + d_{min}) \wedge \\ &min(z_2, z_3) < (z_1 + d_{min}) \wedge \\ &max(x_2, x_3) > (x_1 - d_{min}) \wedge \\ &max(y_2, y_3) > (y_1 - d_{min}) \wedge \\ &max(z_2, z_3) > (z_1 - d_{min}) \end{aligned}$$

.

If a distance $d(p_1, p_2, p_3)$ is found which is less than d_{min} the minimal distance is updated by the new value.

After checking for closest vertices and closest edges the third step is the computationally most complex calculation of the shortest point-to-face distance for all faces of the approximating mesh. A face of a triangulated mesh is given by three vertices p_2, p_3 and p_4. The shortest distance of a point p_1 to a plane defined by such three points is calculated as

$$d(p_1, p_2, p_3, p_4) = \vec{n} \cdot (p_1 - p_2) \tag{3.4}$$

where $\vec{n}$ is the planes normal vector given by

$$\vec{n} = \frac{(p_4 - p_2) \times (p_3 - p_2)}{|(p_4 - p_2) \times (p_3 - p_2)|} \tag{3.5}$$

Analogue to the point-to-line distance calculation this distance is only relevant for the mesh-to-vertex distance if the plane segment closest to the vertex is part of the face, or in other words, if the intersection point between the line defined by p_1 and the planes normal $\vec{n}$ and the plane defined by vertices p_2, p_3 and p_4 is inside the triangle of p_2, p_3 and p_4. If this intersection point is denoted as f and $\vec{x}$ denotes the vector from the coordinate origin to point x than $\vec{f} = \vec{p_2} + r \cdot (\vec{p_3} - \vec{p_2}) + s \cdot (\vec{p_4} - \vec{p_2})$ needs to be fulfilled for $0 \leq r, s \leq 1$ and $0 \leq (r + s) \leq 1$.

Again, it is not necessary to calculate this distance for all faces of a mesh. With analogue knowledge as for the point-to-line distances the set of faces can be narrowed down to those faces with $min(x_2, x_3, x_4) < (x_1 + d_{min}) \wedge min(y_2, y_3, y_4) < (y_1 + d_{min}) \wedge min(z_2, z_3, z_4) < (z_1 + d_{min}) \wedge max(x_2, x_3, x_4) > (x_1 - d_{min}) \wedge max(y_2, y_3, y_4) > (y_1 - d_{min}) \wedge max(z_2, z_3, z_4) > (z_1 - d_{min})$.

Following these calculations for each of the original vertices results in an error value per vertex of the original mesh. The sum of all these individual vertex errors is given by the approximation error $e(v, s)$. This error still depends on the number of vertices v of the original mesh and the original mesh size s. $e(v, s)$ can be normalized to

$$\bar{e} = \frac{e(v, s)}{(v \cdot s)} \tag{3.6}$$

where

$$s = \frac{1}{\sqrt{3}} \left| \frac{1}{2} \begin{pmatrix} M_x - m_x \\ M_y - m_y \\ M_z - m_z \end{pmatrix} \right| \tag{3.7}$$

with M_i the maximal and m_i the minimal value of dimension i of the original mesh. This normalized error $\bar{e}$ is independent of the original mesh scale and surface sampling, thus represents an error measure which is comparable over meshes of different size and sampling density, suitable for all kinds of mesh simplifications and approximations. The value calculated by this normalized error corresponds to the average distance of vertices to the surface of a unit sphere, the sphere with radius 1 and center at $x = 0$, $y = 0$ and $z = 0$.

Next to the decision about which primitive to choose for a cluster approximation as explained in Section 3.1.4 the normalized error can be an indicator for the required texture quality..

3.1.6 Texture Quality

For objects in 3D space the objects form is given by a mesh, containing vertices and faces. Color is usually added to objects by mapping textures to the mesh surface. Figure 3.8 illustrates the individual parts of a textured 3D object: a mesh representing the 3D topology and a texture with little structural information are combined to a textured 3D model. While the previous sections are concerned with mesh representations in the context of model based coding, this section focuses on the texture representation.

The core idea for texture representations in the context of model based coding is as follows. Model simplifications will lead to noticeable artifacts, independent of the texture quality. This means that, independent of the texture quality, residual information is required to overcome these artifacts. In the worst case scenario of the synthetic model not reflecting any content of the captured scene all information remains in the residual information and all information used to texture the model is wasted. In other words, if the model

(a) Untextured Mesh (b) Plain 2D Texture (c) Texture 3.8b mapped to mesh 3.8a

Figure 3.8: Mesh and Texture Example ©2006 Sony Computer Entertainment Inc.

has a low quality and cannot reproduce the real content little information shall be used for the texture information and most information remains in the residual information. On the other hand, if the model provides an adequate representation of the scene content, high quality textures shall be applied to this model. This effectively moves the information from residual to texture information.

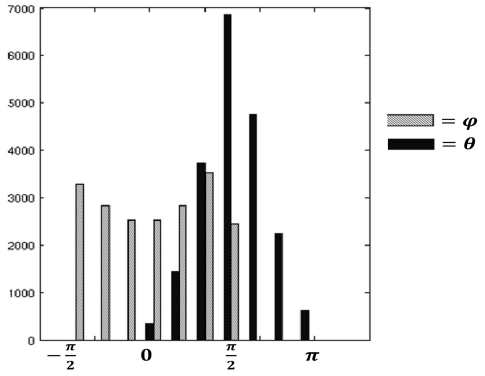
When applied correctly this balance between texture and residual information can be employed to ensure that the only additional data is contained in the mesh data: Color information is in the worst case fully contained in the residual data, in the best case fully contained in the texture data, and in most cases somehow distributed in between both.

This "distributed in between both" can be specified more precisely. An important indicator for the model quality is the model error $\bar{e}$ (see Section 3.1.5), assuming that a perfect model or some vertices from 3D capturing hardware as outlined in Section 1.2 exist for the scene content.

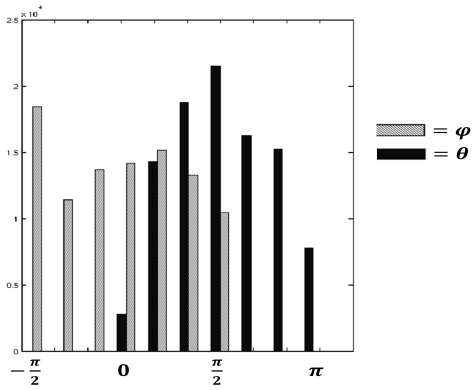
It is important to note that the model error $\bar{e}$ serves only as an indicator for the visible artifacts when rendering a textured model. Exemplarily consider a planar model which is approximated by a plane with a slight parallel offset. If this offset is orthogonal to the viewing direction the rendering result is unchanged. However, if this offset occurs at a different angle towards the viewing angle noticeable artifacts appear. In order to relate the measured model error $\bar{e}$ to the resulting artifacts an approximation is proposed in [P10]. First, as explained in [54], an error which results in the deviation of mesh vertices from an original mesh can be assumed to be a displacement along the surface normal. Second, for uniformly sampled complex objects the distribution of surface normals is almost uniform. This second assumption is motivated by

(a) Actor Model

(b) Stanford Bunny Model



(c) Distribution of Actor Surface Normals with polar angle θ and azimuth angle φ



(d) Distribution of Bunny Surface Normals with polar angle θ and azimuth angle φ

Figure 3.9: Distribution of Surface Normals of the Stanford Bunny and of a Performance Capture Actor

the observation of surface normal distributions of several different 3D objects, among them the Stanford Bunny[7] and a Performance Capture Model[8], both shown in Figure 3.9. While the upper row of Figure 3.9 shows the rendered models, the lower row gives histograms of the normal vector distribution.

As I describe in [P11] for a general model a uniform normal vector distribution and, corresponding to that, uniformly distributed error vectors can be assumed. The displacement vector $\vec{d}$ in a spherical coordinate system for vertices of a general model is therefore given by

$$\vec{d} = \bar{e} \cdot \begin{pmatrix} sin(\tau) \cdot cos(\phi) \\ sin(\tau) \cdot sin(\phi) \\ 0 \end{pmatrix} \tag{3.8}$$

where $\bar{e}$ as defined in Equation 3.6 and τ,ϕ are uniformly distributed di-

[7]Courtesy of the Stanford Computer Graphics Laboratory.

[8]Courtesy of the 3D Vision and Vision-Based Graphics Research Group of the Max-Planck-Center for Visual Computing and Communication.

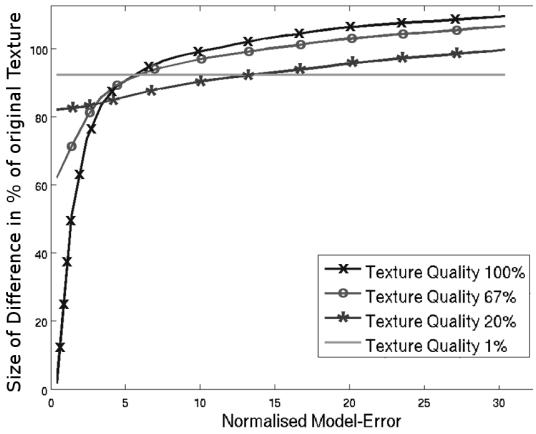


Figure 3.10: Residual Information per Model Error $\bar{e}$ at different Texture Quality Levels

rections of the normal vectors in spherical coordinates, modeled by uniformly random distributed variables. These approximations of the visible error occurring when mapping a texture to an incorrect 3D model can be applied to a 2D texture without an underlying 3D model. The resulting texture is then given by

$$t_{out}(x, y) = t_{in}(x + \bar{e} \cdot sin(\tau) \cdot cos(\phi), y + \bar{e} \cdot sin(\tau) \cdot sin(\phi)) \tag{3.9}$$

With uniformly distributed τ, ϕ in experiments the approximations $\tau = x$ and $\phi = y$ can be made. Furthermore, if output pixels overlap the linear average of those is considered, if output pixels remain empty the original source $t_{out} = t_{in}$ is considered for those pixels.

The relation between model quality (or the model error), texture quality and the residual information was evaluted experimentally with the approximations explained above. Figure 3.10 denotes the residual size of a JPEG-coded texture on the vertical axis and the model error $\bar{e}$ on the horizontal axis. Different texture qualities are presented by different lines of the graph. Here a texture quality is equivalent to the texture resolution: which means that exemplarily a texture quality of 1% describes a texture file of the original resolution downscaled by a factor of 100.

According to [P10] three important observations can be made:

- The higher the texture quality, the larger the possible gain if the model is good, but the larger the residual information also gets in case of an inaccurate model.

Table 3.1: Sizes of Components Contributing to Model-Based Encoded Content

Model	Parameter Size (B)	Texture Size (kB)	Residual Size (kB)	Total Size (kB)	Original Size (kB)
Wooden Block	192	63	8	71	74
Actor	1466	90	503	505	546

- If the model quality decreases beyond a certain point ($\bar{e} > 15$) the smallest residual information is achieved by the DC component of an image only.
- Only for models of very high quality ($\bar{e} < 3$) the texture should be used in its original quality.

The observation that residual information can even increase compared to the original texture size seems surprising at the first glance. The explanation is to be found in new and higher frequent content being added to the residual information by projecting a high-frequent texture onto an erroneous model.

The knowledge gained by these experiments can be applied to optimize model textures according to the quality model obtained through a simplification process. As we present in [P10] this process of model simplification by approximation through geometric primitives, texture adjustment and residual information can be used to efficiently code image content and as a byproduct gain 3D model information which can be further used and processed. Table 3.1 gives the data sizes required to transmit the two models introduced in Figure 3.7. For the encoding process an encoder was implemented according to the structural layout shown in Figure 3.11. The parameters marked by dotted circles are those contributing to the total transmitted data, as given in Table 3.1. All of the individual building blocks of the Model Based Encoder are explained in detail in the sections above.

3.2 Model Re-Use

As described in the previous section model based coding with full 3D models for still images is promising only if a high quality model can be stored at negligible costs. An alternative to reduce model costs per usage is to re-use models several times. Such a scenario is given by a movie sequence. Most objects in movies occur several times and are viewed from multiple directions. The more often models can be reused to render parts of a scene, the larger the benefit. In the extreme case an infinite re-usage of models can be envisioned, or a scenario in which models “are just there”, since they are stored in local databases.

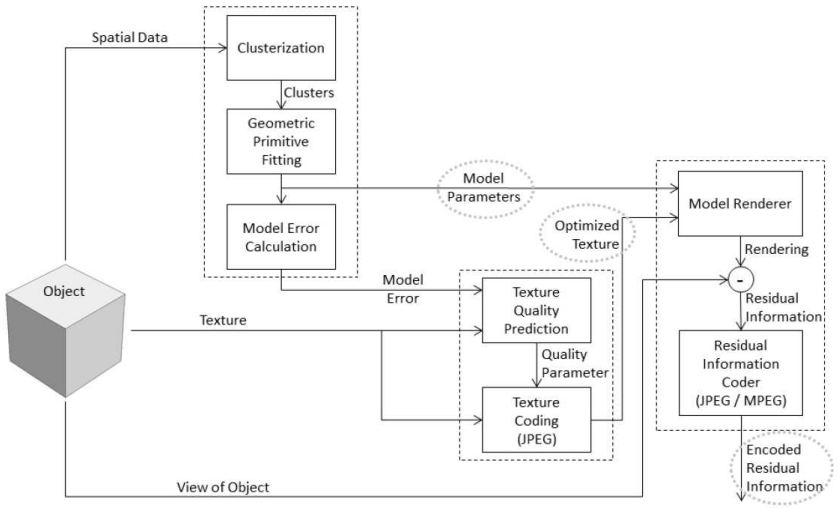


Figure 3.11: Structure of a Model Based Image Coder

Important to the ideas of model re-usage is the form of data that can be encoded model based. As re-occurring models are most common in videos, different from the previous section this section focuses on encoding of video material. The following section outlines the encoding structure of H.264 which still is one of the most widely used video codecs. Based on H.264 a model based enhancement to the codec is presented which allows frame prediction from both, previous frames and synthetic models. As the parts of H.264 relevant for model based encoding have remained unchanged in the current state-of-the-art H.265 encoder, this enhancement is still a valid option for today's generation of video codecs.

3.2.1 The H.264 and H.265 Video Coding Standards

Efficient video coding is a problem that evolves with the requirements created by the kind of video data and the way videos are transmitted. MPEG-2 (or ITU-T H.262) had been a standard for about ten years until the turn of the millenium and had enabled digital video broadcasting worldwide. At that time most video content was standard definition (SD) and broadcasted over satellite, cable or terrestrial. However, at the beginning of the 2000s an increasing popularity of HD content and IP-based networks as an additional channel to distribute videos created the demand for higher coding efficiency.

Individually the ITU-T had standardized H.261 in 1988 and H.263 in 1995, and in 1991 MPEG-1 was standardized by ISO/IEC. Following the first joined standardization of a video coding scheme with MPEG-2 / H.262 the ITU-T

now set out to define the novel video coding standard H.264 together with ISO/IEC, who included H.264 as Part 10 - Advanced Video Coding (AVC) in MPEG-4[9]. Similar to all other ITU-T and ISO/IEC video coding standards only the decoder is standardized. This has the benefit of allowing flexibility on the encoder side, enabling optimizations and trade offs necessary for low-complexity encoding. On the downside, not specifying the encoder side cannot guarantee a certain quality, as coarsely implemented encoders may provide conforming video streams of low quality.

With the central idea of doubling the coding efficiency (or cutting the necessary bits for a given video quality by half), H.264 was designed to meet numerous other demands. Among these demands are the ability to broadcast data not only via cable, satellite or terrestrial transmitters, but also ISDN, DSL LAN and Ethernet as well as mobile networks. Video should be suitable for interactive storage (HDDs, DVDs, ...), conversational services (video telephony), video on demand and multimedia messaging services. This range of demands motivated the use of two layers described in H.264: a video coding layer (VCL) for efficient representation of video data and a network abstraction layer (NAL) to appropriately format and present the VCL for different transport layers or storage requirements.

To achieve the goal of an enhanced coding efficiency by a factor of two several content prediction approaches are necessary to reduce redundancy in information. In [107] Wiegand et al. name several of these. H.264 has enhanced the flexibility of prediction block sizes and shapes compared to previous standards, and increased the precision of motion compensation for these blocks from half to quarter of a sample size. Different from previous standards motion vectors in H.264 are allowed to point over picture boundaries. Motion prediction in H.264 can be done from multiple reference pictures, and the order of reference images is independent from the display order, thus strongly increasing the options for good prediction. Block based image and video coding creates blocking artifacts (see Section 5.1.1). These blocking artifacts can be reduced by a deblocking-filter, which in H.264 is integrated in the prediction loop. In addition to these new concepts H.264 increases coding efficiency at a trade off for coding complexity by allowing smaller coding block sizes and novel approaches for arithmetic and content adaptive entropy coding.

A core component of the NAL are the NAL-units. Each NAL-unit is a package containing an integer number of bytes. The first byte of each package is a header type specifying the data content, the remaining bytes are payload as specified by the header. All information of the VCL is contained in such

[9]ITU-T Recommendation "H.264: Advanced video coding for generic audiovisual services"

NAL-units. In addition to VCL-data non-VCL content such as parameter sets are packaged in NAL units. These parameter sets define the decoding structure of video sequences or individual pictures within a video sequence. This NAL structure enables H-264 for both package- and bitstream oriented transmission schemes.

To enable interactive access to a video stream so called access units are defined. Access units are sets of NAL units which can be decoded independent of other NAL units into a primary coded picture.

In the VCL pictures may be composed of a full frame or a single field when consecutive fields are captured at different times. Usually pictures are split into top and bottom field, where the top field contains all even rows and the bottom field all odd rows. Video data is represented in YCbCr color space which separates luminance from color due to the observation that the human visual system is more sensible to brightness than to color (see Section 5.5.2). On the YCbCr color represenation H.264 downsamples the chroma components in both horizontal and vertical direction by a factor of two, which is known as $4:2:0$ sampling with a precision of $8bits$ per sample. For decoding, picture information is partitioned in macroblocks of 16×16 luma samples. These macroblocks are grouped into slices, where each slice is self-content with respect to decodability given the picture parameter sets. A picture may be composed of one or several slices. This allows to differentiate between three kinds of slices:

- I Slice: if all macroblocks of a slice are coded using intra-prediction (only the information contained in that slice) a slice is called I Slice
- P Slice: if in addition to intra-prediction motion-compensated prediction (information from another slice is motion compensated and used for prediction) from at most one other signal per macro-block is used a slice is called P Slice
- B Slice: if in addition to the encoding types of I and P Slices motion-compensated prediction from two other signals is used a slice is calld B-Slice
- SP Slice: SP Slices are switching P Slides which are coded to efficiently switch between different precoded pictures
- SI Slice: SI Slices are exact macroblock matches in SP Slides to allow random access or error recovery

The structure explained in the paragraphs above is shown in Figure 3.12 [107].

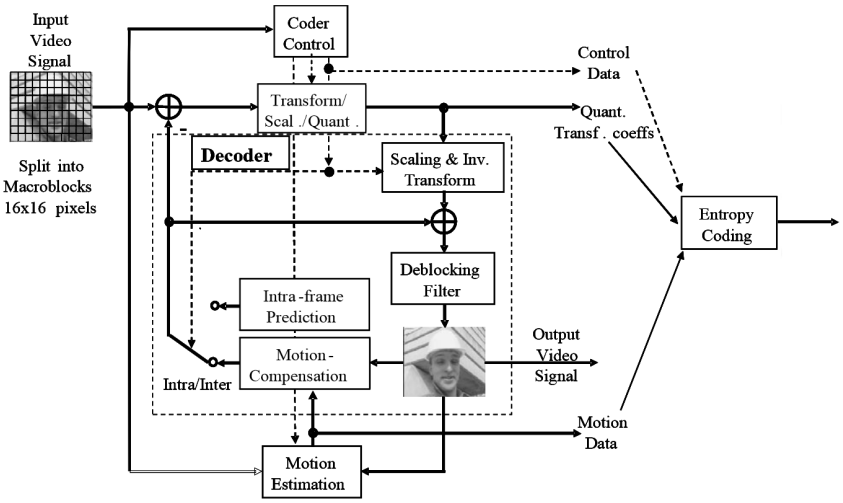


Figure 3.12: Basic Structure of a H.264 Encoder [107]

Ten years after the standardization of H.264 changing demands, novel algorithms and new hardware once again motivated the development of a video coding standard. In 2013 HEVC was introduced by ISO/IEC Moving Picture Experts Group as MPEG-H Part 2 and by ITU-T Video Coding Experts Group as Recommendation H.265 as a new standard for new video coding[10]. Motivating factors were the increasing demand for HD- and beyond-HD formats, increasing diversity of services and increasing video playback on mobile devices. Again, the standardization offices had the ambitious goal to increase the coding efficiency for general video content by two with respect to the previous state-of-the-art, H.264.

In their overview of the HEVC standard [97] Sullivan et al. outline the functionality of HEVC. While the overall structure, as shown in Figure 3.13, remains largely the same compared to the structure of an H.264 encoder (shown in Figure 3.12), there are a few noteworthy differences. A core difference is that the macroblock structure, which was used in H.264, is replaced by coding tree units (CTUs) in H.265. The CTU size is not limited to 16×16 samples any more but can be up to 64×64 samples in size which directly reflects in the coding efficiency. As their name implies, these CTUs can be partitioned into smaller blocks using a tree structure. The decision about inter- or intrapicture prediction is made on the blocks forming a CTU.

In addition to this significant change of the coding block structure several components like motion compensation, intrapicture prediction, quantization or entropy coding were improved with respect to precision, number of options

[10]ITU-T Recommendation "H.265: High efficiency video coding"

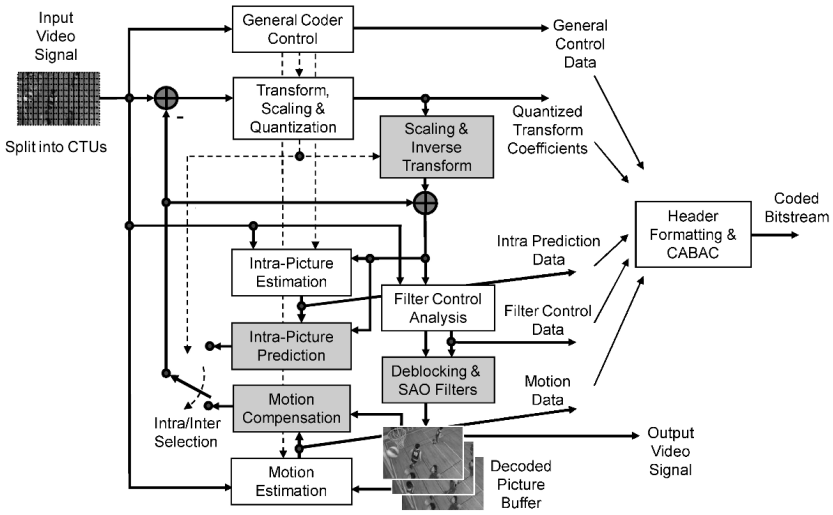


Figure 3.13: Basic Structure of a H.265 Encoder [97]

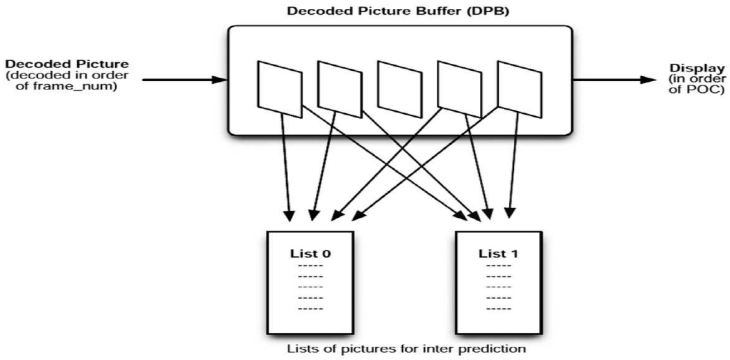


Figure 3.14: List Structure of Decoded Picture Buffer [82]

and efficiency.

Central to the coding efficiency of both, AVC and HEVC, is the prediction from already encoded images which are stored in the encoded / decoded picture buffer of the encoder and decoder respectively. In Figures 3.12 and 3.13 this buffer is indicated by a decoded image. Two separate lists order the pictures in the coded picture buffer for usage in I-, P- and B-Slices. While I-Slices do not require any additional input from the coded picture buffer, P-Slices use one of the two lists and B-Slices refer to both lists for en- and decoding. This list structure of the decoded picture buffer is presented in Figure 3.14.

3.2.2 Hybrid Model Based Video Coding

Video coding for real video contents is already well advanced, the data rates achieved by HEVC are difficult to compete with. However, Model Based Coding can offer an interesting enhancement to the state-of-the-art video encoding schemes, for both HEVC as well as the currently widely used AVC. The core idea is neither to not rely purely on model based coding nor on traditional predictive coding, but to extend the current state-of-the-art encoders by a model based option. If model based encoding can achieve better results than traditional encoding, model based encoding can be used. If the results achievable with a model based approach are worse than without, the encoder should decide for the traditional encoding scheme.

A hybrid scheme integrating model information into existing image and video coding techniques was first introduced by Musmann et al. in 1989 [64]. Instead of fixed size blocks of content the motion of 2D or 3D objects in a scene were parameterized and encoded. With this approach Musman et al. were able to reduce the data rate compared to the state-of-the-art of 1989. In 2D this idea is integrated in AVC where motion prediction for arbitrary shapes is allowed.

A hybrid encoding scheme introducing 3D model based coding for the H.264 standard is described in [P6]. Here the reference implementation of AVC (which is part of both ITU-T and ISO/IEC standard) has been extended by Model Based Coding, but considering the modifications it is inherently clear that the same enhancements as proposed for AVC also apply to HEVC.

In [P6] we assume that the same models are available for encoder and decoder. These models can either be made available through a central database or from previous transmissions. If a model is transmitted as part of the overall transmitted data, any gain in coding efficiency of the video content needs to be balanced by the cost of the model transmission. Furthermore, it is assumed that a scene-analysis step is preceding the encoding step. This means, that the model parametrization fitting to the scene content is already available. The position of a model in a scene, as well as its scale and rotation are already known from previous scene analysis (as described in Section 2.2) and model fitting (see Section 3.1.4) steps.

Novel to the implementation of a model based enhancement for AVC are two elements. First, a renderer needs to be provided to encoder and decoder. The renderer is responsible for turning model and parameter information into image and video content that can be used by the encoder and decoder respectively. It is important that both, encoder and decoder, make use of exactly the same renderer, as deviations from rendered visual information used for encoding content will consequently lead to a different decoding. For the proof-

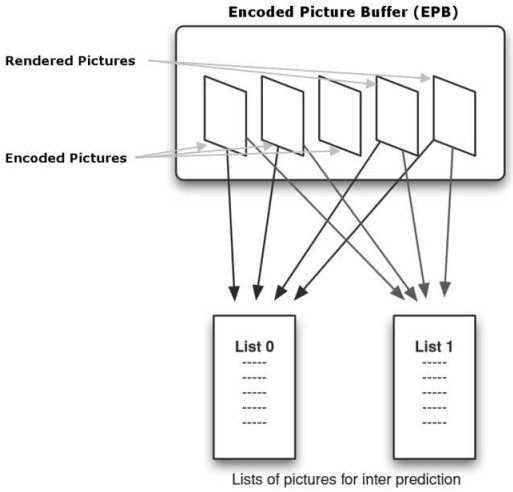


Figure 3.15: List Structure of Encoded Picture Buffer for Model Based Enhancement

of-concept implementation introduced in [P6] a simple OpenGL renderer was implemented. However, as the rendering quality directly correlates to the possible coding gain (as discussed in Section 3.1), renderers with higher rendering quality or a higher level of realism are suggested. An applicable renderer is, for example, the SCENE renderer which is introduced in Chapter 4.2.

Second, the rendered images need to be provided to the encoder (and similarly to the decoder), in a way that the encoder can decide whether to predict image content from the model rendering or from any other available frame. A suitable structure for that, which is already implemented, is the coded picture buffer. Adding renderings of models to the encoded picture buffer provides these renderings for prediction if they are added to the corresponding lists. Such an enhancement is shown in Figure 3.15. A parallel modification needs to be done to the decoder.

As model information is already information before I-slices are encoded, even I-slices can benefit from the model information. The encoder can always try to predict a slice from model renderings or previously encoded images. It automatically chooses intra coding if the quality of possible prediction content is insufficient. The structure of the enhanced AVC encoder with the ability to encode model based is shown in Figure 3.16.

The implementation of the changes introduced above was tested in several experiments. An experiment to proof the validity of the underlying assumptions and the correctness of the implementation was done by creating a video input sequence from a synthetic model. The model itself was then supplied

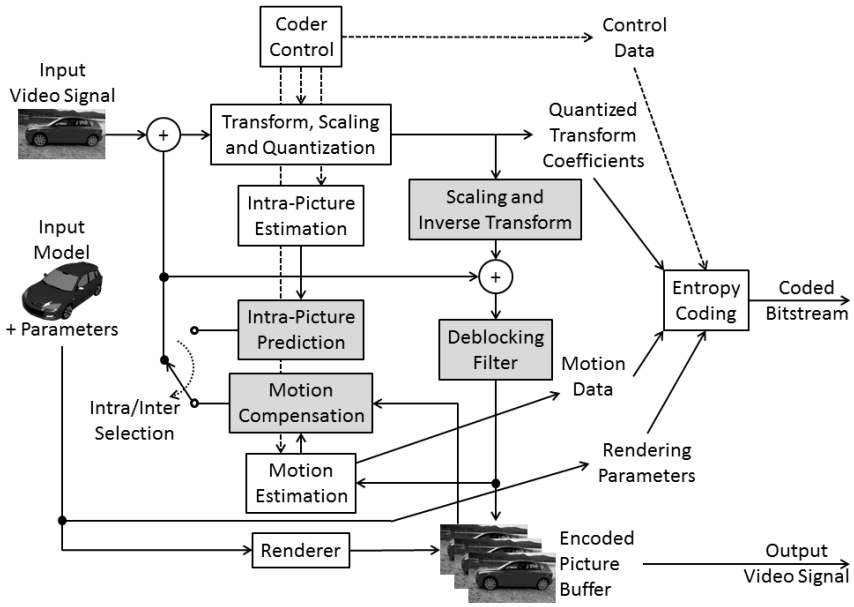


Figure 3.16: Basic Structure of the Model Based H.264 Encoder

Table 3.2: Comparison of AVC reference software to Model Based AVC enhancement on a synthetic scene

Frame No.	AVC Reference Implementation				Model Based AVC Enhancement			
	Bit/pic (bits)	SNR_Y (dB)	SNR_U (dB)	SNR_V (dB)	Bit/pic (bits)	SNR_Y (dB)	SNR_U (dB)	SNR_V (dB)
00 (MCPB)					480			
00 (NVB)	184				184			
00 (M)					2472			
00 (IDR)	172192	46.08	48.51	48.51	408	109.81	103.79	103.79
01 (P)	69048	46.95	48.51	48.51	61920	50.07	67.29	67.33
02 (P)	57272	47.48	48.51	48.51	51528	51.42	66.25	66.29
03 (P)	55480	47.79	48.51	48.51	49648	51.18	65.74	65.78
Total	354176				166640			

to encoder and decoder, providing the same parameter set that was used for rendering of the video sequence. Table 3.2 compares the results of encoding this sequence with the AVC reference implementation and the same sequence encoded with the model based AVC enhancement. In this table MCPB denote the Model Coding Parameter Bits, NVB are NAL Variable length coding Bits, M is the model frame, IDR is an I-frame with the feature of instantaneous decoder refresh, and P are P-frames. The model frame M contains data which ensures that the decoder can place a rendered model at the same position of the decoded picture buffer as the encoder and takes care of the correct placement of lists.

When comparing different encoding rates the gain is an expressive measure of how much better (or how much worse) an encoding scheme is. For compa-

rable visual qualities (by AVC expressed as the SNR of the individual channels Y, U and V) the gain G is calculated as

$$G = \frac{R - T}{T} \cdot 100 \tag{3.10}$$

The gain G is given in percent, R is the size of the reference in bits and T the size of the test, as well in bits. For the experimental data shown in Table 3.2 the coding gain of using the model based AVC enhancement compared to the AVC reference implementation is $G = \frac{354176bits - 166640bits}{166640bits} \cdot 100 = 112\%$ at a considerably better SNR per channel. The largest benefit can be found for the IDR frame, which in standard AVC coding needs to be fully intra-predicted, while perfect inter-prediction is possible in the model based AVC enhancement in this scenario where the video content is an identical replication of the model rendering.

For a more realistic experiment a video showing a Mazda 3 MPS sports car at a beach was considered. Six frames of this video sequence were extracted for tests with the model based encoding approach. These six frames are shown in Figure 3.17. In addition to that, a 3D model of the same car was used[11], see Figure 3.18.Two different scenarios were evaluated: In the first only the car model was given as an additional input to the encoder. In the second scenario apart from the model the 2D background image was set as a background image for the renderer, which is a valid assumption for scenes where objects move through an otherwise static setting.

Table 3.3 presents the different data requirements for the sequence of frames shown in Figure 3.17 when encoded with the AVC reference implementation, encoded with the model based AVC enhancement using only the car model and encoded with the model based AVC enhancement using car model and background information. Figure 3.19 visualizes the difference in the IDR frame that prediction by the models has: the images shown here presents the information still present in the IDR frame after prediction from a rendered model frame.

With Equation 3.10 the coding gain of using the Model Based AVC Enhancement with the model only is 6% and with background information available 29%. However, almost all of this gain is achieved in the encoding of the IDR frame, indicating that rendered model information is used for predicting content if nothing else is available, but previous frames can offer a better prediction in case of a sequence showing the same object.

Whenever a content change occurs the AVC encoder cannot predict information from previous frames and needs to encode a frame using intra-coding.

[11] Mazda 3M MPS 3D Model from Free3DModels, `http://thefree3dmodels.com/stuff/vehicles/mazda_3_mps/13-1-0-5352`, accessed February 2016

Figure 3.17: Frames of a Mazda Video Sequence Input to the Model Based AVC Enhancement

Figure 3.18: Model Input to the Model Based AVC Enhancement

(a) from car model and background

(b) from car model only

Figure 3.19: Visualized Difference of IDR Frame predicted by Model Based AVC Enhancement

Table 3.3: Comparison of data requirements for video sequence encoded with AVC reference software and Model Based AVC Enhancement using only a model and using model and background information

Frame No.	AVC Reference Implementation	Model Based AVC (Model Only)	Model Based AVC (Model and Background)
00 (NVB)	176bits	176bits	176bits
00 (IDR)	26456bits	20536bits	4712bits
01 (P)	14064bits	14384bits	12984bits
02 (P)	14768bits	14920bits	14696bits
03 (P)	15616bits	15424bits	15520bits
04 (P)	14584bits	14896bits	15024bits
05 (P)	15484bits	15448bits	15136bits
Total	101148bits	95784bits	78248bits

Table 3.4: Comparison of AVC reference software to Model Based AVC enhancement on a synthetic scene with scene changes after every second video frame

	AVC Reference Implementation				Model Based AVC Enhancement			
Frame No.	Bit/pic (bits)	SNR_Y (dB)	SNR_U (dB)	SNR_V (dB)	Bit/pic (bits)	SNR_Y (dB)	SNR_U (dB)	SNR_V (dB)
00 (MCPB)					1376			
00 (NVB)	184				184			
00 (M)					2472			
00 (M)					3072			
00 (M)					3072			
00 (IDR)	172192	46.08	48.51	48.51	408	109.81	87.26	87.26
01 (P)	69048	46.95	48.51	48.51	61920	50.30	67.48	67.52
02 (P)	115376	46.24	48.67	48.67	51528	89.28	82.75	82.75
03 (P)	50648	47.83	48.67	48.67	49648	50.22	66.61	66.61
04 (P)	323632	43.80	48.50	48.50	49648	89.44	85.10	85.10
05 (P)	162648	46.60	48.50	48.50	49648	47.91	65.53	65.59
Total	893728				272976			

The assumption that models enable better prediction and an increase in coding efficiency is confirmed experimentally with a video sequence, where a scene change occurs after every second frame. Numerous models suiting each of the individual scenes were initially provided to the encoder. Table 3.4 lists the achievable data rates and the corresponding signal quality for Y, U and V channel.

The described experiments lead to several observations. First, and most significant, is the observation that significant data reductions are indeed possible. Second, synthetically generated information mostly contributes to the encoding of intra-coded blocks which are found at the beginning of video sequences or after scene changes. Third, when enhancing current coding schemes the similarity metric prefers not to use synthetically generated content, even though the synthetic content has a high level of realism.

The third observation is mainly due to the fact that widely used similarity measures (e.g. PSNR) for visual information correspond little to the subjective similarity. Further more, metrics especially designed to consider the human visual system for image similarity measures exhibit difficulties when used to evaluate synthetic content. Questions arising in this context and related research is detailed in Chapter 5.

Chapter 4

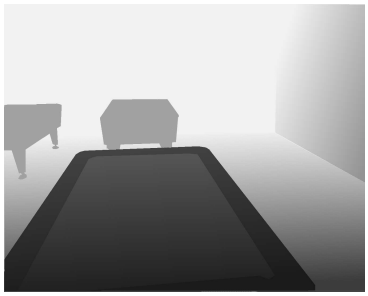
Flexibility Gain

Compared to data acquired with traditional camera systems synthetic scene content enables a tremendous amount of flexibility, which comes as processing and interaction options. With the combined representation for synthetic and captured content introduced in Chapter 2 some of the flexibility known for synthetic content becomes available for captured content as well. In the context of the SCENE project [24] some of the novel processing options enabled by this flexibility are explored, building on the data provided through the SRA. The new possibilities can be grouped into two main groups: flexibility during scene composition and flexibility for content rendering. In the first part of this chapter the additional flexibility during the scene composition is described. Here exemplary adding content from multiple sources and object modifications are achievements that have been facilitated or enabled through the new representation. The second part of this chapter discusses increased rendering flexibility. Synthetic cameras allow to reproduce physical camera properties in post-production, but also enable to go beyond physics and implement effects that stem from physical camera constraints, but can never be realized with real cameras.

4.1 Flexibility in Scene Composition

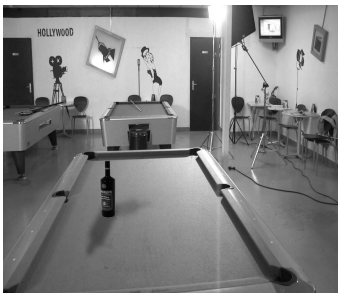
This first section introduces several of the technologies which employ data provided through the SRA. The results of these technologies have been available on synthetic data for a long time already and are commonly applied in video games. Their use for the modification of real captured data however is novel and demands several other technological and algorithmic developments, as the novel acquisition hardware introduced in Chapter 1.2, a common representation (see Chapter 2.1) and algorithms to merge real and synthetic content as

(a) Captured Color: Pool Table



(b) Depth Map for Pool Table

(c) Computer Generated Object



(d) Combined Scene from Captured and Generated Data

Figure 4.1: Captured and Generated Data Combined

presented in Chapter 2.2.

4.1.1 Adding Synthetic Content

Adding synthetic objects to a scene becomes trivial using the Scene Representation presented in 2.1. In the SRA both captured and generated content have spatial positions assigned that define the spatial relation to each other. In the most simplistic case, where captured content is described as an image plane in 3D space, a synthetic object can either be in front or behind the image plane, but also arbitrarily intersecting. With additional information that is available for the captured data the scenario becomes more interesting. With spatial information (e.g. depth data) for the captured color information other objects can be positioned accurately in a scene and interact with the captured data. Lighting information and material proporties further enhance the immersive placement of a synthetic object in a scene.

Figure 4.1 gives an example of how synthetic objects can be addded to scenes. The pool table was captured in one of the first shootings of the SCENE research project [24] by the Motion Scene Camera (see Chapter 1.2.3), which delivers color and depth information. Lighting information and material prop-

Figure 4.2: Recoloring a Spatially and Temporally Consistent Object

erties were extracted from the scene with a 360° camera, a Point Grey Ladybug. The bottle is a computer generated object. In the SRA the computer generated object can easily be positioned on the pool table, and lighting information captured from the environment can be used to render the whole scene as a single composition, where captured and generated data interact seamlessly.

4.1.2 Object Recoloring

Object recoloring is a processing step that relies on spatially and temporally consistent segmentation information. Conventionally color editing in video footage is performed by manually recoloring pixel by pixel and frame by frame. With segmentation information in the scene representation this approach can be dramatically facilitated. Objects can be selected in space and time, and a color change can be assigned to a certain object over a given time span. Approaches that were researched as part of the SCENE project are presented by Sadek et al. and Jachalsky et al. [86, 81]. Alpha matting techniques allow color adjustments even of complex structures such as hair.

Figure 4.2 gives an example of recoloring an object based on a spatially and temporally consistent segmentation. Here the background has been darkened with respect to the foreground dancers. For object recoloring spatial and temporal consistency of the 2D image or video information is sufficient, which differentiates this approach from object retexturing, which is introduced in the next section.

4.1.3 Object Retexturing

Similar to object recoloring in the visual output, but different in the implementation is object retexturing. In object retexturing a new texture is applied to a 3D object. Challenges are to keep the new texture aligned with the object in presence of complex object motion, such as human movements. Alignment

(a) Texture flows from the Painting onto the Actor

(b) Texture remains aligned to Talent throughout Motion

Figure 4.3: Retexturing a Spatially and Temporally Consistent Object

of textures to surfaces has been researched as part of the SCENE project by Budd et al. [5]. Once 3D objects structures are obtained in a scene and made spatially and temporally consistent, a new texture can be applied to this structure.

Figure 4.3 gives an example of such a retexturing step. Texture from the wall painting is transferred onto the body of the actor. The actor afterward moves with the novel texture applied to his body. Throughout the talent movement the new texture stays physically correctly aligned to the actor, just like he would wear a shirt with the new texture applied to.

4.1.4 Clothing Exchange

Being able to exchange the clothing of a talent synthetically presents a special challenge in the area of retexturing. For clothing exchange a virtual wardrobe is created with different clothing pieces for different talent positions. For clothing exchange a skeleton is fitted to the talent object in a scene. The new piece of clothing is than fitted to this skeleton position, and upon actor motion the virtual clothing is dynamically adjusted accordingly. This is especially difficult as physically incorrect cloth motion such as wrong wrinkles or drapery are easily detected by a human observer. This approach has been researched by Furch et al. as part of the SCENE project and is presented in [25].

Figure 4.4 shows the talent in a scene before and after clothing exchange. The jeans jacket the actor is wearing after the synthetic clothing exchange comes from a virtual wardrobe which contains pieces of clothing in different actor positions, but is specifically adjusted to the position a talent takes in a scene.

(a) Talent with original clothing (b) Talent with exchanged clothing

Figure 4.4: Exchanging a Talents Clothing in Post-production

Figure 4.5: Relighting a Scene

4.1.5 Scene Relighting

Synthetically relighting captured scenes is a difficult problem that was approached as part of the SCENE Research Project. A key problem is to detect light sources in the original scene and extract these light sources, leaving an unlit space behind. In addition to light source detection the light reflectance on materials needs to be measured and transferred into material properties. If these steps can be achieved, a scene can be synthetically relighted. Haber et al. have developed and described this approach [33].

Figure 4.5 visualizes an implementation of relighting in the SCENE project. A virtual light, represented by a bright light source in the upper right, emits light onto the unlit scene. The direction of the new light source is clearly visible on the 3D model of the talent standing in the scene.

4.1.6 Pose Modification

Pose modification of 3D models is already a common practice in 3D editing tools. Enabling pose modification for captured video, however, requires a consistent 3D model of the captured object, which implies the need of spatially

(a) Talent in original pose (b) Talent in modified pose

Figure 4.6: Temporally Consistent Modification of a Talents Pose

and temporally consistent object information. While talent pose modifications on visual footage is almost impossible, this kind of processing can be enabled and very much facilitated with the multidimensional consistent data. Ideally, an artist modifies a pose in a 3D editing tool in a single or very few key-frames only. A goal is to transfer the modified key-frame pose to the remaining talent motion, in order to create a seamless interaction with the environment both spatially and temporally. As part of the SCENE project this problem was researched by Neophytou et al. [66].

An application of this pose modification is shown in Figure 4.6. While capturing the video footage the actor was touching a position above the monitor. In some later stage the movie director decided that the actor was supposed to touch the monitor instead. Traditionally this decision requires to reassemble the set, reorder the actors and crew, and shoot the scene again. With the pose modification technique a directors decision as stated can be implemented synthetically at low cost.

4.2 Rendering Mixed Scenes

A central topic for the realism achieved with synthetic data is rendering. For content coming from several sources, like computer generated data and captured image and video content, rendering all data with synthetic camera settings applying to all content is an important aspect of realistic appearance. Therefore, in parallel to the SRA described in Chapter 2 a renderer was developed. This renderer visualizes all content contained in the SRA, independent of the content source, and comprehensively applies defined camera effects.

Camera effects have been a vital part of image and movie productions since image and video acquisition were established at the beginning of the 19th century. At that time camera effects were motivated by physical constraints.

The light requirements of the photo plate enforced either long exposure times which cause motion blur or large apertures which lead to a shallow depth of field. Photographers and movie directors have since then learned to use those physical constraints for artistic means. Motion blurs and a focal plane can be employed to underline the impact of a motive.

However, throughout the last years, image and video acquisition has improved tremendously. High-speed cameras can capture up to 100.000 frames per second[1], with several gigapixels the spatial resolution of novel cameras exceed the resolution of the human retina when viewed at normal viewing distance[2] and the light sensitivity of image sensors constantly increases[3]. Camera technology can therefore capture data by far exceeding the requirements of human perception.

At the same time, by employing camera effects data is already at acquisition time deteriorated. This increases the difficulty of many post-processing steps like content segmentation, or even makes their implementation impossible.

This chapter focuses on using data captured by available data acquisition techniques (see Chapter 1.2) and synthetically generated content for synthetic reproduction of camera effects. These camera effects can be synthetically reproduced at a high level of realism, which enables the usage of undisturbed, high quality data for post-processing steps. Furthermore, synthetically reproduced camera effects are not limited to physical constraints any more. Thus, focal planes not perpendicular to the optical axis or even arbitrarily shaped become possible, and motion blur is not necessarily restricted to the real motion path and speed any more.

As explained above cameras are physically constrained by the amount of light they need to gather at the image sensor. This amount of light can be adjusted either by increasing the exposure time, lens aperture or light in the scene. The amount of light in a scene is usually a directors decision before and during capture. However, Chapter 4.1.5 has explained that synthetic relighting of a scene as a post-processing step is also possible. This Chapter focuses on the settings which are directly linked to the camera, which are aperture and exposure, with the visible results of lens blur and motion blur. As lens blur, or depth of field, depends on the distance of objects to the camera, 3D scene information is a vital ingredient in the synthetization of camera effects. For motion blur a motion path coming from a time-consistent segmentation, as

[1]S-Mize EM - Highspeed Products by AOS Technologies AG, `http://www.aostechnologies.com/high-speed-imaging/products-high-speed/s-mize-em/`

[2]Darpa shows off 1.8-gigapixel surveillance drone, `http://www.extremetech.com/extreme/146909-darpa-shows-off-1-8-gigapixel-surveillance-drone/`

[3]Canon develops 35 mm full-frame cmos sensor for video capture, `http://phys.org/news/2013-03-canon-mm-full-frame-cmos-sensor.html`

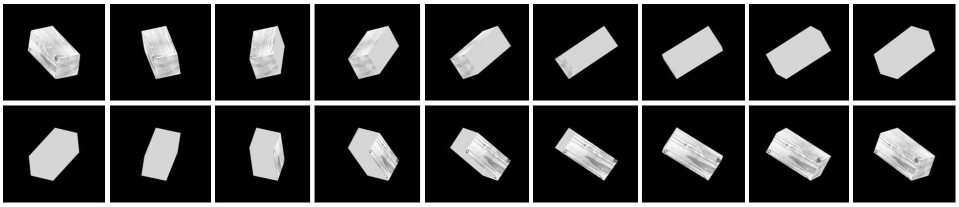


Figure 4.7: Free View Point Rendering of 3D Object

introduced in Chapter 2.2 is essential.

The Motion Scene Camera (introduced in Chapter 1.2.3) provides a good source for 3D scene data which can be fed into the SRA. The implementation of both, motion blur and depth of field effect, is known to the literature (e.g. [78, 46]) for synthetic content. However, only the recently developed technology to capture 3D information, provide frame rates exceeding the human requirements by several orders and track objects over time allow the realistic synthetization of motion blur and depth of field in post-production for captured content.

4.2.1 Free Viewpoint Rendering

One of the core differences between virtual worlds and captured video is that viewers can determine their viewpoint freely in virtual worlds. If 3D structures are available in a scene, either through depth acquisition hardware such as the Motion Scene Camera introduced in Chapter 1.2, or by adding computer generated content to a scene, the viewpoint can also be freely chosen for real or composited content. The Scene Representation allows arbitrary camera movements, which are - from a physical standpoint - constrained only by occlusions in the captured data. For computer generated data and objects that have been captured from multiple directions there are no limits to free camera motion.

Figure 4.7 shows a wooden block which was captured from one side only, including the depth information corresponding to the color information. The object can be embedded in the Scene Representation Architecture and encoded model based, as introduced in the previous Chapter 3. The primitive fitting of the model based encoding scheme fits a cuboid to the three observed block sides. The hidden textures are unknown, therefore approximated by the mean color of the other object sides. This leads to both, a very storage efficient and very flexible object that enables additional processing steps and consumer flexibility in a visual content production chain.

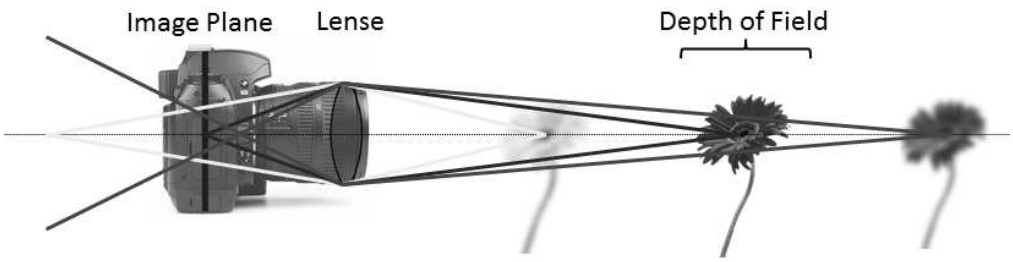


Figure 4.8: Illustration of the Depth of Field Effect

4.2.2 Depth of Field

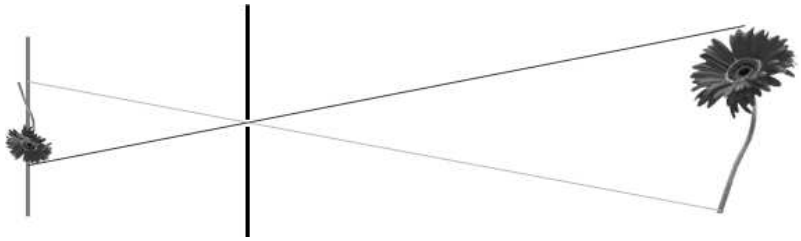
Depth of field describes the distance range from a camera which appears sharp in an image. In front or behind this depth of field may be a blurred region. As the transition from smooth to blurred is continuous, camera manufacturers have defined the depth of field to be the range which is blurred to less than 0.01 inches on a printout enlarged to 8x10 inches. However, in reality human perception can perceive a blur of 0.003 inches already, which sets the acceptable blur a point in an image may have [60].

Figure 4.8 visualizes the depth of field. The optical system of a camera can focus the image only at a certain distance, the so called focal distance, such that it appears in a single spot on the image plane. Points at all other distances necessarily appear as circles (or other shapes influenced by the camera optics), causing a blur.

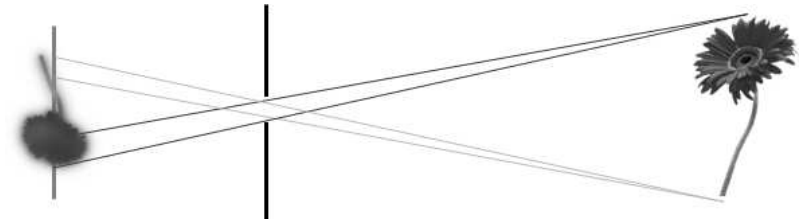
The Physics Behind Depth of Field

The most primitive idea of a camera is a pinhole camera. A pinhole camera does not have any lenses, but gathers all light through a single pinhole, which is theoretically infinitesimal. For such a camera each light ray coming from an object in a scene can take only a single path into the camera through this infinitesimal pinhole (see Figure 4.9a). Therefore such a perfect pinhole camera creates an infinite depth of field. At the same time, however, the amount of light traveling through such an infinitesimal hole is infinitesimal, therefore no sensor will be able to detect it and the image will remain black.

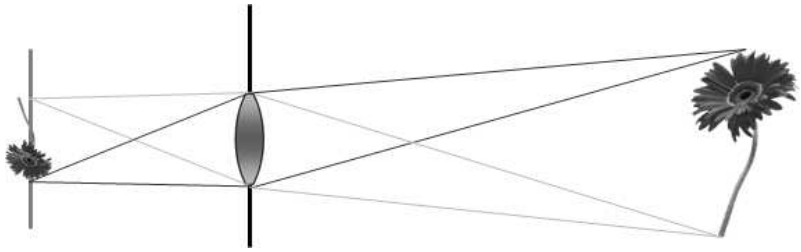
In order to allow a sufficient amount of light to travel through the pinhole, this hole needs to be enlarged. A large hole, however, allows light rays to travel multiple paths and cause blur, as shown in see Figure 4.9b. A lens or multiple lenses assure that light rays coming from a single point of an object are also focused to a single point of the image plane (see Figure 4.9c). On the downside, light rays attenuated in a single point coming from a single point out



(a) Pinhole Camera with infinitesimal hole



(b) Pinhole Camera with larger hole



(c) Lens attenuating rays from the focal plane

Figure 4.9: Camera Models

of the scene need to have a fixed distance. This means that an optical system by default has a focal plane in which objects are sharp, and the remaining range which is blurred.

Synthesizing Depth of Field

The depth of field an optical system produces mainly depends on lens parameters. Engineers have spend decades optimizing lenses to be able to focus on a wide range and minimize depth of field blur. In [73] Pharr and Fernando approximate the size of the circle of confusion C of a system with aperture A, focal length F, focal plane P, object distance O and image distance I by

$$C = \left| A \cdot \frac{F \cdot (P - D)}{D \cdot (P - F)} \right| \tag{4.1}$$

Figure 4.10 illustrates the parameters necessary to calculate the size of the circle of confusion. If the aperture size is set to $A = 1$, focal length $F = 10$

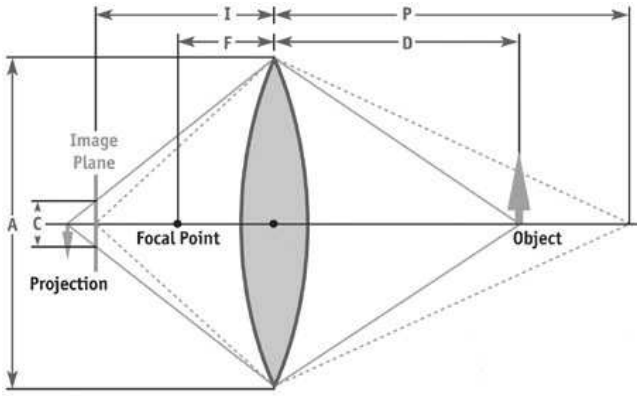


Figure 4.10: Parameters contributing to the Cirlce of Confusion [73]

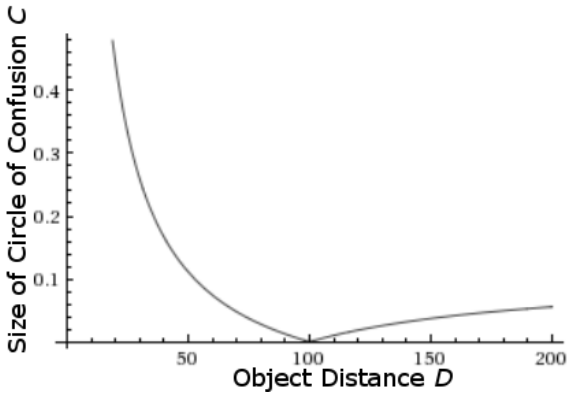


Figure 4.11: Size of the Circle of Confusion C with respect to Object Distance D

and focal plane $P = 100$ a relation between the size of the circle of confusion C and the object distance D can be calculated. This relation is plotted in Figure 4.11. Note that the size of the circle of confusion approaches $C = 0$ as the object distance D approaches the focal plane $P = 100$. Noteworthy is the observation that the circle of confusion does grow neither linearly with the object distance nor symmetric to the focal plane. Much rather, the circle of confusion has a hyperbolic falloff, which causes objects close to the camera to be blurred far more than objects at a comparable distance behind the focal plane.

Small circles of confusion can be approximated by circles, but for larger circles of confusion lens characteristics play an important role. A whole artistic branch is dedicated to finding the most suitable or most beautiful lens blur effect, the so called bokeh. Synthesizing these different artistic lens bokehs is a

fully different path of research with sets on top of the synthetization of camera effects described here, see for example the work by Hach et al. [34] or Wu et al. [110].

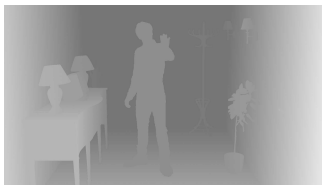
In order to calculate the circle of confusion in addition to a color input image the distance of each pixel to the camera (depth) as well as the camera parameters need to be known. The implementation employed here takes as input data acquired with the Motion Scene Camera [35], which comprises color and depth information. In addition, a synthetic camera is designed with an arbitrary aperture setting. The distance of the focal plane is defined per pixel and stored as a bitmap, accompanying the color and depth information. While for realistic camera settings this seems quite redundant as the whole image can be expected to be at approximately the same focal plane, this implementation opens the door to a whole new set of possibilities, as explained in Section 4.2.4. Figure 4.12 shows the inputs to our approach for synthetic depth of field, which are the RGB color information 4.12a, the measured depth 4.12b and an arbitrary focal plane 4.12c. Here the focal plane is horizontally tilted: the foreground is sharp on the right side of the scene, whereas the background is sharp on the left side and the talent in the middle is in focus as well.

Difference to Reality

A core difference between the synthetic recreation of camera effects and their real counterparts is that synthetic data is always discrete while reality is continuous. To resemble reality the synthetic circles of confusion need to be rendered as superpositions of many blurred circles up to the calculated size. According to [73] the number of circles of confusion which need to be blended should exceed the size of the largest circle of confusion divided by 4 to yield a result without visible artifacts. Higher numbers of blended circles of confusion can reduce artifacts arbitrarily at the cost of complexity.

A second difference to reality can not be overcome by additional computational effort. As shown in Figure 4.13 a physical circle of confusion accumulates light rays coming from *behind* the objects in direct line of sight. With a physical shallow depth-of-field effect a camera can therefore look behind objects. This information is lost when an image is captured with almost infinite depth-of-field and can not be recovered from a single camera view. Light field cameras such as presented by Ng, Ren et al. [67] can overcome this problem, as lens arrays acquire information from multiple viewpoints. If full 3D scene reconstructions, as acquired in [24], exist, light rays from objects hidden behind other objects in focus can be used as well for fully realistic reconstructions of the depth-of-field effect.

(a) Color Input



(b) Depth Input

(c) Focus Input

(d) Rendered Depth-of-Field Blur

Figure 4.12: Inputs and Rendering for Synthetic Depth-of-Field

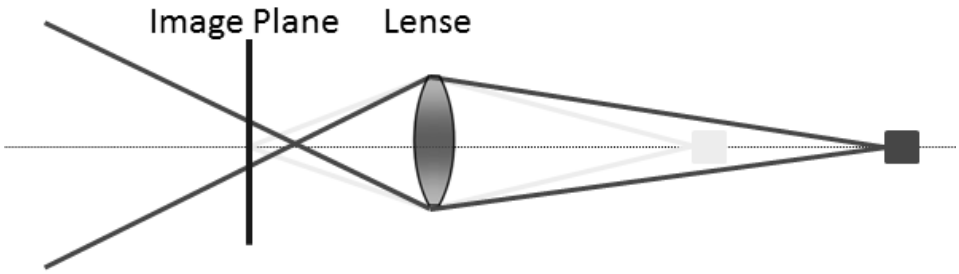


Figure 4.13: The Depth of Field Effect Allows to Look Behind Objects

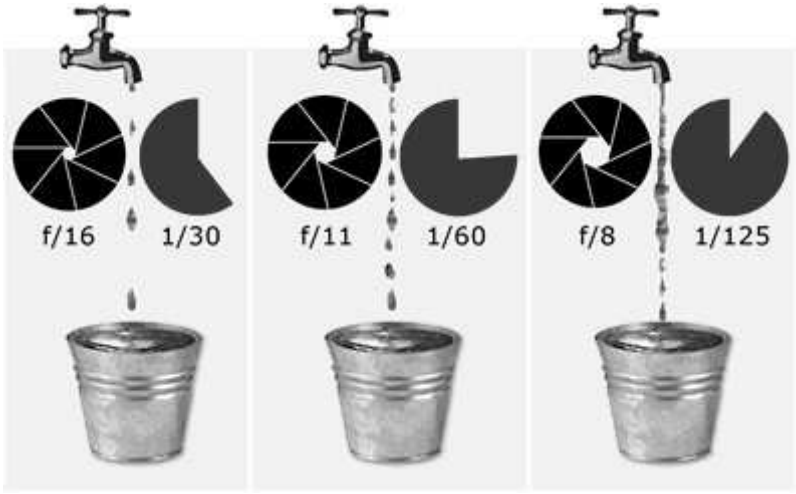


Figure 4.14: Relation between Exposure Time and Lens Opening [12]

4.2.3 Motion Blur

Next to the lens opening a second factor influences the amount of light that can reach the image sensor of a camera. This second factor is the exposure time. The longer light is allowed to travel through the lens, the more light reaches the image sensor and can compensate a small lens opening. At the downside, however, objects moving during this exposure time are necessarily captured on several distinct points on the image plane, causing a blur. The relation between lens opening and exposure time is illustrated in Figure 4.14. Curtin uses a water faucet to represent the lens opening, a water jet representing a light ray and a bucket representing the light sensor. If the bucket captures a certain amount of water (or the light sensor requires a certain amount of light) this amount can either be gathered by a small faucet opening (small lens opening) over a long time (exposure) or by a large faucet opening (large aperture) over a short time [12].

The Physics Behind Motion Blur

In order to maximize the depth of field the lens opening (or camera aperture) needs to be minimal. Using the example of the water faucet given above, in order to fill a bucket the exposure time needs to be prolonged. As soon as content in front of the camera moves, it is projected onto a different position on the image sensor. Small variations in this spatial difference are still gathered by the same bucket of the camera sensor. However, as soon as the projection of an object appears on a different pixel to to spatial motion is perceived as

motion blur.

Synthesizing Motion Blur

Synthesizing motion blur is not a novel idea. In 1983 Potmesil and Chakravarty have already introduced the idea to synthesize motion blur for computer generated contend and presented a thorough mathematical analysis [78]. The core element of their analysis of the effect of object motion and camera shutter is as follows. Potmesil and Chakravarty assume that a moving object can be stopped at time t, and the image of this object at time t is $in(t)$. Then the captured image *out* for an exposure time interval $[0, T]$ is

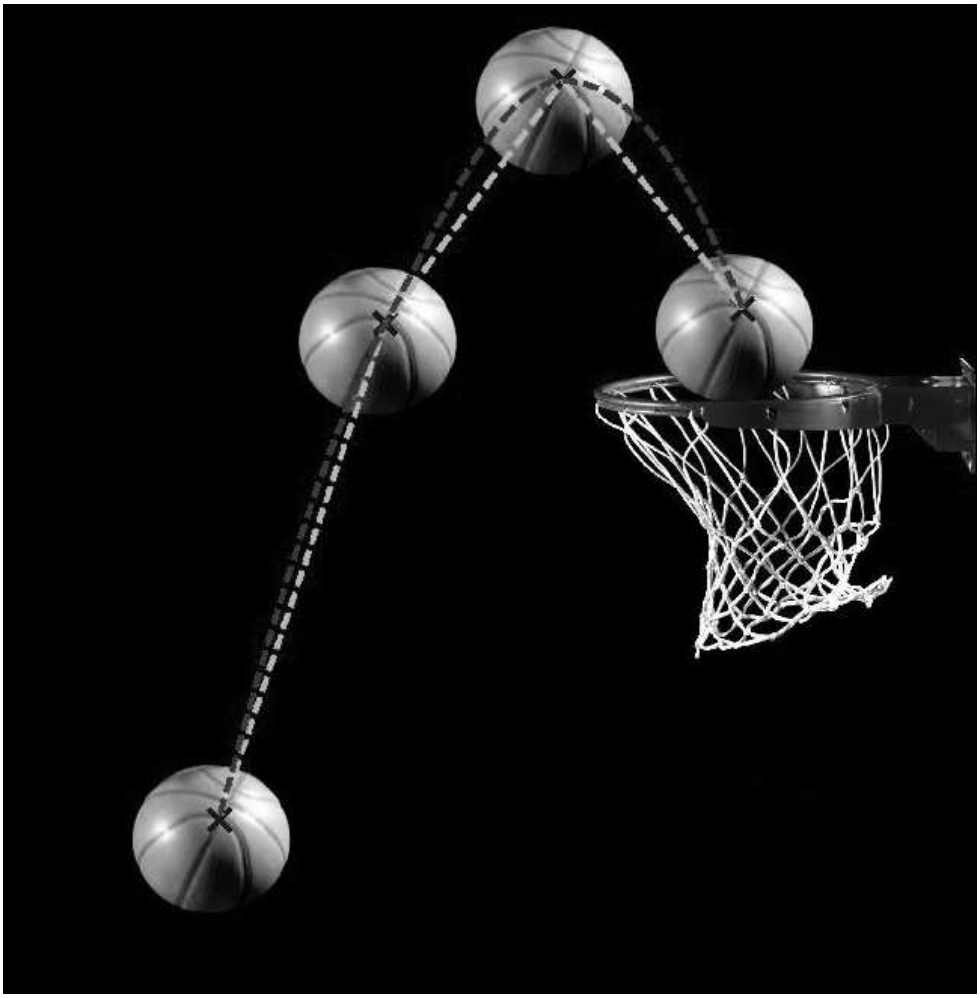
$$out = \int_0^T in(t)dt \tag{4.2}$$

This integral can, however, only be solved if input images $in(t)$ for all time instances t are available. In reality, a camera will have captured a few instances only. If the samples are distributed densely enough (e.g. they are coming from a camera with several thousand frames per second[4] a simple linear merge of all the frames in the time interval is sufficient. In most scenarios, however, not as many sample frames are available. Therefore the motion path needs to be interpolated. By using temporally consistent object segmentation and tracking (e.g. Reso, Jachalsky et al. [81]) motion vectors can be assigned to pixels. Object positions can be interpolated along these motion vectors. Additionally, if several frames are available, object trajectories can be interpolated using cubic interpolation instead of linear interpolation, which creates a smoother appearance of the motion path. Figure 4.15a shows four instances of a basketball captured along with linear motion trajectories (light) and spline interpolated motion trajectories (dark). In Figure 4.15 visual results from the different approaches to create motion blur are presented, 4.15b shows the simple blend of all four frame instances, 4.15c depicts the linear interpolation of the motion and 4.15d presents the cubic interpolation of the object trajectory.

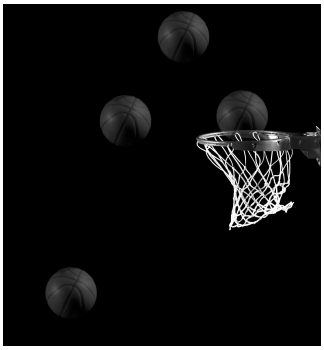
Difference to Reality

When synthesizing motion blur from discretized data the original information cannot be fully reconstructed. This is due to the fact that the image sensor needs to be read out (or for systems with chemical plates the photo plate needs to be replaced). Independent of how many frames can be captured per second this causes gaps in between the acquired data. Whenever objects

[4] S-Mize EM - Highspeed Products by AOS Technologies AG, `http://www.aostechnologies.com/high-speed-imaging/products-high-speed/s-mize-em/`,



(a) Four Instances of a Basketball in Motion, the linear and cubic interpolation of its trajectory



(b) Blending of consecutive frames into motion blur

(c) Linearly interpolated trajectory for motion blur

(d) Cubicly interpolated trajectory for motion blur

Figure 4.15: Motion Blur is Created from Several Instances of a Moving Object

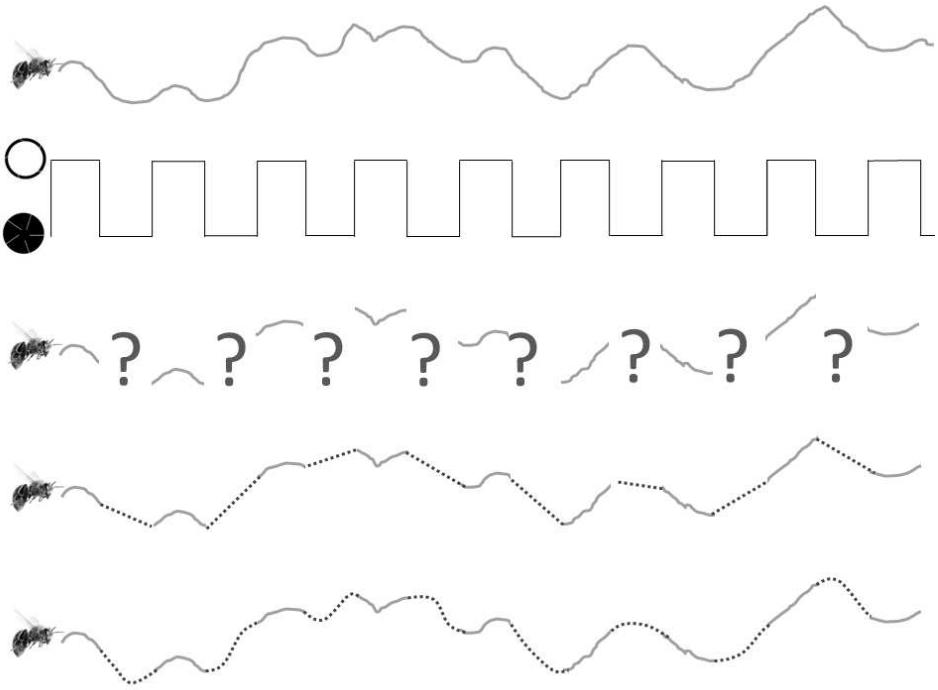


Figure 4.16: Separation into several frames causes loss of information - The unpredictable motion path of a bee (shown in the top row) is captured whenever the camera lens is open (shown in the second row). This leads to unconnected fragments of the motion path (shown in the third row). The two last rows present a linear interpolation for the missing flight path and a cubic interpolation for the missing information.

do not follow known curves which can be algorithmically reconstructed it is impossible to fill these gaps. This missing information is visualized in Figure 4.16. For a bee flying some random path given at the top and the capturing times shown below the full motion path remains unknown. Both a linear and a cubic approximation can fill the gap, but cannot reproduce all the random motion decisions the bee takes during its flight.

4.2.4 Making the impossible possible

As soon as camera effects are synthesized, physical constraints do not remain limiting factors any more. While tilted focal planes, as shown in Figure 4.12d, can be physically implemented with so called “tilt-shift lenses” [36], introducing several focal planes, shaping the focal plane arbitrarily or applying a semantic blur (focusing on a semantically meaningful part) are novel applica-

Figure 4.17: Examples for Synthetic Depth of Field

tions only possible with synthetic depth of field rendering. Figure 4.17 presents an example for the three cases mentioned. The upper images show a white line at the position of the focal plane, the lower images are rendered with this focal plane. From left to right two focal planes at the front and back of the room, a curved focal plane and semantic blur putting focus on hands and faces are employed for rendering. As the amount of blur is defined by any number of focal planes which can be synthetically arbitrarily positioned and shaped, this adds artistic freedom previously unavailable for captured data.

Similar to physically impossible depth of field blur also motion blur can be influenced synthetically beyond physics. As physical motion blur (for a fixed aperture) is determined by the direction and the speed of motion, both of these parameters can arbitrarily be adjusted for segmented objects in full 3D scenes. Based on the Scene Representation Architecture introduced in Chapter 2.1 it has been shown that several artistic manipulations are possible. Slow objects can be blurred with a faster motion blur than other objects to emphasize the speed of an object. Directed motion blurs also direct the view of the observer and can be employed to guide a viewer in a scene.

All of a sudden the toolbox of artistic effects and footage modifications a movie director can employ has been increased dramatically, based on the new Scene Representation and the paradigm change of movie production envisioned in the SCENE research project [24]. Whether and to what extend these effects will be used is from now on mainly an issue of simple and intuitive interfaces as well as an artistic concern of the creative part of a visual production pipeline.

Chapter 5

Image Quality Metrics

Whenever visual content is presented, the question quality of the presented material is an important issue. "Quality" usually means the fitness for a certain use case. In Chapter 2.2 for example the suitability of content for image segmentation is discussed extensively. When images are presented to human observers, quality usually refers to the subjective perception of visual content and the human judgment whether this content has a high level of realism, contains few artifacts, little noise, ... summarizing how good image content reflects the expectations of the human observer.

The question of image quality has been around for decades, and several answers to this question have been developed. The most common answer to the quality of an image is the Mean Opinion Score (MOS) or Differential Mean Opinion Score (DMOS). The approach of obtaining a MOS is to query a large number of people to evaluate the quality of given data, and average their assessments. The MOS has first been used to evaluate the quality of communication channels, and therefor their specification can be found in the ITU-T Recommendation P.800 of the International Telecommunication Union[1]. Here a communication channel is classified in five categories from 1 = excellent to 5 = poor by asking the users to rate the "difficulty in talking or hearing over the connection"[1] in percent.

While collecting user opinions gives the most credible results for the perceived quality, conducting an evaluation to obtain a MOS or DMOS is expansive with respect to time and money. Thus, getting a MOS is usually infeasible for real world applications. Algorithmic approaches which evaluate the quality of a given content are necessary. These algorithms have the sole requirement of coming up with the same score that humans would assign to content; thus

[1] ITU-T Recommendation P.800 "Methods for Subjective Determination of Transmission Quality"

allowing to predict the MOS without having to conduct expensive surveys.

Algorithmic image quality metrics are generally classified in three groups. If algorithms try to evaluate the image quality based purely on a test image, such metrics are called no-reference metrics. If certain image features are given, but still no reference is available, these metrics are called partial-reference metrics, and finally, if a full reference image is available, metrics operating on this information are known as full-reference metrics. While no-reference metrics can be used most comprehensively, they are also the most challenging ones. The largest research advances have been made in the area of full-reference metrics, were for a several decades the SNR was the only usable metric. In 1997 Sarnoff et al. designed a vision model to algorithmically determine and rate the "just noticeable difference" in images and videos, which became known as the Sarnoff Metric [53]. Seven years later a second fundamental step was made with Wang et al. developing a method based on image structures, the Structural Similarity index, which conforms a lot better to MOS evaluations [103]. Another seven years later Mantiuk et al. enhanced the idea of structural similarity to cope with more difficult environments, for example changing lighting conditions [57].

Existing metrics already offer decent solutions to approximate a subjectively generated MOS algorithmically for "classical" image errors. These classical errors include most forms of distortions introduced by either capturing, coding or transmission, like random noise, illumination change or blocking artefacts. When dealing with data reduction through prediction of real content by synthetically generated objects (see Chapter 3) a new problem was perceived: Slight shifts of synthetically generated objects can cause an offset of textures and have a dramatic effect on the PSNR, while the change remains largely unnoticed by human observers. An example is shown in Figure 5.1. The lady in pictures 5.1a and 5.1b looks very much the same, it requires close obersvation to note that the order of the stripes on her shirt has changed. Even if a human observer does notice, this does neither alter the realism nor the perceived quality of the image, thus both images would probably receive a good quality ranking. An algorithm, however, would notice the large difference in image content, as shown in Figure 5.1c, thus assign a poor quality score to the found image. For comparison, Figure 5.1d shows an image with the same PSNR as 5.1b, this time by introducing random noise.

This difference between algorithmic output and subjective perception has been noticed in general for scenarios which synthetically combine content from several data sources, like augmented or virtual reality applications. This chapter is motivated by the described deficiency.

(a) Original (b) Wrong Shirt (c) Error Map (d) Random Noise

Figure 5.1: Example of failing PSNR for Image Quality Assessment ©Collage Vintage, Bette Franke

5.1 Datasets

In order to develop an algorithm which can assess the quality of an image and come to the same result as a subjectively assigned score a dataset is needed to test this algorithm on. Only if an algorithm can perform well on a large set of images presenting different content and deteriorated by different sources of error such an algorithm becomes usable in real life. Therefore, the creation of suitable datasets for image quality assessment metrics has been an important step in the development of image quality metrics.

5.1.1 Datasets with classical Errors

The first widely used database was developed by Sheikh et al. in 2004 and, being created at the Laboratory for Image and Video Engineering, became known as the LIVE Image Quality Assessment Database. A second release was published in 2005, featuring a variety of images and image distortions [91]. 29 input images were distorted by compression artifacts (JPEG2000 and JPEG compression with different quality levels), white noise of varying standard deviations, Gaussian blur with kernels of varying size and and artifacts created by a fast fading Rayleigh channel for data transmission. With these distortions roughly 1000 different images were created. These were evaluated in seven experiments by human observers, who were supposed to rank the images on a linear scale with 5 segments labeled “Bad”, “Poor”, “Fair”, “Good” and “Excellent”. For the final MOS every image was rated by 20 to 30 human observers.

In 2008 Ponomarenko et al. set out to create a database going beyond the contents of the LIVE Database [76]. They created the Tampere Image Database (TID) which was updated in 2013, now including 3000 distorted images created from 25 reference images [75]. The 24 different distortion types

Table 5.1: Comparison of LIVE to TID Database

	LIVE	TID	ESPL
# of reference images	29	25	25
# of distortions	5	24	5
# of test images	1000	3000	525
# of assessments	30.000	250.000	25.000
# of assessments per image	20-30	350	50

include different kinds of additive noise, quantization-, compression- and transmission errors, blurs, intensity shifts, contrast and saturation changes. All images were evaluated by over 970 observers in more than 500.000 comparisons, leading to more than 1.000.000 total evaluations, or roughly 350 assessments per image. Similar to the LIVE Database, TID is available for download with reference images, test images, MOS for the test images and a detailed description about occurring errors.

With the growing demand for image quality assessments of synthetic image contents the ESPL Synthetic Image Database was created by Kundu et al. in 2014 and updated in 2015 [44]. Other than LIVE and TID the ESPL database is not based on photos as reference images, but uses synthetic images from various synthetic sources such as computer animated movies or video games. The ESPL database covers image distortions comparable to the distortions introduced in LIVE and TID, which are High Frequency Noise, Interpolation-, Banding and Ringing-Artifacts, Gaussian Blur and JPEG compression artifacts. The ESPL synthetic database features 25 synthetic reference images distorted by five different error sources. The resulting 525 test images were assessed in roughly 25.000 assessments.

Table 5.1 gives a direct comparison of the main characteristics between LIVE Database and TID ans ESPL Synthetic Image Database. TID and LIVE database both use images from the Kodak Photo CD [22] which was made available by the Eastman Kodak Company for unrestricted use. This makes these two databases comparable with respect to their reference images. A thumbnail preview of the contained images is given in Figure 5.2. Despite the obvious superiority of TID especially with respect to the number of assessments per image, the LIVE Database with twice as many citations remains an important database for research conducted on Image Quality. All three mentioned databases are comparable with respect to the image distortions they induce on their reference images. In order to allow comparisons among existing and novel databases it is advisable to employ the set of common image errors also for the creation of a novel image database.

Figure 5.2: Images of the Kodak Lossless True Color Image Suite used for LIVE Database and TID

All three, the LIVE Database as well as TID and ESPL database, generate several of their image distortions with MATLAB[2]. The theoretical models for errors and their implementation in MATLAB are described in the following, as these error implementations are used for a novel enhanced database, which is introduced in Section 5.1.2.

JPEG Compression Artifacts

JPEG compression is described extensively in the literature. A thorough description is, for example, provided by Wallace [102]. The JPEG algorithm for lossy image compression operates on image blocks of 8×8 pixels. In the context of compression artifacts the quantization of the 64 frequency components calculated by a Discrete Cosine Transform is the major contributor. This quantization is influenced by multiplication of the frequency components by a percentage: Multiplication by $q = 100\%$ does not influence the quantized values, but multiplication with a low value causes many of the frequency components to be rounded to zero after quantization [100]. A quantization matrix $Q_{k,l}$ with $0 \leq k, l < 8$ is modified by quality parameter q as follows:

$$Q\prime_{k,l} = \begin{cases} (Q_{k,l} \cdot (5000/Q_S) + 50)/100 & \text{for} \quad q < 50 \\ (Q_{k,l} \cdot (200 - 2 \cdot Q_S) + 50)/100 & \text{for} \quad q \geq 50 \end{cases} \tag{5.1}$$

JPEG coding artifacts can best be produced by running the JPEG encoding process and observation of the results. MATLAB offers a function to write images in many desired file formats, among them JPEG. When writing to a JPEG file MATLAB can take the image quality q as an additional parameter.

[2]MATLAB, Version 8.4.0 (R2014b). The MathWorks Inc., Natrick, Massachusetts, 2014

Figure 5.3: Blocking Artifacts created by JPEG Compression of Quality $q = 0$

Listing 5.1 gives the source code necessary to create the desired image distortion. An example of such JPEG artefacts for the extreme case of quality $q = 0$ is given in Figure 5.3.

Listing 5.1: Generating JPEG Compression Artifacts

```
q = 5; % quality
imwrite(img,'tmp.jpg','Quality',q);
out = imread('tmp.jpg');
delete('tmp.jpg');
```

JPEG2000 Compression Artifacts

JPEG2000 is described by ITU-T Recommendation T.809[3] and - similar to JPEG - discussed in several publications [8, 93, 79]. The most significant differences between JPEG and JPEG2000 are the following. JPEG applies an 8×8 discrete cosine transform on image macroblocks of size 16×16, while JPEG2000 uses a wavelet transform and partitions the image into macroblocks in the wavelet domain, thus reducing blocking artifacts significantly and achieving higher coding gain and scalable coding, which is a main design criterion for JPEG2000 [47]. While compression artifacts in JPEG mostly result from the quantization step (see previous Section), in JPEG2000 the bitstream assembler subsequent to domain transformation and quantization is the main source of artifacts.

Similar to artifacts created for JPEG, JPEG2000 coding artifacts can also best be modeled by encoding image data with a JPEG2000 encoder for different bit-rates and reconstructing the encoded image. MATLAB considers target data reduction rates r in its JPEG2000 encoder, as shown in Listing 5.2. Significant reduction rates are necessary to result in visible artifacts. For

[3]ITU-T Recommendation T.809 "JPEG2000 Image Coding System"

Figure 5.4: Compression Artifacts created by JPEG2000 Compression of Compression Rate $r = 500$

example, Figure 5.4 was reduced with a target compression rate of $r = 500$ resulting in a size of only $0.05 bits/pixel$.

Listing 5.2: Generating JPEG2000 Compression Artifacts

```
r = 500; % compression rate
imwrite(img,'tmp.jp2','CompressionRatio',r);
out = imread('tmp.jp2');
delete('tmp.jp2');
```

White Gaussian Noise

White noise is noise that occurs uniformly over all frequencies, which means it has a constant power spectral density. Gaussian noise is noise that can be statistically described by a probability density function $p(x)$ of a normal distribution

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{5.2}$$

with mean μ and standard deviation σ. White Gaussian Noise is therefore noise with constant power spectral density and distribution according to Equation 5.2. The *imnoise*-function which MATLAB provides to generate such noise (see Listing 5.3) internally generates a matrix of normally distributed noise N and calculates

$$out = in + \sqrt{\sigma^2} \cdot N + \mu \tag{5.3}$$

where σ^2 is the variance, which is given as an input parameter to the *imnoise*-function. Figure 5.5 illustrates white Gaussian noise, here with a mean of $\mu = 0$ and variance $\sigma^2 = 0.2$.

Figure 5.5: Average White Gaussian Noise with Mean $\mu = 0$ and Variance $\sigma^2 = 0.2$

Listing 5.3: Modeling White Gaussian Noise

```
m = 0; % mean
v = 0.05; % variance
out = imnoise(img, 'gaussian', m, v);
```

In 1928 the physicists H. Nyquist and J. Johnson published the theoretical background confirming that "thermal agitation of electric charge in conductors" [69] and "thermal agitation of electricity in conductors" [41] can be modeled as white noise, which became known as Johnson-Nyquist noise. As thermal noise is omnipresent this is an important image distortion model.

Gaussian Blur

Image blur is an image distortion often caused by objects being out of focus, a too shallow depth of field or either moving camera or moving object during the exposure time. Neither of these causes is modeled correctly using Gaussian Blur only (see Chapter 4.2 on content rendering), but Gaussian Blur is a simply model for general blurring. Blurring is achieved by filtering an image with a 2D Gaussian kernel. Extending the 1D Gaussian distribution from Equation 5.2 to 2D it is

$$p(x, y) = \frac{1}{\sigma\sqrt{2\pi}} \cdot exp\left(\frac{(x - \mu_x)^2 + (x - \mu_y)^2}{2\sigma^2}\right) \tag{5.4}$$

where $[\mu_x, \mu_y]$ is the mean (or center) of the 2D Gaussian bell. For blur filter design a mean offset is (usually) not desired, therefore $\mu_x = \mu_y = 0$. In MATLAB filters can be generated with the *fspecial*-function, which for the Gaussian kernel only requires the filter size and the standard deviation σ of the normal distribution. A 2D Gaussian blurring kernel of 3×3 pixels with $\sigma = 0.5$ is given in Figure 5.6. Blurring of an image with a filtering kernel is generated by convolution of the image with the filter.

0.011	0.084	0.011
0.084	0.620	0.084
0.011	0.084	0.011

Figure 5.6: 2D Gaussian Kernel of size 3×3 with $\sigma = 0.5$

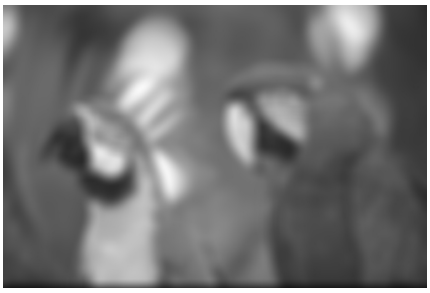


Figure 5.7: Gaussian Blur with $s = 30 \times 30$ Filter Size and Standard Deviation $\sigma = 10$

The MATLAB implementation for generating image distortions employing Gaussian blur is given in Listing 5.4. A visualization of this distortion for filter size $s = 30 \times 30$ and standard deviation $\sigma = 10$ is shown in Figure 5.7.

Listing 5.4: Modeling Gaussian Blur

```
h = [3 3]; % kernel size
s = 0.5; % standard deviation
filter = fspecial('gaussian', h, s);
out = imfilter(img, filter,'replicate','same');
```

As mentioned above, LIVE database, TID and ESPL database, contain several distorted images with the errors described here. By employing the same errors to novel databases evaluations become comparable and research becomes transferable.

5.1.2 Dataset with synthetic Errors

With the merge of real and synthetic data novel sources of image distortions occur. Most important, next to errors introduced at acquisition time (like blurs

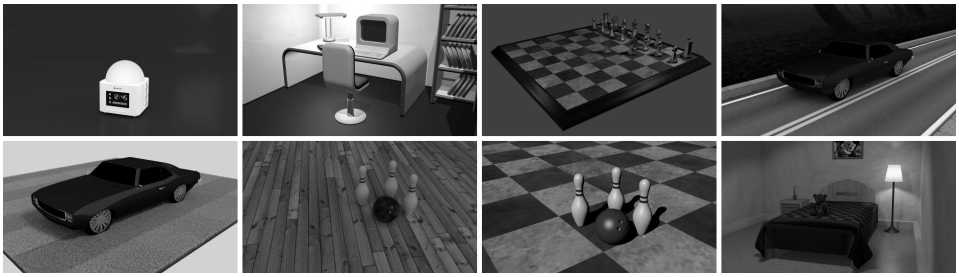


Figure 5.8: Images with Synthetic Content for Generation of Synthetic Errors

or image noise) errors with respect to scene composition and scene rendering influence the resulting visual quality. In augmented reality scenarios scenes are composited by adding synthetic objects to real scenes. While the real content remains unchanged, the synthetic object needs to be scaled, aligned and positioned in a scene. Scaling, aligning and positioning are affine transformations in 3D space, and errors can affect all of these transformations. For a dataset including errors which are likely to occur in scenes of real and synthetic content erroneous affine transformations need to be simulated. This can best be done on content that is synthetically available.

Figure 5.8 shows eight scenes with fully synthetic content. Central elements of these scenes, as e.g. the car, bowling ball or alarm clock, can be modified by affine 3D transformations to simulate possible scene composition errors.

In computer graphics affine transformations are usually expressed in homogeneous coordinates. Homogeneous coordinates were introduced by the German mathematician Möbius in 1827 [61] and relate to Cartesian coordinates by projection. This means that any n-dimensional object in Cartesian coordinates requires an $n+1$-dimensional representation in Homogeneous coordinates. For objects in 3D space this leads to transformation matrices expressed in 4D. The advantage of Homogeneous coordinates compared to Cartesian coordinates is the ability to express arbitrary Cartesian points - even at infinity - with finite Homogeneous coordinates.

The creation of synthetic errors requires a rendering environment for the synthetic data. Blender[4] provides 3D compositing and editing options far exceeding the requirements for synthetic error simulations. Affine transformations in Blender can be scripted in Python[5], thus allowing an automatic and standardized implementation of image distortions by 3D transformation

[4]Blender - A free and open source 3D Creation Suite. `https://www.blender.org/`, accessed 20-Apr-2016

[5]Python - A universal, interpreted higher programming language, `https://www.python.org/`, accessed 20-Apr-2016

errors.

Object Scaling

Adequate object scaling is necessary to integrate computer generated objects into a real scene. Scaling of objects is achieved by moving the object vertices in 3D according to a scaling factor. This scaling factor can be freely chosen along the object axes, which leads to three independent scaling factors s_x, s_y and s_z along the x-, y- and z-axis respectively. A vertex position p is than scaled to vertex position p' by multiplication with the scaling matrix S

$$\vec{p'} = S \cdot \vec{p} \tag{5.5}$$

where S is defined by

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

Object scaling implemented in Python to be used in a rendering software is given by the lines of code given in Listing 5.5. With $s_x = 0.8$, $s_y = 1.0$ and $s_z = 0.9$ the car shown in Figure 5.8 is scaled to the version shown in Figure 5.9.

Listing 5.5: Object Scaling Error

```
x = object.scale[0]
y = object.scale[1]
z = object.scale[2]
# 3 random variables for scaling in x-,y- and z-direction
sx = x * math.fabs(random.gauss(1,0.33))
sy = y * math.fabs(random.gauss(1,0.33))
sz = z * math.fabs(random.gauss(1,0.33))
# scaling with random variables
object.scale=((sx,sy,sz))
# file rendering
bpy.ops.render.render( write_still=True )
```

The classical image distortions introduced above (see Section 5.1.1) match single error values to quality measures deduced by arbitrary error metrics, as defined in the following sections. Such a mapping is not trivial for transformations in 3D space, as not a single, but three parameters contribute to the resulting error. A single parameter to describe the scaling difference of an

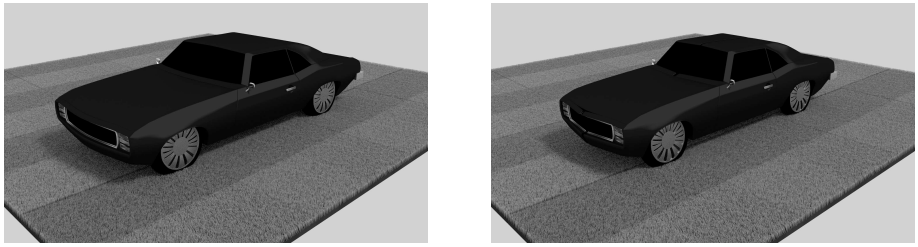


Figure 5.9: Left: Original Image, Right: Car Scaled with $s_x = 0.8$, $s_y = 1.0$ and $s_z = 0.9$

object is the variance of the scale, assuming the original object scale is 1. This leads to the definition of the scaling error s, with

$$s = \sum_{k=x,y,z} (s_k - 1)^2 \tag{5.7}$$

Object Translation

The position of an object in a scene is a crucial factor for the realistic appearance of the rendered scene. Deviations from the correct position can have several effects: objects can merge into other scene objects, they can loose contact from surfaces or shift on a surface. Spatial translation in 3D can be expressed by the translation summands t_x, t_y and t_z, which cause translations along the x-, y- and z-axis respectively. A vertex position p is therefore shifted to position p' by multiplication with the translation matrix T:

$$\vec{p'} = T \cdot \vec{p} \tag{5.8}$$

where T is defined as

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.9}$$

The Python implementation of object translation that cvan be used by the Blender rendering software is given in Listing 5.6. With $t_x = 0.0$, $t_y = -0.1$ and $t_z = 0.1$ the car shown in Figure 5.8 is translated to the version shown in Figure 5.10.

Listing 5.6: Object Translation Error

```
# b as vector of object dimensions
```

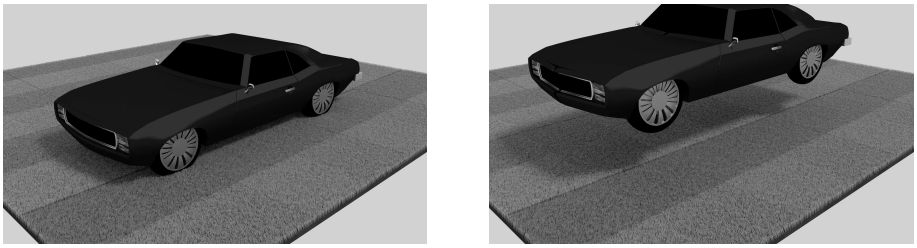


Figure 5.10: Left: Original Image, Right: Car Translated with $t_x = 0.0$, $t_y = -0.1$ and $t_z = 0.1$

```
b=bpy.data.objects["OBJECT"].dimensions
# x,y,z as dimensions in x-,y- and z- direction
x=b[0]
y=b[1]
z=b[2]
# 3 values for absolute translation
tx = x * random.gauss(0,0.33)
ty = y * random.gauss(0,0.33)
tz = z * random.gauss(0,0.33)
# translation in x-,y- and z-direction
bpy.ops.transform.translate(value=(tx, ty, tz))
# file rendering
bpy.ops.render.render( write_still=True )
```

With a similar reasoning as given for the single parameter representing the object scale error a single parameter representation for the object translation error is desired. This single representation is given by the Euclidean vector describing the object translation in 3D space. This leads to the single translation error parameter t defined as

$$t = \sqrt{t_x^2 + t_y^2 + t_z^2} \tag{5.10}$$

Object Rotation

Most objects are not rotationally invariant. For these objects it is crucial to not only determine their position and scale, but also their alignment with the environment. Alignment is possible with respect to the three coordinate axes. Different from scaling and translation the rotation needs to be defined per axis in a single matrix. For rotational angles α_x, α_y and α_z around x-, y- and z-axis respectively the rotation matrices R_x, R_y and R_z are defined as

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_x) & -sin(\alpha_x) & 0 \\ 0 & sin(\alpha_x) & cos(\alpha_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.11}$$

$$R_y = \begin{bmatrix} cos(\alpha_y) & 0 & -sin(\alpha_y) & 0 \\ 0 & 1 & 0 & 0 \\ sin(\alpha_y) & 0 & cos(\alpha_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.12}$$

and

$$R_z = \begin{bmatrix} cos(\alpha_z) & -sin(\alpha_z) & 0 & 0 \\ sin(\alpha_z) & cos(\alpha_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.13}$$

These rotation matrices can me multiplied to a vector $\vec{p}$ defining an object vertex to rotate this vertex to a new position defined by vector $\vec{p'}$:

$$\vec{p'} = R_x \cdot R_y \cdot R_z \cdot \vec{p} \tag{5.14}$$

It is important to note that matrix multiplications are not commutative. The order of applying shift, scaling and rotations are important, even the order of rotations around different axis result in differently aligned results. The Python code shown in Listing 5.7 which was employed for the creation of the data base with synthetic errors calculates the total rotation as $R = R_z \cdot R_y \cdot R_x$ according to the above stated Equation 5.14. Figure 5.11 shows the car from Figure 5.8 rotated by $\alpha_x = 0.1$, $\alpha_y = 0.2$ and $\alpha_z = -0.1$.

Listing 5.7: Object Rotation Error

```
# rotate over x-axis
bpy.ops.transform.rotate(value = random.gauss(0,60), axis =
    (1, 0, 0))
# rotate over y-axis
bpy.ops.transform.rotate(value = random.gauss(0,60), axis =
    (0, 1, 0))
# rotate over z-axis
bpy.ops.transform.rotate(value = random.gauss(0,60), axis =
    (0, 0, 1))
# file rendering
bpy.ops.render.render( write_still=True )
```

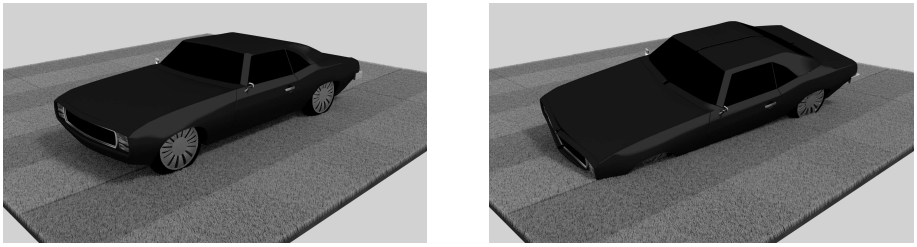


Figure 5.11: Left: Original Image, Right: Car Rotated by $\alpha_x = 0.1$, $\alpha_y = 0.2$ and $\alpha_z = -0.1$

As for translation and scaling it is desirable to express the total rotation of the object by a single parameter. For an arbitrary rotation matrix R the angle of rotation α around a free rotation axis $\vec{n}$ is given by

$$\alpha = arccos\left(\frac{tr(R) - 1}{2}\right) \tag{5.15}$$

where $tr(R)$ is the trace of the rotation matrix R. If R is defined as the product of the rotation matrices R_x, R_y and R_z as given above, it is

$$\begin{aligned} tr(R) = & cos(\alpha_x) \cdot cos(\alpha_y) + \\ & cos(\alpha_x) \cdot cos(\alpha_z) + \\ & cos(\alpha_y) \cdot cos(\alpha_z) + \\ & sin(\alpha_x) \cdot sin(\alpha_y) \cdot sin(\alpha_z) \end{aligned} \tag{5.16}$$

With α_x, α_y and α_z as arbitrarily chosen rotation angles around the corresponding Euclidean axes the single total angle of rotation α can easily be calculated.

A novel database was composed, using the 8 reference images shown in Figure 5.8. These 8 reference images were distorted by the four described classical errors (Gaussian Noise, Gaussian Blur, JPEG and JPEG2000 coding artifacts) as well as the three new image distortions which are due to pre-rendering errors (object translation, scaling and rotation). For each of the reference images and distortion types 30 different distortion levels were generated, leading to a total number of 1680 test images. All images are rendered at a resolution of 1920×1080 pixels, thus matching the quality requirements current visual systems pose. These test images are accompanied by descriptions which contain the name of the reference image as well as the parameter choices that have created the corresponding test image. For research it is important to have subjective evaluations of the perceived image quality. The experiments leading to subjective quality scores are described in the following Section 5.2. Afterward

different approaches to evaluate visual content objectively are presented, with a suggested approach for the described database in Section 5.5.3.

5.2 Subjective Metrics

When evaluating the quality of any kind of data, subjective metrics aim to assign the quality value to the given information that conforms to the average consumer opinion. This "average consumer opinion" or "mean opinion" is expressed by the "mean opinion score", short MOS. For a given number of N information assessors with their individual opinion o_i for $1 \leq i \leq N$ the MOS is calculated as

$$MOS = \frac{1}{N} \sum_{i=1}^{N} o_i \tag{5.17}$$

However, different subjects tend to evaluate the same information differently, based on the information content. Given, for example, two subjects, of whom one prefers mountains, the second the seaside. If both subjects are asked to evaluate images of mountains and seaside which have undergone certain image distortions, the first subject will tend to rate images showing mountains higher than seaside pictures and vice versa, thus hiding the real effects under investigation. The "Differential Mean Opinion Score" or short DMOS aims to reduce such effects. This is done by presenting not only distorted, but also the original information to subjects. With the individual opinion of this reference information r_i and N, o_i and i as defined above the DMOS is calculated as

$$DMOS = \frac{1}{N} \sum_{i=1}^{N} (o_i - r_i) \tag{5.18}$$

Different ways to obtain assessor scores for information have been used and researched. Mantiuk et al. mention four major methods and compare their effectiveness [58]. These four methods are the so called "Single Stimulus", "Double Stimulus", "Forced Choice", and "Similarity Judgments". Additionally, in 2002 Keelan introduced the "Quality Ruler" method with the goal to overcome some of the negative effects observed in single stimulus methods. All five methods differ with respect to the required observations and the quality of their results. A common part of all subjective studies however are the different quality scores. Different implementations have been tested from 5 to 100 different quality levels of which the assessors where able to choose. A second common attribute of all studies is the timing of the individual stimuli. Especially with media that has no inherent time like images, this information can

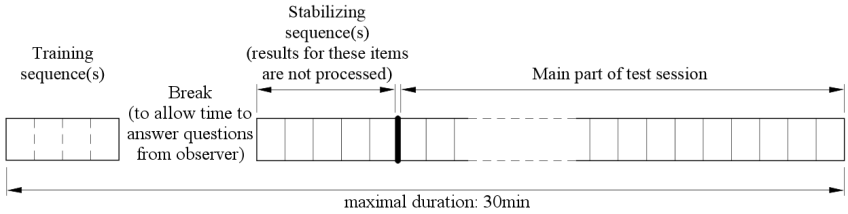


Figure 5.12: Structure of a Test Session for Subjective Quality Assessments[7]

be exposed to the observer for any amount of time, which might again lead to different quality opinions.

According to Section 2.7 of Recommendation ITU-R BT.500-11 a test session should not last more than half an hour to prevent fatigue effects. Additionally, each session should start with detailed experimental instructions and training sequence, followed by a break in which sufficient time for questions concerning the experiment is given. Afterward a series of experiments are run with the purpose of stabilizing the experimental outcomes. This stabilizing sequence is not evaluated. Subsequent to the stabilizing sequence the main experiment starts, of which opinion scores are recorded and further processed[6]. The general structure of a test session is given in Figure 5.12.

The following subsections introduce the mentioned subjective quality assessment methods. Subsection 5.2.6 introduces findings about requirements to the quality scale, and Subsection 5.2.7 presents observations regarding the timing of stimuli.

5.2.1 Single Stimulus

The Single Stimulus method, also known as Absolute Categorical Rating, is - due to its simplicity - one of the most widely used approaches for subjective evaluations of media data. Formal descriptions of the method can be found in Section 6.1 and 6.2 of Recommendation ITU-T P.910[8] and Section 6 of Recommendation ITU-R BT.500-11[9]. Evaluations are generated by showing a single image for a given time t and a subsequent query for the observed image quality. ITU-R BT.500-11[10] suggests to use a sequence of three displays per

[6]ITU-R Recommendation BT.500-11 “Methodology for the Subjective Assessment of the Quality of Television Pictures”

[8]ITU-T Recommendation P.910 “Subjective Video Quality Assessment Methods for Multimedia Applications”

[9]ITU-R Recommendation BT.500-11 “Methodology for the Subjective Assessment of the Quality of Television Pictures”

[10]ITU-R Recommendation BT.500-11 “Methodology for the Subjective Assessment of the Quality of Television Pictures”

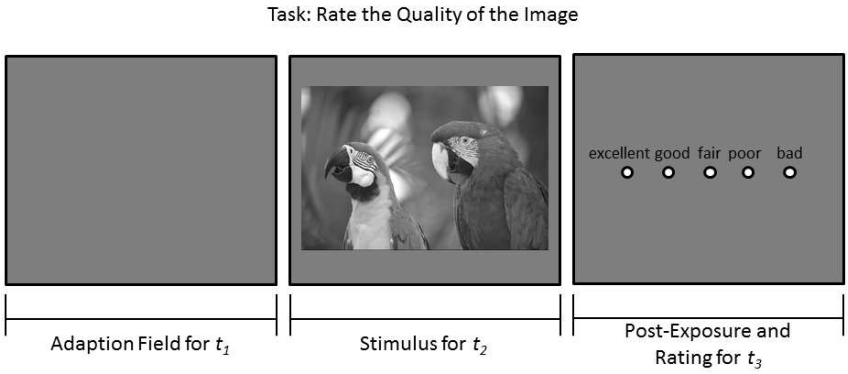


Figure 5.13: Structure of a Single Stimulus Experiment

evaluation: first a mid-gray adaption field for $t_1 = 3sec$, second the stimulus for $t_2 = 10sec$, and third a mid-gray post-exposure field for another $t_3 = 10sec$. The rating of the assessor may be collected either during the stimulus or the post-exposure display. However, as detailed in Subsection 5.2.7, durations of the different displays can be optimized to achieve a higher throughput. The general layout of a single stimulus experiment is given in Figure 5.13. With one evaluated image per experiment the evaluation of n test images plus one reference image requires $n + 1$ of such single stimulus experiments to evaluate the full set of images.

5.2.2 Double Stimulus

Double Stimulus is a method, in which not one, but two images are given for evaluation per experiment. As outlined in Section 6.4 of Recommendation ITU-T P.910[11] and Section 4 of Recommendation ITU-R BT.500-11[12] two stimuli of the set of reference and test stimuli are shown in random order after each other. ITU-T P.910 suggests grey adoption displays of $t_1 = 2sec$ and to present each stimulus for $t_2 = 10sec$. The second stimulus should be followed by another post-exposure display for $t_3 = 10sec$, during which the assessor is asked to evaluate both stimuli. Equivalent to the Single Stimulus method the display times can be optimized to achieve higher throughput (see Subsection 5.2.7. The layout of a double stimulus experiment is given in Figure 5.14. The double stimulus setup presents two images per trail, but each stimulus can be combined with every other stimulus of the set. For n test images plus

[11]ITU-T Recommendation P.910 "Subjective Video Quality Assessment Methods for Multimedia Applications"

[12]ITU-R Recommendation BT.500-11 "Methodology for the Subjective Assessment of the Quality of Television Pictures"

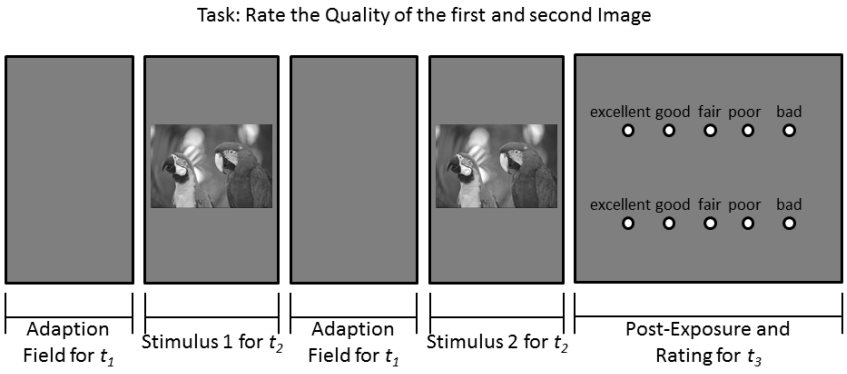


Figure 5.14: Structure of a Double Stimulus Experiment

one reference image the total number of experimental setups necessary to be evaluated therefore is $\binom{(n+1)}{2} = \frac{n\cdot(n+1)}{2}$. ITU-T P.910 furthermore recommends that the stimuli should not only be presented in every combination, but to consider the order as well. If each combination of stimuli is shown twice, the second time in reverse order, the total number of necessary experiments increases to $n \cdot (n+1)$ for n test images plus one reference image.

5.2.3 Forced Choice

Mantiuk et al. describe the Forced Choice method as a popular image assessment method in computer graphics [58]. This method presents two images at the same time, and forces the assessor to select the image of higher quality. A choice is enforced by not allowing the observer to select equal quality. Comparable to the Double Stimulus experiment this method requires comparisons of each pair of images. Here, however, the order of presentation does not matter, as both images are presented at the same time. The total number of necessary experiments to evaluate each possible couple of stimuli is therefore given by $\binom{(n+1)}{2} = \frac{n\cdot(n+1)}{2}$ for n test images and one additional reference image. Figure 5.15 gives the structure of a forced choice experiment, where the time $t_1 = 2sec$ as suggested by ITU-T P.910 for the double stimulus experiment, $t_2 = 10sec$ according to the same assumption for evaluation time, but can also be unlimited.

5.2.4 Similarity Judgment

In Similarity Judgment experiments assessors not only decide for the better of two stimuli but rank the difference. Here, "no difference" is a valid option. According to Mantiuk et al. this method is mostly used in functional measure-

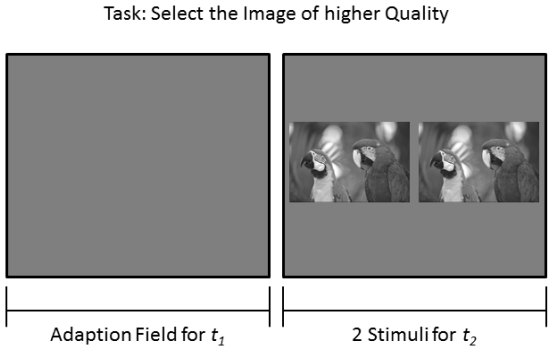


Figure 5.15: Structure of a Forced Choice Experiment

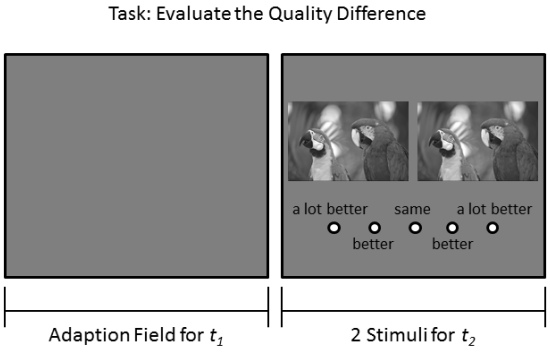


Figure 5.16: Structure of a Similarity Judgment Experiment

ments, relying on relative judgments between stimuli. [58] With two images shown at the same time this experimental setup requires as many trials as the Forced Choice Experiment, $\frac{n \cdot (n+1)}{2}$ for n test images plus a reference image. After one adaption display of $t_1 = 2sec$ the voting time should be scheduled to at least $t_2 = 10sec$ or unconstrained. Figure 5.16 gives the experimental structure of one trial of the Similarity Judgment method.

5.2.5 Quality Ruler

For Quality Ruler experiments a set of reference images with known quality are generated. These images are presented with a test image, and assessors are asked to place the test image at the position of the reference images where the quality is matching. In practice, this is usually implemented by showing a test image and a reference image side by side, and asking the user to decrease the quality of the reference image until the perceived quality of test image and reference image match. [80] The voting time t_2 can therefore require

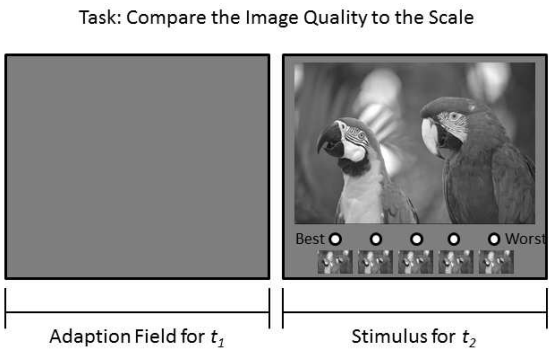


Figure 5.17: Structure of a Quality Ruler Experiment

any amount of time, depending on how many different reference images the observer needs to compare to. Each experiment is started with an adoption display of $t_1 = 2sec$ as suggested in the Double Stimulus recommendation of ITU-T P.910[13]. It is important that the different quality levels of the reference images do not exceed the "just noticeable difference" in order to present a continuous scale to the observer. At the same time, the reference images need to span a wide range of qualities to include all possible perceived qualities of the test images.

Figure 5.17 presents the experimental structure of the Quality Ruler experiment, with the implementation adjustment to show the reference scale not simultaneously, but element-wise next to the test image. As the reference images are known to the assessor and shown at the same time as a scale, this quality ruler setup requires exactly n trials to evaluate n test images.

5.2.6 Quality Scale

Literature agrees to the ITU-T Recommendation P.800[14] that the "mean opinion score" or "differential mean opinion score" should be expressed in the five quality categories, "excellent", "good", "fair", "poor" and "bad". However, the experiments leading to these quality categories can employ very different scales and a lot finer granularity. ITU-T Recommendatin P.910 already introduces four different quality scales[15]. These four different quality scales are

[13]ITU-T Recommendation P.910 "Subjective Video Quality Assessment Methods for Multimedia Applications"

[14]ITU Recommendation P.800 "Methods for Subjective Determination of Transmission Quality"

[15]ITU Recommendation P.910 "Subjective Video Quality Assessment Methods for Multimedia Applications"

given in Figure 5.18 and cover a wide range of possible scales. While the 5 different quality levels of Figure 5.18a directly correspond to the MOS categories, the quality scale given in Figure 5.18b offers additional intermediate grades for quality assessment. This 9-point quality scale can be further extended to an 11-point quality scale (as shown in Figure 5.18c), extending the range from the 9-point scale to both sides by 1. The new numbers 0 and 10 are defined in the recommendation as "the number 10 denotes a quality of reproduction that is perfect faithful to the original" and "the number 0 denotes a quality of reproduction that has no similarity to the original"[16]. These additional quality levels are required as the MOS-score "excellent" was originally defined as "having no difficulty to talk and hear over the connection" and "bad" corresponding to the opposite, that talking and hearing over a given connection was very difficult[17]. Extending this desire for more granularity with respect to the quality scale ultimately leads to a continuous scale, as shown in Figure 5.18d. On such a scale a user can select an arbitrary quality level corresponding to the subjectively observed quality of a stimulus.

Rouse et al. have analyzed the scores given by assessors on a quasi-continuous scale (100 quality levels) [84]. The histogram of the recorded scores is given in Figure 5.19. With over 23% of the assessors directly using one of the five MOS categories and over 42% evaluating stimuli quality by either the MOS categories or their midpoints, it becomes obvious that a continuous scale for quality assessment is superfluous. According to the analysis of Rouse et al. a quality scale with 9 or 11 quality levels - depending on the experimental conditions - suffices fully.

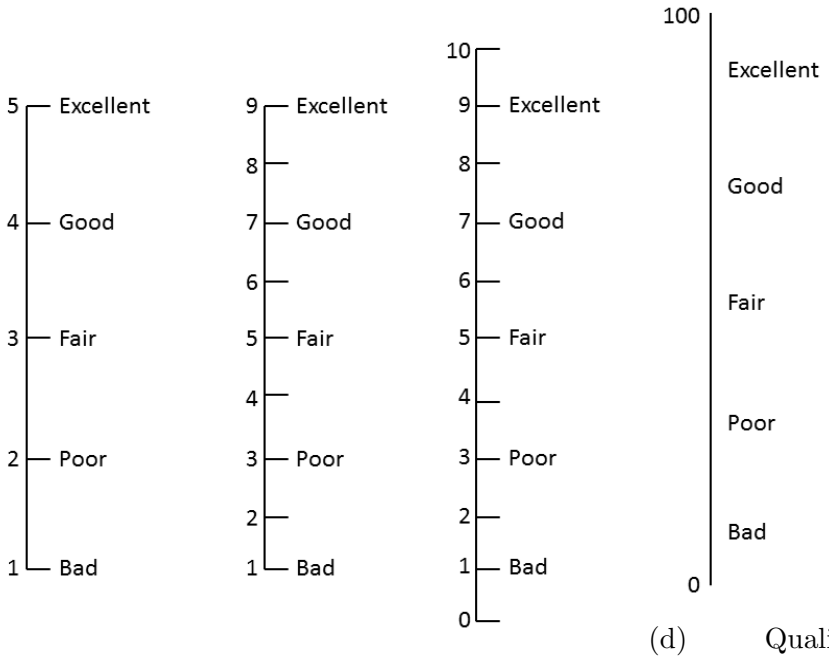
5.2.7 Timings and Durations

While the ITU References agree about the exposure time for stimuli, experiments indicate that exposure times can be optimized to achieve a higher throughput in the experiments. Section 6.1 of Recommendation ITU-T P.910 suggests a presentation time of the stimuli of $t_2 = 10sec$, but also mentions that "the presentation time may be reduced or increased according to the content of the test material"[19]. In [58] Mantiuk et al. present results of a study suggesting that a presentation time of $t_2 = 3sec$ suffices for assessors to rate the quality of still image stimuli. With an adaption display for $t_1 = 2sec$ and

[16]ITU Recommendation P.910 "Subjective Video Quality Assessment Methods for Multimedia Applications"

[17]ITU Recommendation P.800 "Methods for Subjective Determination of Transmission Quality"

[19]ITU-T Recommendation P.910: "Subjective video quality assessment methods for multimedia applications", §6.1



(a) Quality Scale with 5 Quality Levels

(b) Quality Scale with 9 Quality Levels

(c) Quality Scale with 11 Quality Levels

(d) Quality Scale with continuous Quality Levels

Figure 5.18: Quality Scales given in ITU-T Recommendation P.910[18]

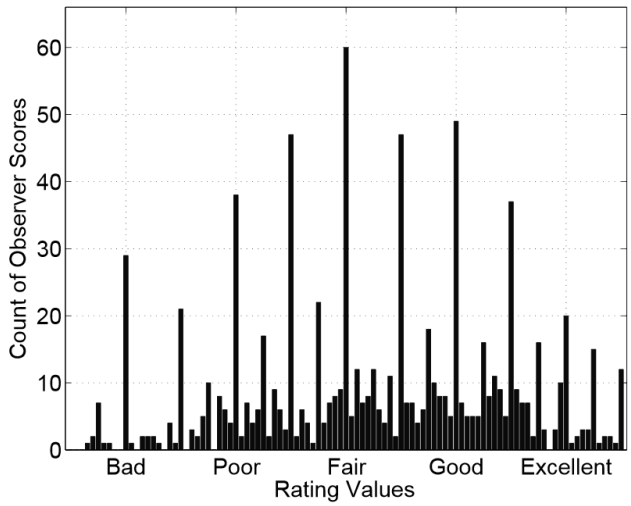


Figure 5.19: Histogram of Raw Assessor Scores on Quasi-Continuous Scale [84]

Table 5.2: Duration of Subjective Evaluation Experiments for n test images

Setup	Expected Duration
Single Stimulus	$(n+1) \cdot 10sec$
Double Stimulus	$n \cdot (n+1) \cdot 15sec$
Forced Choice	$0.5 \cdot n \cdot (n+1) \cdot 7sec$
Similarity Judgment	$0.5 \cdot n \cdot (n+1) \cdot 7sec$
Quality Ruler	$n \cdot 7sec$

an average voting time of $t_3 = 5sec$ the expected duration of an experiment can be calculated. Table 5.2 gives the expected experimental duration to obtain a rating for a set of n test images under different experimental settings. Further more, Recommendation ITU-T P.910 suggests to have images evaluated by at least 15 observers, which needs to be considered in the total duration of an experiment.

5.2.8 Evaluating the Synthetic Image Database

knowing the criteria for subjective tests described in the previous sections a questionnaire was designed. The questionnaire was implemented as a website, with the aim to achieve to major goals. First, the database should be evaluated by many people at different locations and from a range of backgrounds. Local studies would have restricted study participants to a certain background and age, as well as constrained the number of participants. Second, as the same visual media is consumed on a range of devices, from smart phones to wall-size projections, a web-based questionnaire allows to perform evaluations on the devices, that people typically employ to consume media. A fundamental decision needed to be taken concerning the experimental setup. With 1680 test images the time requirements as a function of the number of test images becomes critical. According to Table 5.2 only the Single Stimulus and the Quality Ruler setup are linear with respect to the number of test images. Quality ruler experiments require a predefined images at different quality levels. Creating such quality levels is difficult, especially for the 3D transformations which are dependent on multiple parameters. Therefore a Single Stimulus setup was chosen as a general experimental setup. The outline of the questionnaire is given in Figure 5.20. In the following the individual steps of an experimental session and their design decisions are explained.

An experimental session starts with an introductory text. Here, the purpose of the test as well as the rough structure and time requirements are explained. In addition, a motivation for participants if the experimental session

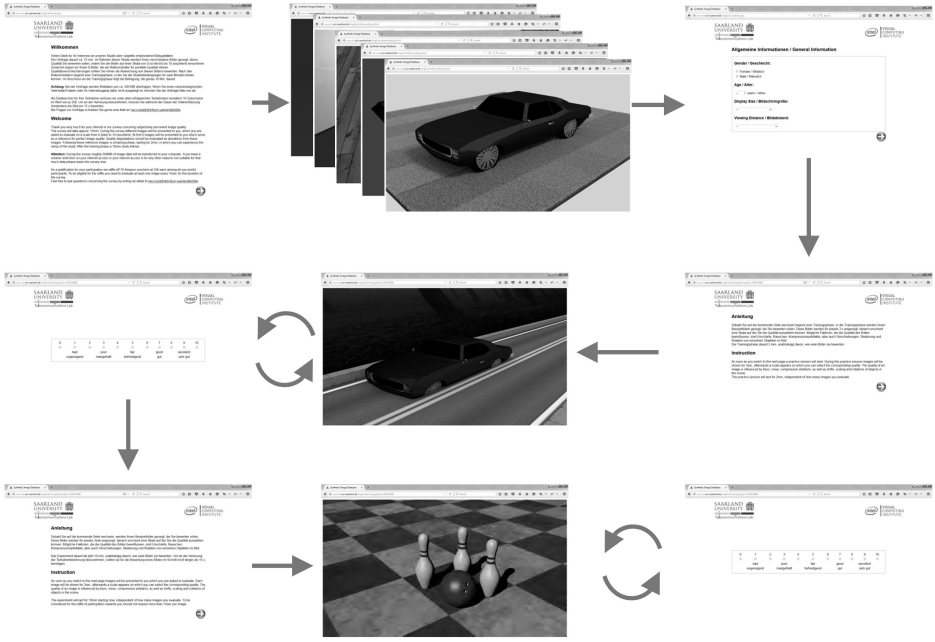


Figure 5.20: Structure of Experiment to Obtain Subjective Evaluations for the Proposed Synthetic Image Database

is concluded successfully is announced.

The second step of the experimental session is the presentation of the reference images. Each image is presented for 5sec and automatically exchanged by another one, until all reference images have been presented to the viewer. All of the synthetic scenes included in the synthetic image database lack the realism of photos. This lack of realism should not be interpreted as a quality degradation by the assessor. Therefore it is important to present the ground truth, undistorted images to the assessor before the evaluation. At the same time, assessors should not compare reference to test image directly. In order to enforce a break and a cognitive distraction the third step of the experimental session is included.

The third step consists of a brief questionnaire querying personal assessor data. This data includes gender, age, display size and viewing distance. In addition, the display resolution is read through the browser. Allowing a wide range of viewing devices comes with a trade-off: Visual artifacts which are easily visible on a large size TV screen might be unnoticeable on a smart phone. Knowing about the viewing distance d, display size s and resolution r the perceived resolution per degree of vision r_{1° can be calculated as

$$r_{1^\circ} = tan(1^\circ) \times d\frac{r}{s} \tag{5.19}$$

The questionnaire is followed by a 2min stabilization session. This stabiliztion session is introduced by an instruction and has the purpose of familiarizing the assessor with the process of image evaluations and the images he can expect. The process is an iteration of test image presentations for 3sec each, followed by a quality scale on which the assessor can select the observed quality level from 11 different quality levels between 0 and 10. More important than the familiarization with this process is the introduction of possible image distortion levels. This prohibits cases were assessors assign the lowest observed quality to an image, but later during the experiment encounter an even worse image quality which cannot be ranked correctly any more, given that the lowest score has already been assigned to an image of better quality. The quality scale which assessors use to rank images is coherent to the findings presented in Section 5.2.6. Evaluations done during this stabilizing session are not recorded.

The stabilizing session is followed by the actual test session. This test session is again introduced by an instruction, announcing the same conditions as observed in the stabilizing session. The main difference is that the test session lasts for 10min, compared to the 2min stabilizing session. During the test session results are recorded. Assessors are requested to rate at least 40 images during the 10min test session, which corresponds to 25sec per image,

or 22sec to make a choice on the quality scale. During the first 40 images all reference images are randomly intermixed. This hidden reference approach has two advantages. First, ranked references allow to calculate the glsdmos (as introduced in Equation 5.18). Second, reference images provide a simple way to validate user input, as reference images should be - over all - ranked with a comparably high quality score. The test session automatically terminates after 10min.

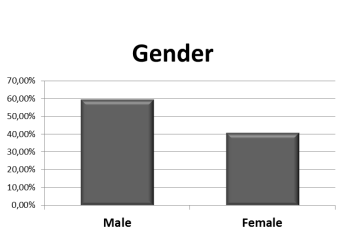
Stabilizing session and test session are timed to exactly 2min and 10min respectively. The presentation of the reference images in the beginning lasts exactly 80sec. With an approximate 1min per instruction page the whole experiment lasts roughly 15min. Even for slow reading assessors the total experimental time stays well below 30min, which is indicated as a critical attention span in the Recommendation ITU-T P.910[20].

The described survey was conducted with roughly 200 participants, leading to about 17.000 image assessments, or on the average 10 assessments per error image. It is important to note that with 30 test images per reference and distortion, about 300 assessments per error type and reference image exist which allow a fine analysis of error and opinion score.

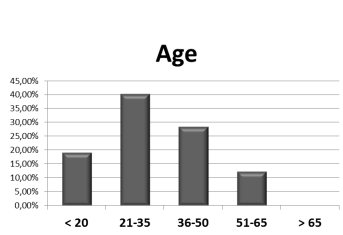
Figure 5.21 gives an overview of some assessor characteristica. Figure 5.21a shows the gender distribution, which indicates by almost 20% more male assessors than female assessors. Figure 5.21b shows the age distribution of assessors, pointing to the majority of assessors between 21 and 35 years. Both, gender and age distribution, can be accredited to the distribution channels of the survey. The survey was announced at university, especially around the male dominated Computer Science and Physics labs, as well as through social media channels of male researchers. These distribution channels led to a dominating group of male university students among the assessors, which is reflected by the gender and age distribution.

Viewing distance and display size, given in Figures 5.21c and 5.21d, are strongly correlated. This is intuitively clear: the larger the display, the larger the viewing distance generally is. Small display devices are usually hand-held, notebooks are positioned either on labs or on a table, but in a distance that users can operate the attached keyboards. Desktop monitors are usually even further away, with a detachable keyboard in reach. Some people seem to operate large screen setups, and sit even further away from those than in traditional desktop scenarios. Projectors or display walls with display diagonals larger than 65″ have not been used for the evaluation of the image database. From display size, viewing distance and the window resolution that was read

[20]ITU Recommendation P.910 “Subjective Video Quality Assessment Methods for Multimedia Applications”

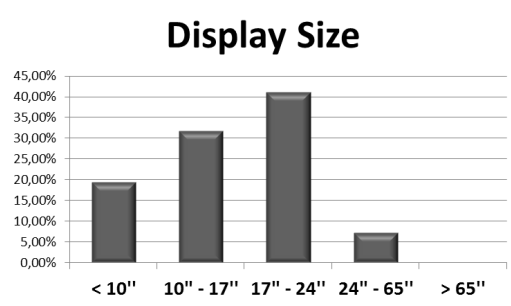


(a) Gender Distribution

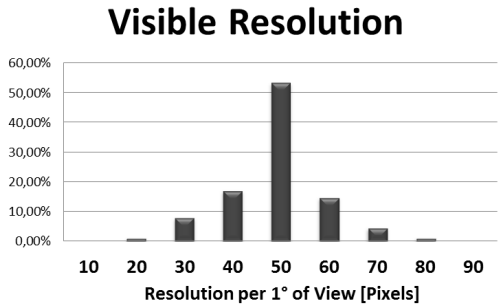


(b) Age Distribution

(c) Viewing Distance Distribution



(d) Display Size Distribution



(e) Resolution Distribution

Figure 5.21: User Statistics from the SID2015 Evaluation

internally during the survey the visible resolution per 1° of vision was calculated with Equation 5.19 and is displayed in Figure 5.21e. This analysis shows that with over 50% the majority of the survey participants have a resolution of $\sim 50\frac{px}{1°}$ of vision, which corresponds to an HD monitor with $1920px$ horizontal resolution at a viewing distance (in cm) that is $3.\bar{3}$ times the diagonal of the display size (in inches). This relation is displayed in Figure 5.22.

The collected evaluations were used to calculate MOS for the individual distorted images. With the obtained MOS and the error parameters as introduced in Section 5.1.2 an 'Ideal MOS' can be computed, which is defined as the best fitting exponential curve between MOS and error parameter. The ideal MOS for the seven different error classes (Gaussian Blur, White Noise, JPEG and JPEG2000 coding artifacts, object scaling, object translation, and object rotation) are shown in Figure 5.23. For visualization the error parameter is normalized to $[0, 1]$. In all cases an increasing error parameter conceptually corresponds to a decreasing visual quality. It is interesting to note that the inclination of the curves indicating the ideal MOS for scaling, translation and rotation is significantly lower than for the classical errors. Considering also the correlation between MOS and ideal MOS (see Table 5.3 and 5.4) it becomes clear that the mapping from three error parameters to a single error parameter and the relation between error parameter and observed quality are problem-

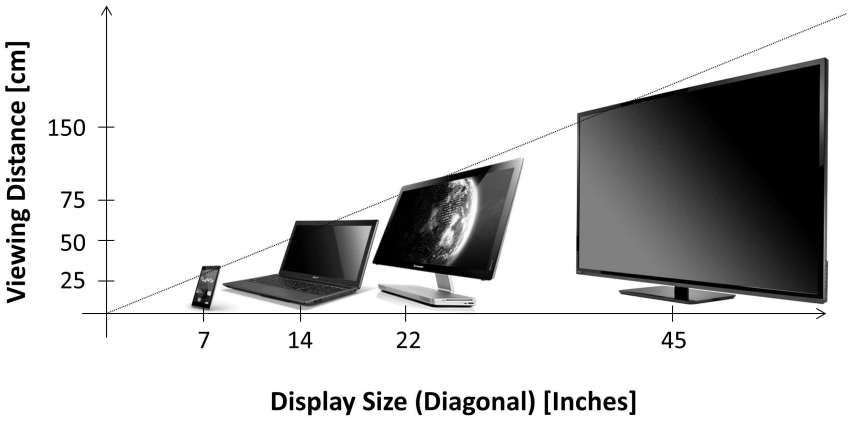


Figure 5.22: Display Size vs. Viewing Distance for HD Display and Visible Resolution of $50px$ per 1° of Vision

atic. First, the same error parameter which points into different directions in 3D space can lead to different subjective evaluations. Exemplarily, an object translated towards the camera only appears slightly scaled, while a translation by the same value into the ground or into the air reduces the perceived reality dramatically. Second, the same error parameter may lead to different perceived qualities depending on the original object size and environment.

For evaluation of the created database and the subjective scores different metrics were calculated on the database, which are introduced later in this work. Correlations between the MOS scores obtained experimentally and the ideal MOS (the least square fit exponential function), PSNR metric (Chapter 5.5.1), SSIM Index and HDR-VDP2 Metric (both Chapter 5.5.2) were calculated. Correlations are calculated for different error classes individually and for all distorted images together. Error classes are JPEG, Noise, Transformation, Classical, and All. The JPEG class includes JPEG and JPEG2000 image compression artifacts. The Noise class contains Gaussian white noise and Gaussian blur. Transformation includes rotations, scaling and translations. The Classical error class is a super-class of JPEG and Noise errors, and the All-error class is a super-class of JPEG, Noise and Transformation errors. Table 5.3 gives Spearman's ρ [96] for the described correlations, Table 5.4 gives Kendall's τ [42] for the same. The outperforming metric with respect to the calculated correlation measure for each error class is marked bold in both tables.

Three main observations can be made based on the analysis of the database and the corresponding MOS-, PSNR-, SSIM and HDR-VDP values. First, the correlation between MOS values and ideal MOS in Tables 5.3 and 5.4

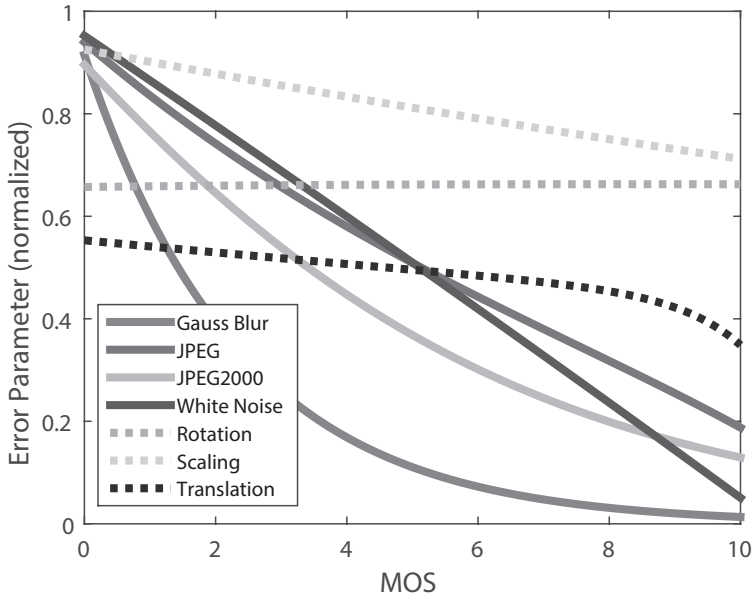


Figure 5.23: Ideal MOS for Different Error Types: Classical Errors shown with Straight Lines, Transformation Errors with Dotted Lines

Table 5.3: Spearman - Correlation between MOS and existing metrics

	JPEG	Noise	Transformation	Classical	All
Ideal MOS	$\rho = 0.84$	$\rho = 0.88$	$\rho = 0.46$	$\rho = 0.81$	$\rho = 0.83$
PSNR	$\boldsymbol{\rho = 0.72}$	$\rho = 0.59$	$\rho = 0.31$	$\boldsymbol{\rho = 0.69}$	$\rho = 0.42$
SSIM	$\rho = 0.69$	$\boldsymbol{\rho = 0.64}$	$\boldsymbol{\rho = 0.36}$	$\rho = 0.67$	$\boldsymbol{\rho = 0.60}$
HDR-VDP 2	$\rho = 0.51$	$\rho = 0.56$	$\rho = 0.24$	$\rho = 0.52$	$\rho = 0.37$

Table 5.4: Kendall - Correlation between MOS and existing metrics

	JPEG	Noise	Transformation	Classical	All
Ideal MOS	$\tau = 0.65$	$\tau = 0.69$	$\tau = 0.32$	$\tau = 0.64$	$\tau = 0.65$
PSNR	$\boldsymbol{\tau = 0.53}$	$\tau = 0.41$	$\tau = 0.21$	$\boldsymbol{\tau = 0.50}$	$\tau = 0.29$
SSIM	$\tau = 0.50$	$\boldsymbol{\tau = 0.47}$	$\boldsymbol{\tau = 0.25}$	$\tau = 0.48$	$\boldsymbol{\tau = 0.41}$
HDR-VDP 2	$\tau = 0.35$	$\tau = 0.41$	$\tau = 0.17$	$\tau = 0.36$	$\tau = 0.25$

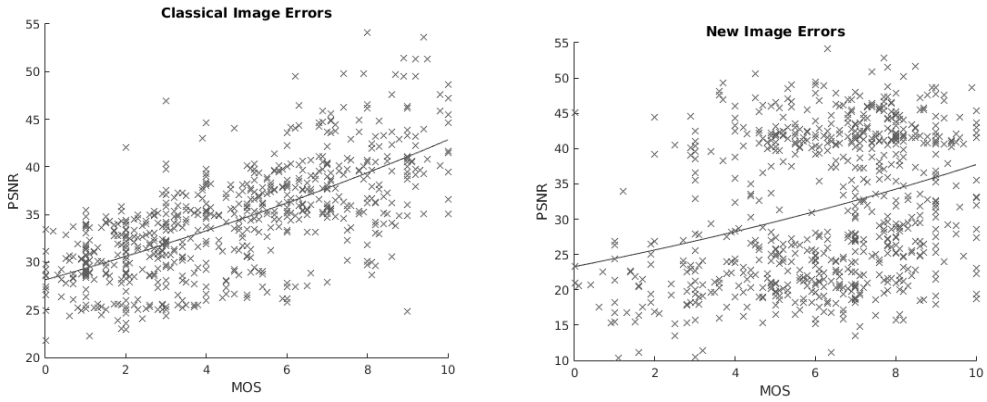


Figure 5.24: PSNR vs MOS for Classical (left) and Novel (right) Image Errors

is significant enough to draw conclusions from the subjective experiments. Second, the correlation between MOS and ideal MOS is significantly smaller for the new transformation errors, which is intuitively clear as the mapping from three error parameters to a single error value does not necessarily correspond to the perceived error. This observation can also be visually confirmed, as the ideal MOS depicted in Figure 5.23 for Rotation, Scaling and Translation (dotted lines) is significantly less dependent on the error parameter as for classical errors (straight lines). Third, also for the analyzed quality metrics the correlation between MOS and computed quality in the correlation tables is significantly smaller for transformation errors than for other error classes. Here, however, no error parameter is used, therefore the cause clearly are insufficient metrics. Figure 5.24 displays this observation for the PSNR metric: a visual evaluation already shows that PSNR and MOS are significantly more correlated for classical errors than for transformation errors.

The Synthetic Image Database SID2015 is available for download[21]. It contains all reference and test images. The test images are accompanied by info-files which contain the corresponding reference image and parameter choice to generate each test image. Info-files are available as Matlab and plain text files. For all images the synthetic sources and a detailed description of the system the images were generated with to allow exact reproduction of all images are provided. Additionally, the MOS scores for all images in a Matlab file are provided. The database is structured as presented in Figure 5.25: for each type of error there exists one folder, containing the distorted images. In addition to the distorted images there is one text file and one MatLab file per folder, both

[21]SSID - Saarbrücken Synthetic Image Database. `http://www.nt.uni-saarland.de/SSID/`

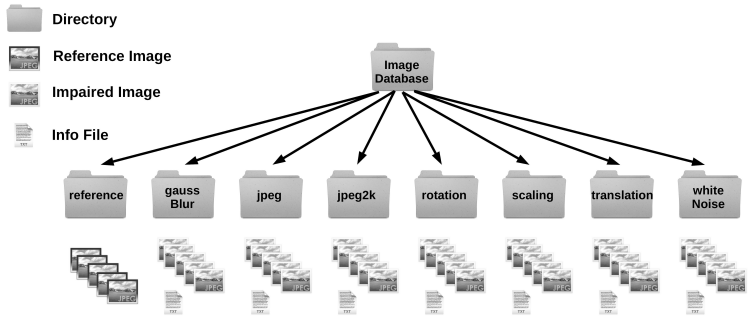
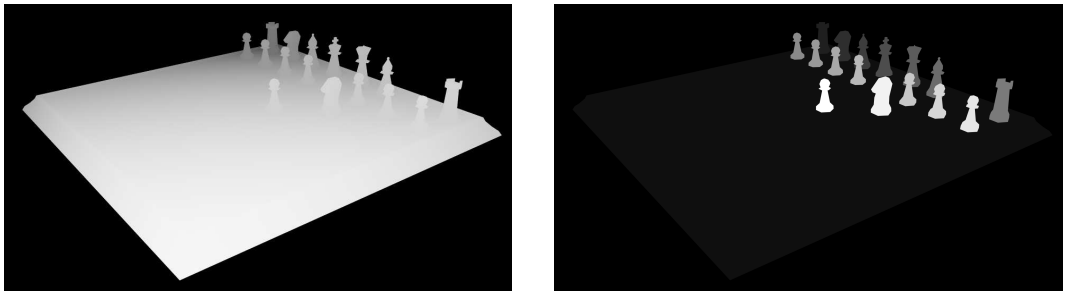


Figure 5.25: Structure of the SID2015 - Synthetic Image Database



(a) Depth Map

(b) Segmentation Mask

Figure 5.26: Depth and Segmentation for the Chess Scene

containing the same details which are image name, reference image name and error parameter.

A huge benefit of fully synthetic content over captured images is that additional data can be produced. We provide Python scripts for Blender with our database that allow depth map and segmentation map rendering (see Figure 5.26). Additional sensory data, for example multiple camera views, can be generated based on the provided sources.

5.3 No-Reference Metrics

No-reference metrics are the class of quality metrics which evaluate the quality of some test information without any ground truth or reference. No-reference metrics are the most powerful class of metrics among partial-reference and full-reference metrics, as the set of information which can be tested with no-reference metrics includes all information that can be tested with both, partial- and full-reference metrics. At the same time there is a lot of information which

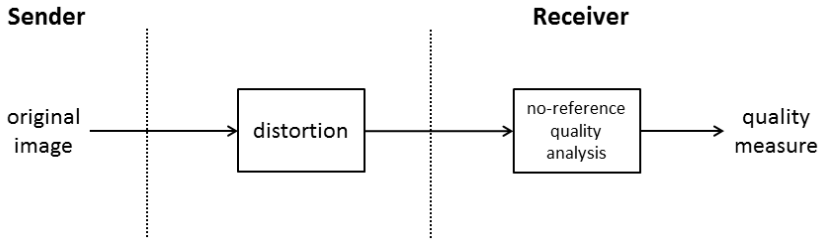


Figure 5.27: Structure of a No-Reference Image Quality Metric

has no reference and no information about content, therefore cannot be evaluated by partial- or full-reference metrics. Exemplary in the visual domain are images captured by a camera. Image distortions which occur during data capture already, such as defocus or ringing artifacts from initial compression, can only be assessed with no-reference metrics. Figure 5.27 outlines the structure of a no-reference metric.

While no-reference metrics are most powerful, they are also most difficult to implement. Of the variety of image or video distortions research on no-reference metrics has therefore mainly focused on image blur (e.g. [59, 9]), ringing artifacts caused by compression (e.g. [48, 104]) and noise (e.g. [18, 45]).

5.3.1 Image Blur Detection

To detect image blur different approaches can be used. Marziliano et al. observe that blurring increases the width of edges in an image. They therefore propose to first employ an edge detector and second measure the width of the edge by observing the local extrema positions closest to the edge. All observed edge measurements over an image are averaged to give the global edge measure, characterizing the amount of blur in an image. [59]

Crété-Roffet et al. base their approach on a different observation: Blurring decreases the variations between neighboring pixels. If an image is already blurred, blurring it again changes the variations between neighboring pixels less than blurring a sharp image. The no-reference metric proposed by Crété-Roffet et al. therefore applies a strong blurring filter to a test image and compares the resulting variation to the original variation. The closer both variations are, the stronger the blur in the test image. [9]

Both approaches were shown to correlate well to subjective tests.

5.3.2 Coding Artifact Detection

Detection of coding artifacts is a critical task to detect and measure the encoding quality. Common to several forms of standard video and image compression like H.264 (see Section 3.2.1), JPEG or JPEG2000 (see Section 5.1.1) are two kinds of visible image distortions: ringing and blocking artifacts.

Blocking artifacts are a result of the coding block structure employed by H.264 and JPEG. Both coding schemes employ sub-image blocks which are individually encoded by a transform intro frequency space and subsequent quantization of the frequency components. As the quantization step is lossy, reconstructions of neighboring blocks may become visually different. These visible differences between neighboring sub-image blocks are known as blocking artifacts.

Ringing artifacts are common artifacts to all encoding schemes based on reductions in the frequency domain. The above mentioned codecs name just a few of them. A reduction in the sampling frequency for any kind of information leads to spectral repetitions of the information. In the spatial domain this leads to repeating wrinkles which extend circularly from boundaries. These wrinkles, or rings, are the visual artifacts denoted as ringing artifacts.

Wang et al. [104] and Vlachos [101] describe no-reference metrics to detect blocking artifacts in images. Wang et al. compare a test image to a shifted version of itself and analyze the frequency content of the difference. In frequency domain blocking artifacts become visible as peaks of the power spectrum at frequencies f corresponding to the block sizes s by $f = \frac{1}{s}$. [104]

The blockiness measure proposed by Vlachos makes use of the usually known sub-image block size of coding schemes. By calculating the cross-correlation of pixels inside of a sub-image block to pixels outside of such a block differences can be measured and related to blocking artifacts due to coding. [101]

5.3.3 Image Noise Detection

For noise detection without a reference image both Lee and Hoppel [45] and Farias and Mitra [18] employ the idea that noise corresponds to local variance of a flat area. The approach suggested by Farias and Mitra splits a test image into blocks of size 7×7, which are again split into 9 overlapping sub-blocks of size 3×3. For each of the sub-blocks the the variance can be calculated. An initial noise estimate can be calculated from the histogram of these sub-block variances according to Lee and Hoppe. However, Farias and Mitra observe that such a noise estimate usually overestimates the actual noise due to the content effect: variance in the image content is added to the noise variance.

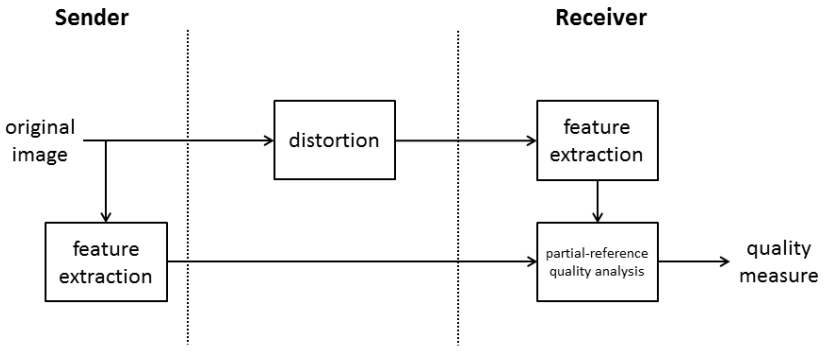


Figure 5.28: Structure of a Partial-Reference Image Quality Metric

They suggest to iteratively refine the noise measurement by a threshold based fade-out of the histogram function.

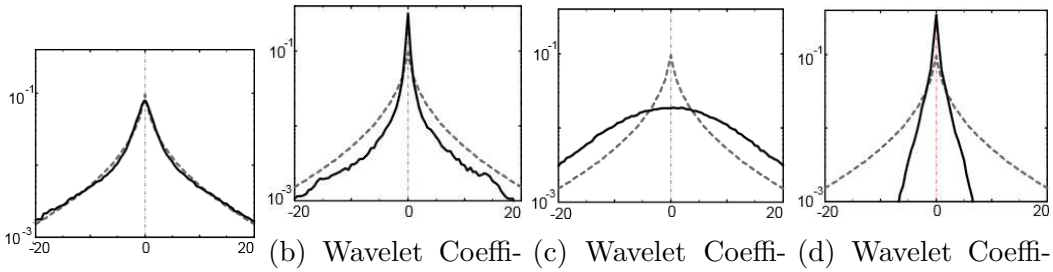
5.4 Partial-Reference Metrics

No-reference metrics form the class of most widely applicable objective image quality assessment metrics. At the same time they are most difficult to implement. All of the image distortion detections described in Section 5.3 are designed for very specific distortion cases. While combined evaluations for these different cases are possible, there are currently no no-reference metrics which work on arbitrary images with arbitrary distortions.

Partial-Reference Metrics or reduced-reference metrics aim for a trade-off between necessary reference information, range of image distortions and quality of distortion detection. A core idea is to provide a small set of information characterizing the information to be assessed. This smaller set of information can than be analyzed to shed light on the quality of the underlying information. Figure 5.28 outlines the structure of a partial-reference metric.

One of the current state-of-the-art approaches is presented by Wang and Simoncelli. Wang and Simoncelli have analyzed histograms of wavelet transforms of images and detected that these histograms exhibit two features which enable the use of these histograms for a partial-reference metric. On the one hand the histogram curve can well be approximated by a generalized Gaussian density model, which is important for the generation of only few reference features. On the other hand, multiple kinds of image distortion can easily be detected based on a comparison between the test image and the Gaussian reference model. [105]

Figure 5.29 illustrates these findings. While the wavelet coefficient his-



(a) Wavelet Coefficient Histogram of Original Image

(b) Wavelet Coefficient Histogram of JPEG2000 Encoded Image

(c) Wavelet Coefficient Histogram of Image under White Noise

(d) Wavelet Coefficient Histogram of Gaussian Blurred Image

Figure 5.29: Comparison of Wavelet Coefficient Histograms used for Partial-Reference Metric [105]

togram of an undistorted image is shown in Figure 5.29a, Figures 5.29b, 5.29c and 5.29d present coefficient histograms of images under JPEG2000, AWGN and Gaussian blur respectively. The grey dotted line represents the curve given by an approximating Gaussian density model. [105]

Several other proposals for partial-reference metrics exist. Common to all of them is the effort to find a set of parameters which suffices to describe an image well enough to detect image distortions. Most literature derives these parameters from frequency representations of images, but the methods to reduce this information differ. Based on the approach published by Wang and Simoncelli Wufeng suggested an enhanced version of steering pyramid coefficient reduction [111] and Abdelouahad et al. make use of Bessel K Forms to reduce the number of relevant coefficients [1].

5.5 Full-Reference Metrics

For both, human observers and machines, image errors become most prominent when a test image can be compared to a reference image. The requirement of having a reference image available however places strong constraints on the usage scenarios. A common scenario is the encoding of image and video information. As the encoder receives an undistorted input image, after the encoding process input and output can be compared. Therefore, full reference metrics offer a valuable contribution to the quality analysis of encoded information. Figure 5.30 outlines the structure of a full-reference metric.

With reference information available full reference metrics are the easiest class of image quality assessment metrics. This has led to a significant amount of research done in this area. Nevertheless, the current state-of-the-art offers

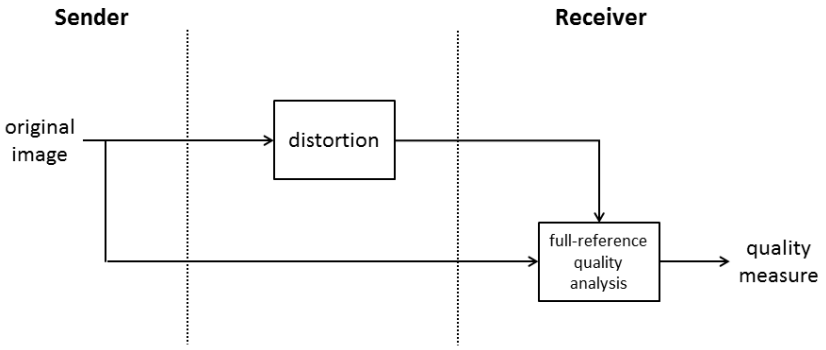


Figure 5.30: Structure of a Full-Reference Image Quality Metric

still substantial room for improvement.

First full reference image quality metrics were purely based on image statistics. Such statistical methods are given e.g. by SNR, MSE or PSNR. These methods do not consider the human viewer at the end at all: purely deviations of the information content are considered. Statistical methods for image quality analysis are introduced in Section 5.5.1.

A next step in the development of image quality metrics has been made with the observation, that the human visual system detects certain image distortions easier than others. Examples are the Structural Similarity Index [103] or methods further enhancing this index like HDR-VDP-2 [57]. Metrics based on the human visual system are detailed in Section 5.5.2.

Novel kinds of data, especially merges of real and synthetic data, introduce novel kinds of errors (see Section 5.1.2). While statistical quality measures are vulnerable to these kinds of image distortions, also metrics based on the human visual system detect and weight these errors. However, human observers often hardly notice image changes as the distorted images remain physically plausible, and contain all requirements - like sharpness, low noise and no coding artifacts - that human observers relate to a high image quality. As the human brain is responsible for this decision about physical plausibility and fulfillment of quality requirements, metrics involving the human cognitive system have been developed to deal with novel kinds of errors. A metric employing some capabilities of the human cognitive system is introduced in Section 5.5.3.

5.5.1 Qualtiy Metrics based on Image Statistics

Quality metrics based on image statistics work purely on the differences contained in the pixel values of a test image t and a reference image r. The most simplistic approach considers the per-pixel difference only. The total error of

a test image of size x, y is than given by the Sum of Absolute Differences SAD

$$SAD = \sum_{i=0}^{x} \sum_{j=0}^{y} |r(i,j) - t(i,j)| \tag{5.20}$$

While it is conceptually similar instead of the absolute value the difference can be squared. Similar to the absolute value this ensures positive values, but additionally facilitates statistical calculations. In a similar fashion to the SAD the Sum of Squared Differences SSD is defined as

$$SSD = \sum_{i=0}^{x} \sum_{j=0}^{y} (r(i,j) - t(i,j))^2 \tag{5.21}$$

Calculating the sum of absolute differences or the sum of squared differences has a significant drawback: it depends on the size of test and reference image. This drawback can be overcome by taking the mean of the calculated values. In these equations the value of the difference is commonly denoted as the error, which introduces the Mean Absolute Error MAE and Mean Squared Error MSE

$$MAE = \frac{1}{x \cdot y} \sum_{i=0}^{x} \sum_{j=0}^{y} |r(i,j) - t(i,j)| \tag{5.22}$$

$$MSE = \frac{1}{x \cdot y} \sum_{i=0}^{x} \sum_{j=0}^{y} (r(i,j) - t(i,j))^2 \tag{5.23}$$

All of the above mentioned measures describe the difference between test image and reference image. To evaluate the impact of this difference, the error needs to be related to the original signal. The larger the amplitude of the original signal, the smaller the impact of a certain error. Such a relation is created by the Signal to Noise Ratio SNR, which puts the noise measured by the Mean Squared Error in relation to the signal power

$$SNR = \frac{\frac{1}{x \cdot y} \sum_{i=0}^{x} \sum_{j=0}^{y} r(i,j)^2}{MSE} \tag{5.24}$$

Instead of comparing an error value to the signal itself, for error evaluations the comparison to the dynamic range of a signal has been found to be more adequate. The dynamic range is given by the peak signal, thus motivating the Peak Signal to Noise Ratio $PSNR$

$$PSNR = \frac{max_{i \in [0,x]} \big(max_{j \in [0,y]} (r(i,j)^2) \big)}{MSE} \tag{5.25}$$

The PSNR is still a very common metric for image quality analysis. On the one hand side PSNR can be easily implemented and has a very low computational complexity which is an important criterion for real-time applications. On the other hand, PSNR is sufficient quality tool for many other fields requiring information analysis, e.g. signal analysis in communication systems. For image quality assessment, however, PSNR relates poorly to subjective image quality findings. According to Wang et al. the correlation coefficient between PSNR and MOS is only at 0.3267 [104], which shows that PSNR is - even though widely used - not very useful for image quality assessment.

Calculating with Ratios: Ratios and Averages

Special care needs to be taken when calculating with ratios and their averages. As ratios are used throughout multiple fields of science (e.g. biologists for leaf area indices [77], hematologists for descriptions of morphology of nematodes [21] or engineers of various fields to calculate SNRs [39, 14]), this small excursus is dedicated to the introduction of an approach to correctly calculate averages of ratios compared to ratios of averages. Both, ratios of averages and averages of ratios have their purpose, and the value of both has has been discussed in the literature [19, 72].

In 2011 Brown indicates that for experimental results both numerator x and denominator y are usually positive and averaged over all experiments [4]. When obtaining experimental results Brown observes three different scenarios:

1. the (x_i, y_i) pairs, for $i = 1, 2, \ldots, n$ are known;

2. x_i, $i = 1, 2, \ldots, n$, and y_j, $j = 1, 2, \ldots, m$ are known, but the values are not paired; and

3. the data are measured as a ratio, so the data are $r_i = x_i/y_i$, for $i = 1, 2, \ldots, n$.

While the first case allows to calculate two different averages, this is not possible for the latter two. In the first case the ratio of averages can be calculated as

$$r = \frac{\frac{1}{n}\sum_{i=1}^{n} x_1}{\frac{1}{n}\sum_{i=1}^{n} y_1} \tag{5.26}$$

and the average of ratios is calculated as

$$q = \frac{1}{n} \sum_{i=1}^{n} \frac{x_i}{y_i} \tag{5.27}$$

With unpaired values x_i, y_i only r can be calculated and if the data is measured directly as ratios only q can be calculated. According to Culley (2002) q and r are often used interchangeably and q is often used as an approximation of r [11]. However q need not equal r, and can even differ largely.

For illustration purpose imagine a program calculating the SNR of a video frame and reporting this SNR. In addition, the program reports an average video SNR by averaging the SNRs of a number of frames. Calculating this example with numbers illustrates, that the program will usually report a higher average SNR of the video than is actually present in the data. Assume a signal power which is $9W$ in the first frame and $11W$ in the second frame. Furthermore, assume noise power of $3W$ for the first frame and $1W$ for the second frame. The average signal power of both frames is $10W$, the average noise power $2W$, resulting in a $SNR = 10 \cdot log\,(5)$. However, the program reports $SNR_1 = 10 \cdot log\,(3)$ for the first frame and $SNR_2 = 10 \cdot log\,(11)$ for the second frame, thus calculating an average of $SNR = 10 \cdot log\,(7)$.

According to [4] there is no particular mathematical relationship between r and q. This widely held assumption is, however, wrong; r can indeed be calculated from ratios x_i/y_i given a sufficiently large set of measured ratios r_i and some knowledge about the variable distribution. Even without any knowledge there is an approximation to r which is significantly better than the approximation q.

Proposition Assume random variables x, y with expected values μ_x, μ_y and standard deviations σ_x, σ_y respectively. Define the ratio of averages r as

$$\begin{aligned} r &= \frac{\frac{1}{n}\sum_{i=1}^{n} x_1}{\frac{1}{n}\sum_{i=1}^{n} y_1} \\ &= \frac{\mu_x}{\mu_y} \end{aligned} \tag{5.28}$$

For the ratio of averages it holds that

$$\begin{aligned} \frac{1}{r} &= \frac{\sum_{i=1}^{n} y_1}{\sum_{i=1}^{n} x_1} \\ &= \frac{\mu_y}{\mu_x} \end{aligned} \tag{5.29}$$

thus

$$\frac{\sum_{i=1}^{n} x_1}{\sum_{i=1}^{n} y_1} \cdot \frac{\sum_{i=1}^{n} y_1}{\sum_{i=1}^{n} x_1} = 1 \tag{5.30}$$

Calculation of r requires that either the individual values x_i, y_i or the sums or averages of those are given. If only ratios x_i/y_i are available, r - according to the literature - cannot be calculated. A common approximation for r from a set of ratios is the average of ratios q, defined as

$$q = \frac{1}{n} \sum_{i=1}^{n} \frac{x_i}{y_i} \tag{5.31}$$

However, it is well known that q need not equal r, and the multiplicative inverse of q cannot be calculated as nicely as it can be for r, as usually

$$\frac{1}{n} \sum_{i=1}^{n} \frac{x_i}{y_i} \cdot \frac{1}{n} \sum_{i=1}^{n} \frac{y_i}{x_i} \neq 1 \tag{5.32}$$

With this observation there is a specific mathematical relationship between r and q and r can even be calculated exactly or well approximated given the individual ratios $\frac{x_i}{y_i}$. For known $\frac{x_i}{y_i}$ it is

$$h = -\left(\frac{d}{4c_1} - c_1 - \frac{1}{4c_2}\right) \pm \sqrt{\left(\frac{d}{4c_1} - c_1 - \frac{1}{4c_2}\right)^2 - \frac{c_1}{c_2}} \tag{5.33}$$

with

$$c_1 := \frac{1}{n} \sum_{i=1}^{n} \frac{x_i}{y_i} \tag{5.34}$$

$$c_2 := \frac{1}{n} \sum_{i=1}^{n} \frac{y_i}{x_i} \tag{5.35}$$

and

$$d := \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i}{y_i}\right)^2 \tag{5.36}$$

and h is a close approximation or exact calculation of r.

Proof Write random variable $x_i = \mu_x + \delta_x$ where δ_x is a random variable with $\mu_{\delta_x} = 0$ and $\sigma_{\delta_x} = \sigma_x$. Use the same nomenclature for y_i, and assume only ratios $\frac{x_i}{y_i}$ are available. Now for a sufficiently large set of ratios $\frac{x_i}{y_i}$ for each $\frac{x_i}{y_i} = \frac{\mu_x+\delta_x}{\mu_y+\delta_y}$ there exist (somewhere in the set) the ratios $\frac{\mu_x-\delta_x}{\mu_y-\delta_y}$, $\frac{\mu_y+\delta_y}{\mu_y-\delta_y}$ and $\frac{\mu_x-\delta_x}{\mu_y+\delta_y}$. Due to commutativity of sums we can reorder the sum of ratios, such that four ratios as given above can be considered together.

The average of such a block of four corresponding ratios is calculated as

$$\begin{aligned}
&\frac{1}{4}\left(\frac{\mu_x+\delta_x}{\mu_y+\delta_y}+\frac{\mu_x-\delta_x}{\mu_y-\delta_y}+\frac{\mu_x+\delta_x}{\mu_y-\delta_y}+\frac{\mu_x-\delta_x}{\mu_y+\delta_y}\right)\\
=&\frac{1}{4}\left[\left(\frac{\mu_x+\delta_x}{\mu_y+\delta_y}+\frac{\mu_x-\delta_x}{\mu_y-\delta_y}\right)+\left(\frac{\mu_x+\delta_x}{\mu_y-\delta_y}+\frac{\mu_x-\delta_x}{\mu_y+\delta_y}\right)\right]\\
=&\frac{1}{4}\left(\frac{2\mu_x\mu_y-2\delta_x\delta_y}{\mu_y^2-\delta_y^2}+\frac{2\mu_x\mu_y+2\delta_x\delta_y}{\mu_y^2-\delta_y^2}\right)\\
=&\frac{1}{4}\frac{4\mu_x\mu_y}{\mu_y^2-\delta_y^2}\\
=&\frac{\mu_x\mu_y}{\mu_y^2-\delta_y^2}
\end{aligned} \tag{5.37}$$

At this point it can be noticed that the average of a set of ratios only varies with respect to δ_y. If a set of ratios is variable only with respect to either numerator or denominator, the r can be calculated as the average of ratios by keeping the variable content in the numerator. For example, PSNR (see Section 5.5.1) signals often have a constant dynamic range and therefore have a constant peak amplitude value. To calculate r from a set of ratios inverting the PSNR to Noise-to-Peak-Signal - ratios gives the desired result:

$$r = \frac{1}{\frac{1}{n}\sum_{i=1}^{n}\frac{y_i}{x_i}} \tag{5.38}$$

In many cases, like SNR, both, numerator and denominator, of experimental results are variable, thus require further investigation. For binary distributed errors $\delta_x = \delta_y = \pm\delta$ the average of m such blocks remains the same. However, if the error δ_y is not binary distributed the average of $m = n/4$ such blocks as given in Equation 5.37 for large m is calculated as

$$\begin{aligned}
&\lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} \frac{\mu_x \mu_y}{\mu_y^2 - \delta_{y_k}^2} \\
=& \lim_{m\to\infty} \frac{1}{m} \frac{\mu_x \mu_y \sum_{k=1}^{m} \prod_{l\in\{1,m\}\setminus k} \left(\mu_y^2 - \delta_{y_l}^2\right)}{\prod_{k=1}^{m} \left(\mu_y^2 - \delta_{y_k}^2\right)} \\
\approx& \frac{\mu_x \mu_y}{\mu_y^2 - E(\delta_y^2)}
\end{aligned} \tag{5.39}$$

This gives the definition of c_1 as

$$\begin{aligned}
c_1 :&= \frac{1}{n} \sum_{i=1}^{n} \frac{x_i}{y_i} \\
&\approx \frac{\mu_x \mu_y}{\mu_y^2 - E(\delta_y^2)} \\
&= \frac{\mu_x}{\mu_y - E(\delta_y^2)/\mu_y}
\end{aligned} \tag{5.40}$$

In an analogue way the average of inverted ratios is calculated and define

$$\begin{aligned}
c_2 :&= \frac{1}{n} \sum_{i=1}^{n} \frac{y_i}{x_i} \\
&\approx \frac{\mu_x \mu_y}{\mu_x^2 - E(\delta_x^2)} \\
&= \frac{\mu_y}{\mu_x - E(\delta_x^2)/\mu_x}
\end{aligned} \tag{5.41}$$

At this point the ratio r can be calculated for two special cases: if the noise $\delta_x = \delta_y$ and if the signal-to-noise ratio $E(\delta_x^2)/\mu_x^2 = E(\delta_y^2)/\mu_y^2$. The derivation of r for these cases can be found in Paragraph 5.5.1.

To calculate r for arbitrary x_i, y_i even more information is needed, which can be obtained by the squared ratio

$$d_1 := \frac{1}{n} \sum_{i=1}^{n} \frac{x_i^2}{y_i^2} \tag{5.42}$$

Considering the same four blocks as before $\frac{\mu_x+\delta_x}{\mu_x+\delta_x}$, $\frac{\mu_x-\delta_x}{\mu_x-\delta_x}$, $\frac{\mu_x+\delta_x}{\mu_x-\delta_x}$ and $\frac{\mu_x-\delta_x}{\mu_x+\delta_x}$ these blocks can be averaged

$$\begin{aligned}&\frac{1}{4}\left(\left(\frac{\mu_x+\delta_x}{\mu_y+\delta_y}\right)^2+\left(\frac{\mu_x-\delta_x}{\mu_y-\delta_y}\right)^2+\left(\frac{\mu_x+\delta_x}{\mu_y-\delta_y}\right)^2+\left(\frac{\mu_x-\delta_x}{\mu_y+\delta_y}\right)^2\right)\\ =&\frac{(\mu_x^2+\delta_x^2)\cdot\left(\mu_y^2+\delta_y^2\right)}{\left(\mu_y^2-\delta_y^2\right)^2}\end{aligned} \tag{5.43}$$

Similar to equation 5.39 the average of $m = n/4$ blocks as given in 5.43 is calculated as

$$\begin{aligned}&\lim_{m\to\infty}\frac{1}{m}\sum_{k=1}^{m}\frac{(\mu_x^2+\delta_x^2)\cdot\left(\mu_y^2+\delta_y^2\right)}{\left(\mu_y^2-\delta_y^2\right)^2}\\ =&\lim_{m\to\infty}\frac{1}{m}\frac{\sum_{k=1}^{m}(\mu_x^2+\delta_x^2)\cdot\left(\mu_y^2+\delta_y^2\right)\cdot\prod_{l\in\{1,m\}\setminus k}\left(\mu_y^2-\delta_{y_l}^2\right)^2}{\prod_{k=1}^{m}\left(\mu_y^2-\delta_{y_k}^2\right)^2}\\ \approx&\frac{(\mu_x^2+E(\delta_x^2))\cdot\left(\mu_y^2+E(\delta_y^2)\right)}{\left(\mu_y^2-E(\delta_y^2)\right)^2}\end{aligned} \tag{5.44}$$

According to equations 5.42, 5.43 and 5.44 d_1 is defined as

$$\begin{aligned}d_1:&=\frac{1}{n}\sum_{i=1}^{n}\frac{x_i^2}{y_i^2}\\ &\approx\frac{(\mu_x^2+E(\delta_x^2))\cdot\left(\mu_y^2+E(\delta_y^2)\right)}{\left(\mu_y^2-E(\delta_y^2)\right)^2}\end{aligned} \tag{5.45}$$

Note that for

$$d_2:=\sum_{i=1}^{n}\frac{y_i^2}{x_i^2} \tag{5.46}$$

it is

$$\frac{d_1}{c_1^2}=\frac{d_2}{c_2^2} \tag{5.47}$$

therefore d_2 does not add any further information. In the following calculations d_2 is ignored and $d := d_1$.

Now it is

$$
\begin{aligned}
d &= \frac{(\mu_x^2 + E(\delta_x^2)) \cdot \left(\mu_y^2 + E(\delta_y^2)\right)}{\left(\mu_y^2 - E(\delta_y^2)\right)^2} \\
&= \frac{\left(\frac{\mu_x}{\mu_y} + \frac{E(\delta_x^2)}{\mu_x\mu_y}\right) \cdot \left(\frac{\mu_y}{\mu_x} + \frac{E(\delta_y^2)}{\mu_x\mu_y}\right)}{\left(\frac{\mu_y}{\mu_x} - \frac{E(\delta_y^2)}{\mu_x\mu_y}\right)^2} \\
&= \frac{\left(r + \frac{E(\delta_x^2)}{\mu_x\mu_y}\right) \cdot \left(\frac{1}{r} + \frac{E(\delta_y^2)}{\mu_x\mu_y}\right)}{\left(\frac{1}{r} - \frac{E(\delta_y^2)}{\mu_x\mu_y}\right)^2}
\end{aligned}
\tag{5.48}
$$

In Equation 5.48 $\frac{E(\delta_y^2)}{\mu_x\mu_y}$ and $\frac{E(\delta_x^2)}{\mu_x\mu_y}$ can be substituted by

$$
\begin{aligned}
c_1 &= \frac{\mu_x\mu_y}{\left(\mu_y^2 - E(\delta_y^2)\right)} \\
\Leftrightarrow \frac{1}{c_1} &= \frac{\mu_y^2}{\mu_x\mu_y} - \frac{E(\delta_y^2)}{\mu_x\mu_y} \\
\Leftrightarrow \frac{1}{r} - \frac{1}{c_1} &= \frac{E(\delta_y^2)}{\mu_x\mu_y}
\end{aligned}
\tag{5.49}
$$

and analogue

$$
\begin{aligned}
c_2 &= \frac{\mu_x\mu_y}{(\mu_x^2 - E(\delta_x^2))} \\
\Leftrightarrow r - \frac{1}{c_2} &= \frac{E(\delta_x^2)}{\mu_x\mu_y}
\end{aligned}
\tag{5.50}
$$

Substituting the last two equations into d gives

$$
\begin{aligned}
d &= \frac{\left(2 \cdot r - \frac{1}{c_2}\right) \cdot \left(\frac{2}{r} - \frac{1}{c_1}\right)}{\frac{1}{c_1^2}} \\
\Leftrightarrow 0 &= r^2 + r \cdot \left(\frac{d}{2c_1} - 2c_1 - \frac{1}{2c_2}\right) + \frac{c_1}{c_2} \\
\Leftrightarrow r &= -\left(\frac{d}{4c_1} - c_1 - \frac{1}{4c_2}\right) \pm \sqrt{\left(\frac{d}{4c_1} - c_1 - \frac{1}{4c_2}\right)^2 - \frac{c_1}{c_2}}
\end{aligned}
\tag{5.51}
$$

Special Cases If special assumptions about the variance hold, the ratio r can be calculated from equations 5.40 and 5.41 already. It is

$$\begin{aligned}\frac{1}{c_1} &= \frac{\mu_y - E(\delta_y^2)/\mu_y}{\mu_x} \\ &= \frac{\mu_y}{\mu_x} - \frac{E(\delta_y^2)}{\mu_x\mu_y} \\ &= \frac{1}{r} - \frac{E(\delta_y^2)}{\mu_x\mu_y}\end{aligned} \tag{5.52}$$

and

$$\begin{aligned}\frac{1}{c_2} &= \frac{\mu_x - E(\delta_x^2)/\mu_x}{\mu_y} \\ &= \frac{\mu_x}{\mu_y} - \frac{E(\delta_x^2)}{\mu_x\mu_y} \\ &= r - \frac{E(\delta_x^2)}{\mu_x\mu_y}\end{aligned} \tag{5.53}$$

Same noise: $\delta_x = \delta_y = \delta$ Assuming $\delta_x = \delta_y = \delta$ Equations 5.52 and 5.53 can be solved for r, giving

$$\begin{aligned}0 &= \frac{1}{r} - \frac{1}{c_1} - r + \frac{1}{c_2} \\ 0 &= 1 - r \cdot \frac{1}{c_1} - r^2 + r \cdot \frac{1}{c_2} \\ 0 &= r^2 + r \cdot \left(\frac{1}{c_1} - \frac{1}{c_2}\right) - 1 \\ r &= -\frac{1}{2} \cdot \left(\frac{1}{c_2} - \frac{1}{c_1}\right) \pm \sqrt{\frac{1}{4} \cdot \left(\frac{1}{c_2} - \frac{1}{c_1}\right)^2 + 1}\end{aligned} \tag{5.54}$$

Proportional Variance: $E(\delta_x^2)/\mu_x^2 = E(\delta_y^2)/\mu_y^2$ By rewriting equations 5.52 and 5.53 as

$$\begin{aligned}\frac{1}{c_1} &= \frac{1}{r} - \frac{\delta^2}{\mu_x\mu_y} \\ &= \frac{1}{r} \cdot \left(1 - \frac{E(\delta_y^2)}{\mu_y^2}\right)\end{aligned} \tag{5.55}$$

and

$$\begin{aligned}\frac{1}{c_2} &= \frac{1}{r} - \frac{\delta^2}{\mu_x \mu_y} \\ &= r \cdot \left(1 - \frac{E(\delta_x^2)}{\mu_x^2}\right)\end{aligned} \tag{5.56}$$

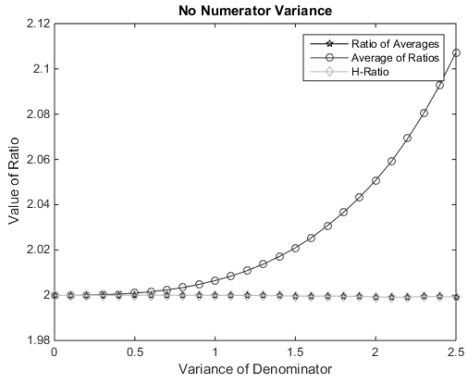
and furthermore assuming $\frac{E(\delta_x^2)}{\mu_x^2} = \frac{E(\delta_y^2)}{\mu_y^2}$ (proportional variance on x_i and y_i) gives

$$\begin{aligned}\frac{r}{c_1} &= \frac{1}{r \cdot c_2} \\ \Leftrightarrow r^2 &= \frac{c_1}{c_2} \\ \Leftrightarrow r &= \sqrt{\frac{c_1}{c_2}}\end{aligned} \tag{5.57}$$

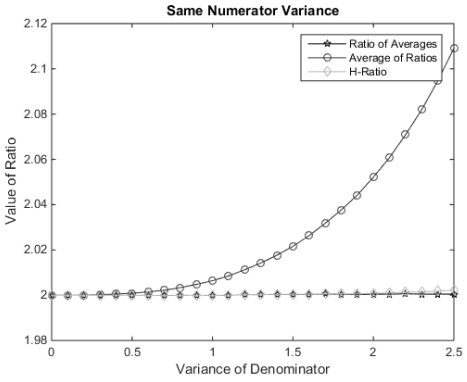
Figure 5.31 shows the resulting calculations for the ratio of averages from sets of 1.000.000 randomly distributed individual ratios. In the first three cases, which are no variance in the numerator (Figure 5.31a and calculated with Equation 5.38), same variance in numerator and denominator (Figure 5.31b and calculated with Equation 5.54) and proportional noise in numerator and denominator (Figure 5.31c and calculated with Equation 5.57) the H-Ratio h approximates the Ratio of Averages r exactly, thus the lines are overlapping. Only without any knowledge about the numerator variance the approximating Equation 5.51 needs to be employed, which does not lead to the exact value of r but to a much better approximation than is offered by the Average of Ratios q (see Figure 5.31d). In all Figures the Ratio of Averages is denoted by the line with star markers, the Average of Ratios has circle markers, and the H-Ratio has diamond markers.

5.5.2 Quality Metrics based on the Human Visual System

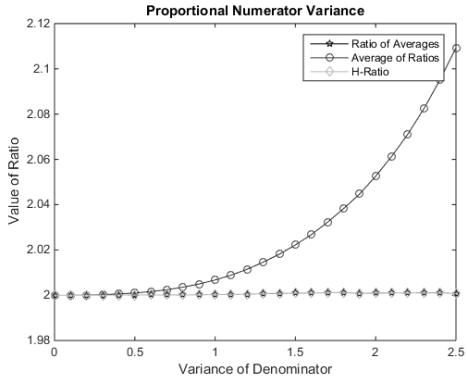
The human visual system plays an important role for all kinds of image processing. Most literature defines the human visual system as the human eye and the part of the human brain that is responsible for acquisition and processing of visual information. In the context of quality measurements, however, a distinction can be made between quality degradations that can not physically be



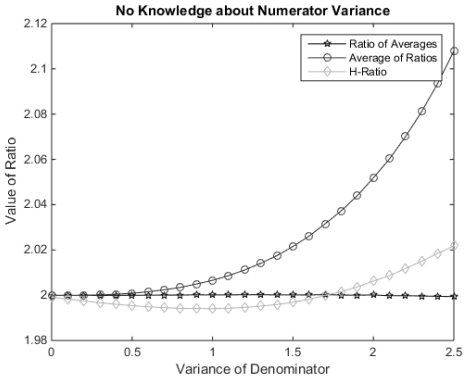
(a) Constant Numerator or Numerator with zero Variance



(b) Numerator has the same Variance as the Denominator



(c) Numerator has a proportional Variance to the Denominator



(d) No knowledge about the Variance of the Numerator

Figure 5.31: Calculating the Ratio of Averages from Individual Ratios

detected by the human eye and quality degradations that are corrected by the human brain. It therefore makes sense to separate both parts of the human pipeline for visual information acquisition and processing into Human Visual System, which contains the physical part of human vision, and the Human Cognitive System containing the cognitive processes relevant for vision.

Maintz describes the core elements of the human eye necessary for vision in his work on digital and medical image processing. [55] The human eye can detect electromagnetic waves in the spectrum from about $300nm$ to $700nm$. When electromagnetic waves of this range reach the human eye several physical elements of the eye play an important role. The iris defines the size of the opening (the pupil), through which light can enter the eye. In bright conditions it closes the opening, in dark conditions it widens the opening such that more of the spare light can enter the eye. Behind the iris is the lens, which takes care of focusing the light rays. The nearest and farthest point a human eye can focus on are called near point and far point, respectively. Healthy eyes of young people have a near point at $7cm$ and a far point at infinity. Light rays are focused on the retina, where light rays are turned into electrical signals send to the brain. Cells sensible to light rays are called photoreceptors, and the human eye contains two kinds of those, rods and cones. Roughly $100million$ rods are distributed all over the retina, while only about $6million$ cones are located mostly central in the retina. Rods only detect brightness resulting in grayscale images. Three different kinds of cones, each sensitive to a different range of electromagnetic waves, allow to differentiate colors. A schematic diagram of the human eye visualizing these attributes is shown in Figure 5.32.

The consequences induced by these physical constraints of the human acquisition tool for visual information, the eye, are manifold. The limited number of rods and cones introduces low-pass filter characteristics. This for modern displays effectively makes the retina, not the displaying devices, the bottleneck for image resolution, leading to the term “retina display”. Apart from the overall number of rods and cones, the number of rods, responsible for luminance detection, outnumber the cones, responsible for color detection, by far, making the human eye a lot more sensible to luminance changes than to color changes. At the same time stimuli of the rods have an impact on neighboring stimuli, leading to a spatial masking of luminance. An optical illusion of a checkerboard with a bright light patch illustrates this characteristic of the human visual system (see Figure 5.36a). Both of the checkerboard squares A and B actually have the same brightness, but different environments lead to the illusion that B is darker than A. Illusions like this one motivate the consideration of the human visual system as an important factor in image quality analysis.

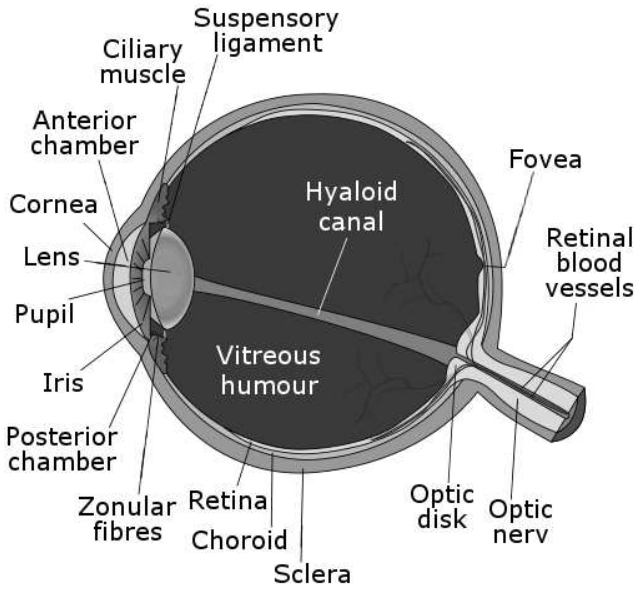


Figure 5.32: Schematic Diagram of the Human Eye

The Sarnoff JND Model

One of the first persons to notice and algorithmically tackle the problem of image quality metrics lacking the correlation to subjective quality assignments was J. Lubin. With the Sarnoff Corporation Lubin developed a model to describe the Just Noticeable Difference (JND) Vision Model, which became known by the company name as Sarnoff Vision Model or Sarnoff metric. [52]

A core idea of the Sarnoff model is that human observers can detect luminance differences easier than chroma differences. The Sarnoff model therefore splits an image into luminance and chroma information, one luminance channel and two chroma channels. Subsequent to this split each of the three channels is decomposed images of decreasing resolution by pyramid decomposition to consider difference detections for different frequency contents. These differently scaled images can than be compared to reference images who have undergone the same process. Differences detected here between test and reference information are related through psycho-physical experiments to the JNDs, where 1 JND describes the difference that 75% of the human observers will detect. In a final stage masking considers effects described by Carlsen and Cohen [7], that noise is less visible in "busy" areas of an image. The architecture of the Sarnoff JND Model is shown in Figure 5.33. [52]

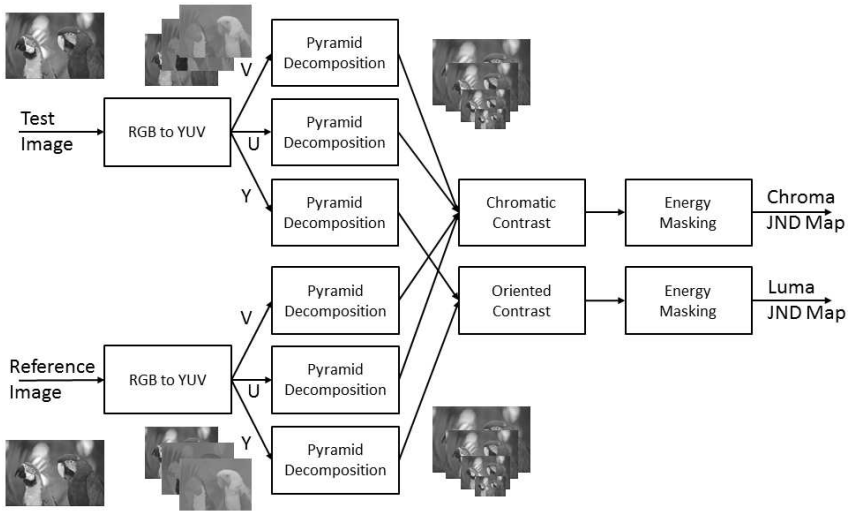


Figure 5.33: Architecture of the Sarnoff JND Model [52]

The Structural Similarity Index

Following the same line of thought as the Sarnoff Model Wang et al. have introduced an image quality metric based on structural similarity. The core idea is that the human visual system, as introduced above, is most sensitive to brightness changes in an image. Brightness changes occurring in all color channels are perceived as image structures. With this observation Wang et al. postulate the - in 2004 novel - philosophy of image degradations corresponding to perceived changes in structural information [103].

The system proposed by Wang et al. compares three different image components: luminance, contrast and structure. The luminance l_T of test image T with $M \times N$ pixels is given by its mean pixel intensity

$$l_T = \frac{1}{M \cdot N} \sum_{x=1}^{M} \sum_{y=1}^{N} T(x, y) \tag{5.58}$$

The luminance of a test image l_T can than be compared to the luminance of a reference image l_R. Specifically, it is

$$l(T, R) = \frac{2 \cdot l_T \cdot l_R + C_1}{l_T^2 + l_R^2 + C_1} \tag{5.59}$$

where C_1 is a constant introduced to avoid instability when the denominator would become close to zero which is proportional to the square of the dynamic range of an image.

If the image luminance or mean pixel intensity is removed from an image the contrast is can be measured, which corresponds to the standard deviation of the image. The contrast σ_T therefore is

$$c_T = \sqrt{\frac{1}{(M-1)\cdot(N-1)} \sum_{x=1}^{M} \sum_{y=1}^{N} (T(x,y) - l_T)^2} \tag{5.60}$$

which can - equivalent to the luminance - be compared to the reference contrast c_R in a contrast comparison

$$c(T,R) = \frac{2 \cdot c_R \cdot c_T + C_2}{c_R^2 + c_T^2 + C_2} \tag{5.61}$$

Again, C_2 is a constant introduced to avoid instability and C_2 is proportional to the square of the dynamic range of an image.

Finally, the structure comparison is computed on test and reference image after shifting and normalizing the image information, with

$$s(T,R) = \frac{c_{TR} + C_3}{c_T \cdot c_R + C_3} \tag{5.62}$$

where

$$c_{TR} = \frac{1}{(M-1)(N-1)} \sum_{x=1}^{M} \sum_{y=1}^{N} (T(x,y) - l_T) \cdot (R(x,y) - l_R) \tag{5.63}$$

and C_3 as C_1 and C_2. The structural similarity between a test and a reference image $SSIM(R,T)$ is than calculated as

$$SSIM(T,R) = l(T,R)^\alpha \cdot c(T,R)^\beta \cdot s(R,T)^\gamma \tag{5.64}$$

with $0 < \alpha, \beta, \gamma$. The structure of the SSIM defined by Equations 5.58 to 5.64 is visualized in Figure 5.34.[103]

The High Dynamic Range - Visible Difference Predictor

Based on the ideas delevoped by Wang et al. in their works on structural similarity, Mantiuk et al. have extended this visual model to differentiate more complex scenarios. A high dynamic range visible difference predictor (HDR-VDP) was introduced in 2005 [56] and completely overhauled in 2011, forming HDR-VDP 2 [57]. According to Mantiuk et al. the vision model presented in [57] is applicable to a wide range of viewing conditions, especially

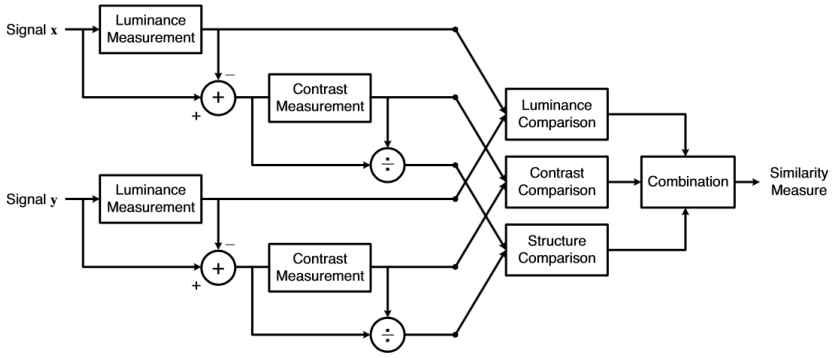


Figure 5.34: Structure of SSIM Metric [103]

luminance changes. It furthermore mimics several aspects of the human visual system and by doing that presents an improved quality metric for images.

For HDR-VDP 2 both, test and reference image, are send through a model of the optical and retinal pathway. This model returns the response of cones and rods to light, where the model includes the spectral sensitivity and the luminance masking of photoreceptors. Subsequent to this model of the optical and retinal pathway multi-scale decompositions, employing steerable pyramids [92], are computed. To the difference between the multi-scale decompositions of test image and reference image neural noise masking some of the computed differences is applied. The outcome of this modeling step is a contrast difference linearized to human perception, which can be employed to compute a visibility and a quality metric for the input test and reference stimuli.

The visibility metric calculates a probability map or single probability value representing the probability of detecting a visible difference per pixel or for the overall image respectively. To derive this information from the contrast difference first a reconstruction of the data followed by a spatial integration and application of scaling according to psychometric experiments is computed. In a similar manner a predicted MOS can be calculated from the contrast difference by a logistic function. A simplified structure of the HDR-VDP 2 approach is shown in Figure 5.35 [57].

5.5.3 Quality Metrics based on the Human Cognitive System

After light information has been captured by the eye it is processed by the human brain, or the human cognitive system. Here content is segmented and sorted into known classes. Sorting is often successful, even if the underlying information itself is insufficient for correctly assigning semantic meanings to

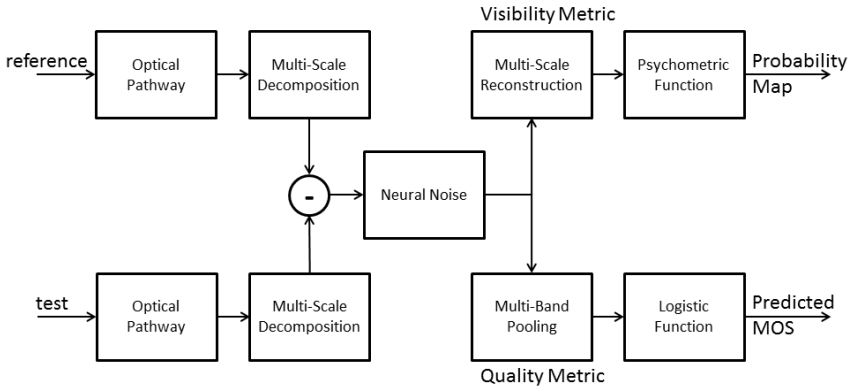


Figure 5.35: Structure of HDR-VDP 2 [57]

(a) Checkerboard (b) Hidden Dalmatian (c) Kanizsa's Triangle

Figure 5.36: Optical Illusions illustrating the Human Visual System and the Human Cognitive System

image content. This capability of the brain can well be demonstrated with Figures 5.36b and 5.36c. In 5.36b most viewers detect a dalmatian sniffing on the ground, and 5.36c is by most observers viewed as a white triangle covering three black circles. In both cases, the outlines of the dalmatian and the triangle seem to be visible. However, neither dalmatian nor triangle have any actual contours, both are results of the human brain connecting a collection of shapes to semantically known shapes. The motivating Figure 5.1 of this chapter showing stripes on a shirt in a different order is another example of a decision made by the human brain: changes which are not in the area of interest and which do not effect the realism of an observed scene are simply ignored.

A logical consequence for the design of visual quality metrics is to model parts of the human cognitive system that automatically correct and ignore certain image content changes that are due to changes in the scene composition.

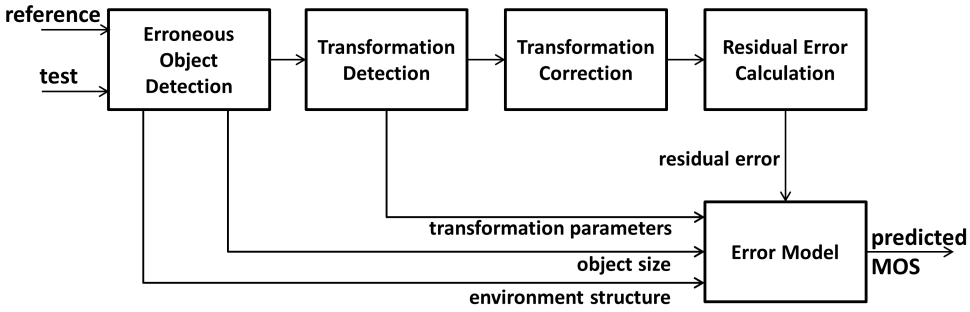


Figure 5.37: Structure of our Proposed SC-VQM

As scene compositions currently usually require synthetic content, a Synthetic Content - Visual Quality Metric (abbreviated SC-VQM) is proposed.

The SC-VQM is designed to detect changes in visual content that affect the perceived realism different to noise or compression artifacts, as they are corrected (partially) by the human brain. The idea to achieve this goal is straight forward: To detect object changes and correct those before calculating a residual error. This idea is outlined in the block diagram in Figure 5.37 and described by the following six steps.

[a] Erroneous object detection: Distorted objects in a scene composition are detected

[b] Erroneous object matching: Objects in test image are matched with objects in reference image

[c] Object size calculation: The portion of the image affected by the distorted object is calculated

[d] Environment structure analysis: The environment of the distorted objects is analyzed for the amount of structures contained

[e] Object correction: The object in the test image is corrected according to the reference object, transformation parameters are recorded

[f] Residual error calculation: The residual error between corrected object and reference image is calculated

[g] Approximate MOS by detected parameters: All parameters from the previous analysis steps are combined in an error model to predict a MOS

(a) Reference Image (b) Test Image

Figure 5.38: Example Image Set Illustrating the Implementation of SC-VQM

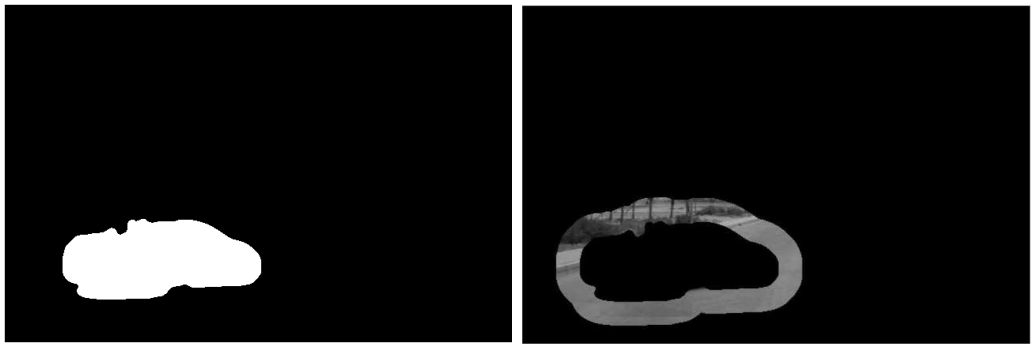
Implementation

The SC-VQM is implemented along the idea outlined above. The following paragraphs explain the implementation details of the six steps and are visualized using the images shown in Figure 5.38, a synthetic scene of a sports car on a street. In the test image shown in Figure 5.38b the car is transposed with respect to the car shown in the reference image, Figure 5.38a.

[a] Erroneous Object Detection A characteristic of erroneous objects is that image errors accumulate in the areas of these objects. We use this characteristic and in a first step compute the average image error as the MSE introduced in Equation 5.23. If objects are misplaced the image error in these areas is above the average image error, while the error is below in other areas. By filtering the error areas with a disk-shaped stencil, object areas can be distinguished. Two things are important to note: First, the object outline is only rough, but covers the whole area in which an object is misplaced with respect to the original. Second, the averaging disk size depends on image size and viewing conditions, to differentiate between noise and relevant objects.

The result of this detection step is a mask with outlined areas. If multiple objects in an image are moved, all of these areas are marked and noted. For the sample images shown in Figure 5.38 the object detection mask is given in Figure 5.39a.

[b] Erroneous Object Matching To match objects between test and reference images there are two possible cases: a transformed object may be overlapping in reference and test image (only one erroneous region detected) or they may be spatially distinct (two erroneous regions). With the additional possibility to have several wrong objects in an image, each region needs to be



(a) Mask outlining Erroneous Objects (b) Environment of Erroneous Object

Figure 5.39: Mask and Environment of Erroneous Object

Figure 5.40: SIFT Matching between Test and Reference Image

matched with itself and with all other error regions. For region matching Scale Invariant Features (SIFT), as proposed by Lowe [51], are employed. For each area detected in the previous step the closest match between reference and test image is recorded.

Figure 5.40 shows detected features between reference (top) and test image (bottom). The translation of the car between test and reference image can already clearly be seen by the feature lines (white) running slightly tilted between both images.

[c] Object Size Calculation The size of a distorted object was observed to be critical for the perceived visual quality. The size of an erroneous object

(a) Filled Background (b) Registered Object

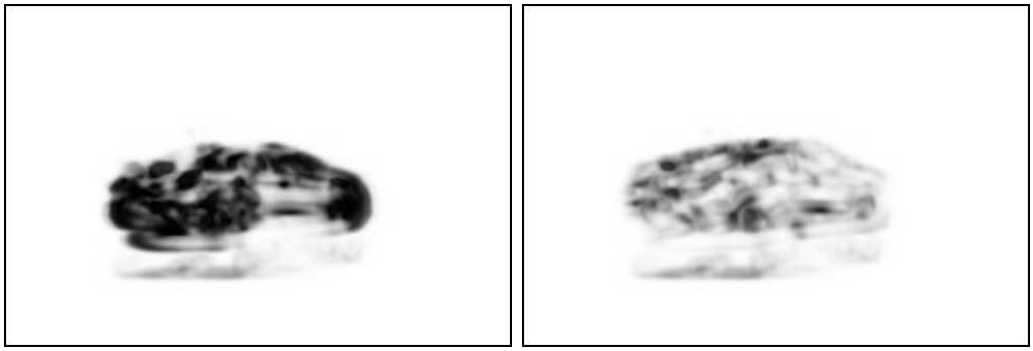
Figure 5.41: Filled Background and Registered Object

is therefore calculated by considering the object masks detected in the *Object Detection* step of two matching areas, as determined in the *Object Matching* step. The object size is given by the average pixel count of both matching object masks.

[d] Environment Structure Analysis A second critical factor in the perceived quality degradation of object transformations is the amount of background structure. Unstructured backgrounds tend to 'hide' object transformations from human perception. To determine the amount of structure that is found in the environment of an object an environment region, that is proportional to the object size calculated in the previous step, is determined. The object environment is given by a boundary of this proportional size around the object mask, as returned from the object detection step. The environment for the sample images from Figure 5.38 is given in Figure 5.39b.

To determine a single environment structure parameter the edge detector proposed by Canny [6] is employed. The amount of edge pixels found by this edge detector is normalized by the size of the structure environment, which is determined analog to the object size calculation above. This allows to have comparable structure parameters across distorted objects of different sizes.

[e] Object Correction Reallocating the distorted object from the test image to its original position in the reference image is an important task to calculate the visual disturbance of the picture irrespective of any transformations. Initially, the misplaced object is removed from the test image and the created hole is filled with a hole filling algorithm. Second, the SIFT feature correspondences are used to get a rough registration of the object in the test image [51]. As SIFT feature matching leaves inaccuracies in the order of single



(a) Original SSIM Error Map (b) SSIM after Registration

Figure 5.42: SSIM Before and After Object Registration

pixels a Levenberg-Marquardt least-square optimization with a Fourier-Mellin transform module is employed afterward to achieve an image registration with sub-pixel precision for exact object placement [108]. The order of applying the SIFT registration before the Fourier-Mellin transform based registration is advantageous, as the SIFT registration works robustly, but with a certain inaccuracy, while the Fourier-Mellin transform becomes unstable for images that are too different from each other but works with a high precision when images are closely aligned already. The proposed implemented concatenation is both robust and precise. Finally, the registered object is fitted onto the filled background image. Filled background image and test image after object registration are shown in Figure 5.41. Next to the registered image this step retrieves the scaling, translation and rotation values between reference and test object.

[f] Residual Error Calculation The residual error of the registered object in the filled image is calculated using SSIM, as introduced in Chapter 5.5.2. Here any other metric (SNR, PSNR, MS-SSIM, HDR-VDP2) could fit in, but SSIM is a widely used and well established metric, better conforming to the human visual system than purely statistical metrics like PSNR. For visualization purpose we show the SSIM maps of the original and of the corrected test image in Figure 5.42. For the image quality assessment we consider the mean SSIM of the map shown in Figure 5.42b.

Calibration

The implementation as described in the previous section returns six parameters describing the test image, which are the residual error e, object size g,

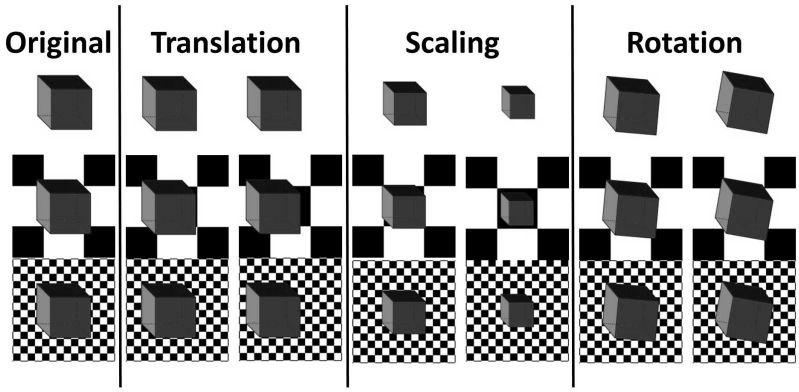


Figure 5.43: Test Images for Calibration

environment structure k, object rotation r, scaling s and translation t. A general relation between these error parameters and the predicted MOS MOS_p is defined, which needs to fulfill the following properties:

- $MOS_p \propto e \wedge MOS_p \propto r \wedge MOS_p \propto s \wedge MOS_p \propto t$
- $MOS_p \propto g \wedge MOS_p \propto \frac{1}{k}$
- if $r = s = t = 0$, then $MOS_p \propto e$
- if $g = 0$, then $MOS_p \propto e$
- if $e = 0$, then $MOS_p \propto r, s, t$

Without any knowledge about weighting or gradient of the different factors, a general model is proposed in which each parameter p may be tuned by a constant factor c_f and a constant exponent c_x, so $c_f \cdot p^{c_x}$. This leads to a general definition of the underlying model, as given in Equation 5.65.

$$\begin{aligned} MOS_p = & c_{f_e} \cdot e^{c_{x_e}} + \\ & c_{f_r} \cdot r^{c_{x_r}} \frac{g^{c_{x_{g1}}}}{k^{c_{x_{k1}}} + 1} + \\ & c_{f_s} \cdot s^{c_{x_s}} \frac{g^{c_{x_{g2}}}}{k^{c_{x_{k2}}} + 1} + \\ & c_{f_t} \cdot t^{c_{x_t}} \frac{g^{c_{x_{g3}}}}{k^{c_{x_{k3}}} + 1} \end{aligned} \tag{5.65}$$

This model has 14 free parameters, which need to be set in a calibration step. For calibration a data-set containing a scaled, rotated and translated

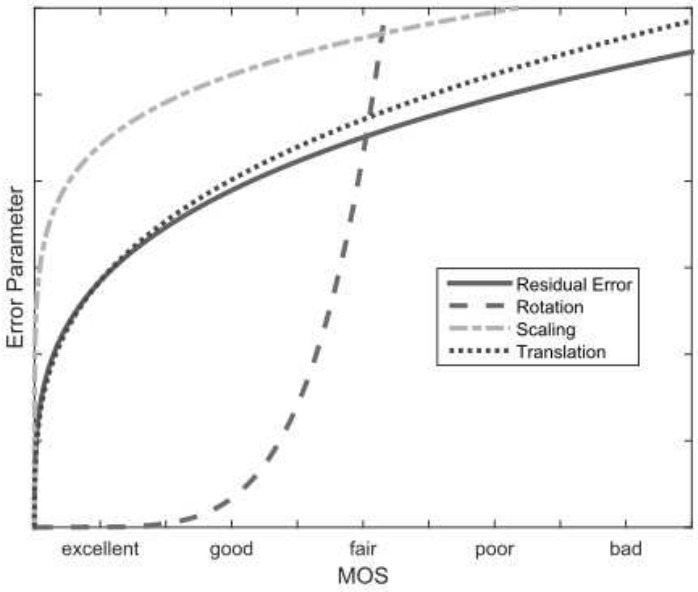


Figure 5.44: Visualization of Free Parameters from Calibration

cube (shown in Figure 5.43) was designed and evaluated by a small group (10 people) of assessors. Based on the obtained MOS the values for the 14 free parameters given above were calculated by fitting the error model in a least square sense. The obtained parameters are, in the order of their appearance in Equation 5.65: $c_{f_e} = 7.2$, $c_{x_e} = 3.5$, $c_{f_r} = 10.3$, $c_{x_r} = 0.2$, $c_{x_{g1}} = 2.0$, $c_{x_{k1}} = 14.6$, $c_{f_s} = 4.5$, $c_{x_s} = 6.5$, $c_{x_{g2}} = 16.9$, $c_{x_{k2}} = 15.5$, $c_{f_t} = 12.1$, $c_{x_t} = 3.2$, $c_{x_{g3}} = 12.6$, $c_{x_{k3}} = 17.8$.

For fixed object size and environment structure the relation between measured error descriptors and MOS based on these parameters is plotted in Figure 5.44. Noteworthy is that small residual errors, translation and scaling remain unnoticed, while small rotations directly lead to a decreased perceived quality. These parameters have then been used to calculate predicted MOS values on the SID2015 database.

5.5.4 Results of SC-VQM

The correlation between the predicted MOS values MOS_p according to the proposed SC-VQM and the MOS values given for the SID2015 database are given in Tables 5.5 and 5.6, where the first gives Spearmans ρ [96], the latter Kendalls τ [42]. Image qualities have also been calculated according to the PSNR metric (Chapter 5.5.1), SSIM Index and HDR-VDP2 Metric (both

Table 5.5: Spearman - Correlation between MOS and PSNR, SSIM, HDR-VDP 2 and SC-VQM

	JPEG	Noise	Transformation	Classical	All
PSNR	$\boldsymbol{\rho = 0.72}$	$\rho = 0.59$	$\rho = 0.31$	$\boldsymbol{\rho = 0.69}$	$\rho = 0.42$
SSIM	$\rho = 0.69$	$\boldsymbol{\rho = 0.64}$	$\rho = 0.36$	$\rho = 0.67$	$\rho = 0.60$
HDR-VDP 2	$\rho = 0.51$	$\rho = 0.56$	$\rho = 0.24$	$\rho = 0.52$	$\rho = 0.37$
SC-VQM	$\rho = 0.69$	$\boldsymbol{\rho = 0.64}$	$\boldsymbol{\rho = 0.46}$	$\rho = 0.67$	$\boldsymbol{\rho = 0.62}$
p-Test	—	—	$\rho = 0.38$	—	—

Chapter 5.5.2) and their correlation to the subjective MOS has been analyzed. It is important to note that the proposed metric SC-VQM employs the SSIM metric to calculate the residual error, since - out of the three metrics - SSIM results in the best correlation to the MOS values of SID2015. In the cases of classical image errors our metric falls back to a pure residual error calculation, which is therefore identical to the values achieved by SSIM.

Adding 14 free parameters to an error model might allow to adjust a model fairly well even to uncorrelated data, thus falsely suggesting an improved correlation. Hypothesis testing is therefore performed by calculating the p-value. Here this is done by employing uncorrelated data (randomly distributed) for the five error descriptors that are returned from our analysis step (translation, rotation, scaling, structure and size). In a next step the 14 free parameters are optimized in a least-square sense for this random data. If it is possible to generate an equally good or better correlation to the MOS values based on randomized input, the approach is necessarily wrong. The correlations for the hypothesis test as described are included in the last line of Tables 5.5 and 5.6. It is important to note that the best fit is achieved with $c_{f_r} = c_{f_s} = c_{f_t} = 0$: having 14 free parameters does not per se result in a better correlation between MOS and MOS_p.

The proposed visual quality metric is especially potent for image distortions due to content manipulations enabled in synthetic contents. The proposed metric analyzes scene objects for transformations, and employs detected transformation parameters as well as the object size and its environment structure for visual quality prediction. The quality model is calibrated with a small set of test images, and the proposed metric is tested on the SID2015 database [P3]. A comparison of correlations between the different metrics shows that SC-VQM increases the correlation between MOS and predicted MOS for transformation errors by 28%. The significance of this result is confirmed by the p-test, which shows that the increased correlation is indeed a result of a correla-

Table 5.6: Kendall - Correlation between MOS and PSNR, SSIM, HDR-VDP 2 and SC-VQM

	JPEG	Noise	Transformation	Classical	All
PSNR	$\boldsymbol{\tau = 0.53}$	$\tau = 0.41$	$\tau = 0.21$	$\boldsymbol{\tau = 0.50}$	$\tau = 0.29$
SSIM	$\tau = 0.50$	$\boldsymbol{\tau = 0.47}$	$\tau = 0.25$	$\tau = 0.48$	$\tau = 0.41$
HDR-VDP 2	$\tau = 0.35$	$\tau = 0.41$	$\tau = 0.17$	$\tau = 0.36$	$\tau = 0.25$
SC-VQM	$\tau = 0.50$	$\boldsymbol{\tau = 0.47}$	$\boldsymbol{\tau = 0.32}$	$\tau = 0.48$	$\boldsymbol{\tau = 0.42}$
p-Test	—	—	$\tau = 0.26$	—	—

tion between the detected translation, rotation, scaling, size and environment structure, as well as the optimized residual error value.

For further validation and plausibility check the motivating image of this Chapter, Figure 5.1 was evaluated with the proposed metric SC-VQM. While SSIM assigns a MOS score of "Fair" to the image ($MOS_p = 3$), SC-VQM evaluated the test image close to "Excellent" ($MOS_p = 4.6$). Intuitively this new estimated MOS of "Excellent" conforms considerably better to the subjective experience than the predicted MOS "Fair".

Chapter 6

Conclusion and Future Work

In the year 1875 the head of the US patent office, Charles H. Duell, resigned and proposed to close the office subordinate to him, as he was certain that everything that could possibly be invented was already invented [49]. History has taught that the opposite is the case: the more inventions are made, the more new possibilities arise. Research in general is very much comparable to the specific case of inventions: serious research can never claim to completely cover or even conclude a topic.

The work presented here introduces, discusses and advances several ideas that may lead towards a (facilitated) synthetic processing of data that is captured in the real world and is perceived by human observers. While many questions are answered, even more interesting problems for future research arise. This chapter summarizes and concludes the previously discussed topics, and identifies some of the questions that arise.

Chapter 1 has introduced the motivation for this work, and in part 1.2 novel capture devices are presented. Especially with respect to data acquisition changes are constant and fast. For color information acquisition resolution and light sensitivity has made tremendous advances throughout the previous 20 years. Other information channels, like depth, thermal information or electromagnetic waves next to visual light have emerged more recently. However, similar to sensors for visible light these other sensors are advancing quickly. During the research time for this work 3D data acquisition has changed from large calibrated multi-camera setups and immobile, slow laser scanners over different research prototypes working with structured light, time of flight or triangulation methods up to reliable end products like the Microsoft Kinect System or the Dell Venue 8 Tablet with built-in Intel RealSense depth camera. It would be overconfident to assume the development of data acquisition methods is saturated at its current state. An important task in the design and implementation of many topics related to computer vision and the advances in

synthetic processing of captured contents is to ensure scalability with new data channels and improvement with better incoming data. Based on data acquisition hardware major future work remains to ensure algorithms and methods not only work but enhance their results as the quality and amount of input data increase.

In Chapter 2 a basis for research on captured and synthetic data is described. Traditionally, captured data and synthetically generated data present two different worlds. Merging both worlds can and should not happen by merging one world into the other. Merging captured data into a synthetic content is - currently - not fully possible. Adding spatial information as well as lighting and material properties are steps into this direction. Merging synthetic information into captured data could be done more easily by rendering the synthetic content to 2D image or video. This, however, would come at the cost of loosing all the flexibility and the post-processing options that are desired. An important task therefore is to create a representation that allows fluent transitions from captured to synthetic data. The SRA introduced in part 2.1 allows both, traditional captured data and fully synthetic objects, as well as any intermediate representation. This representation, which can be fed with data coming directly from a camera, therefore enables a paradigm change in video production. Many effects and creative decisions that before needed to be taken at acquisition time can now be shifted to post-production. The SRA has been implemented and used successfully in the SCENE research project [24]. However, interfaces and import and export functionality has been developed only to an extend where it becomes usable by researchers. With the continuing research on acquisition hardware and novel algorithmic requirements the SRA or a similar representation need constant adjustments to the needs of data representations. On a development side there is the usual gap between a research prototype and an piece of hard- or software that can be used by the community or even professionals.

Additional flexibility cannot come through a representation only. The SRA enables more flexibility, but as a data representation it depends on further information input. Such information can be generated algorithmically. An important step on the path to increasing flexibility is to differentiate objects in a scene. Part 2.2 introduces different approaches that produce object differentiation, or semantic information in a scene. This step makes increasing use of additional data coming from acquisition devices. Differentiating objects in 3D employs spatial information coming from depth sensors or triangulation systems, and object segmentation in 2D can employ additional image channels like near infrared, thermal infrared or depth.

The research results presented with respect to multi-channel segmenta-

tion show that the use of additional information not visible to the human eye enhances the segmentation of semantically meaningful objects. Object boundaries can be invisible in color space but prominent in infrared or depth. Second, use of additional information channels enforces the use of dynamic parameter allocation. Using boundaries known from experiments on only color based segmentation, such dynamic parameter decisions reduce the required human input even compared to only color based segmentation. While the SLIC algorithm proposed by Achanta et al. [2] requires an input image, the number of desired superpixels and the weighting factor m to weight the color information, the presented implementation requires only the input image and the desired number of superpixels. Experiments with respect to robustness allow the conclusion that in presence of noise or changing lighting conditions boundary recall may vary. However, independent of the amount of initial RGB information (clear, noisy, change of brightness) employing additional infrared information yields superior segmentation results compared to segmentations based on RGB only.

Looking at the proposed way of adding additional channel information to the information used for segmentation the idea to add even further information channels suggests itself. Adding any number of further information channels that can be captured can increase segmentation results. An important and difficult question remains of how the different channels can contribute, such that mostly constructive information remains.

Storing data in the SRA is very attractive concerning the possibilities and flexibility that are maintained and created. However, this flexibility comes at the cost of storage requirements. With the goal to optimize flexibility, data is optimally stored in its raw form or even expanded to allow better and easier interaction with the individual elements. Plenty of meta-data that assists creative processes increases the storage requirements of the SRA tremendously.

At the same time the model information contained in the scene compositions offers the potential for storage reduction. With model based encoding an approach is presented that employs the available model information for data size reduction. The results of the experiments described in Chapter 3 show that a hybrid extension for Model Based Coding of MPEG can lead to very promising results. Essential for the compression are the amount of scene content that can be predicted from models as well as the quality of the models. Fully implemented the model based coding scheme has the ability to compress beyond the minimal entropy content of a frame-based video.

Further work is required to combine the model based prediction presented here with the model aided prediction suggested by Galpinab et al. in [26]. Additionally, a combination with the ideas presented by Eisert et al. in [16]

can further increase the coding efficiency: as a 3D model is made available for inter-coded frames already, it is a small step to reuse it for model aided motion compensation without additional costs.

In this context it was observed that prediction from models employing the same error measures as for prediction from previous frames is suboptimal. This is due to a large gap between perceived error and calculated error. Rendered models tend to have a high perceived quality (good resolution, realistic appearance) but a low SNR compared to the original image, as object edges, material color gradients or other details mostly irrelevant to a human observer can differ slightly. This observation is a key motivation for the research described in Chapter 5.

The main reason for a representation that allows both captured and computer generated data is the additional flexibility for post-processing. Many steps that previously required a tremendous amount of manual processing are facilitated and often enabled through the new representation and the paradigm change in visual media production associated with this new representation. The flexibility gain that is described in Chapter 4 is mostly demonstrated in the SCENE research project [24]. With the additional knowledge coming from novel acquisition hardware, additional information added to analyzing algorithms and the new representation scene compositions, object shapes and textures as well as the camera settings for rendering can be modified as post-processing steps. Traditionally, directors decisions which required material not available as footage required a re-shoot of video footage, which is usually associated with huge expenses of money and time. The processing possibilities presented in this chapter allow a multitude of content adjustments that can now be done synthetically at low expenses.

Many of the algorithms presented in this Chapter have been demonstrated successfully, but are still in their fledgling stage. Future work is required to enhance their scope and ease of application as well as to increase their robustness for real life scenarios. At the same time movie directors and artists along the processing chain of movie productions need to be educated about the possibilities that come with these new tools, as well as the requirements that such tools already place on data capture and acquisition. Camera effects, for example, which are now implemented at acquisition time distort data and render many effects impossible. If camera effects are postponed to the rendering stage many tools benefit from the sharper image information and the higher quality of visual data.

Future work is also required with respect to interfaces to the many new tools. Especially for visual information a text-based interface is of little help, as artists need to see the impact of employed effects visually, and preferably

in real time. At the same time the number of possible options and tools for artists increases, which requires smarter and better designed user interfaces to enable the target group of all these effects, creative artists, to work with the designed tools.

Chapter 5 is motivated by the observation that existing image quality assessment metrics are insufficient for scenes containing or fully consisting of synthetic objects. This observation is based on experiments described in Chapter 3 and experiments purely made to test this hypothesis.

Existing databases for image quality assessment metric development and testing are presented, and in this context it is observed that there are no existing image databases to test synthetic scenes or compositions with synthetic elements for pre-rendering errors. Such a database has been created and subjectively evaluated. Subjective testing methods in general and especially the designed test are introduced in detail.

In the following Chapter 5 describes image quality assessment metrics, with focus on full-reference metrics. In this area existing metrics considering image statistics and metrics in addition considering the human visual system are introduced. However, these metrics are insufficient when it comes to quality assessment of images with scene composition errors. For such errors the idea is to model parts of the human cognitive system and employ this model information in the quality metric. The presented approach is - to the authors knowledge - the only approach using computer vision algorithms to model parts of the human cognitive system for enhanced image quality assessment, and significantly enhances the current state-of-the-art.

However, as an initial work in this area it calls for a whole line-up of further questions and tasks. First of all, image quality assessment can be improved with better and more calibration data. In an optimal case more data-sets like the newly introduced synthetic image database are desirable, of which some can be used for metric calibration, others for metric validation.

The error model that is employed is based on a set of assumptions that are formulated in 5.5.3. This error model results in an improved correlation between MOS and predicted MOS, but is not necessarily the best or the correct error model. Medical and psychological studies are necessary to learn more about the correlation between perceived image distortion and object motion, based on object size and background structure.

Finally, geometric transformations cover an important part but not the full range of pre-rendering distortions that are possible in synthetic scenes. Lighting and texture, material properties or vertex normals are other parameters that influence visual results. Evaluating their influence on the perceived reality will be an important task for visual quality predictors suitable for augmented

and virtual reality scenarios.

The problem of merging synthetic and captured worlds to benefit from the best of both worlds remains an interesting and challenging research problem. However, this work presents achievements all along the processing chain of virtual content production that bring video production a small step towards a synthetic world.

Bibliography

[1] Abdelkaher Ait Abdelouahad, Mohammed El Hassouni, Hocine Cherifi, and Driss Aboutajdine. A reduced reference image quality measure using bessel k forms model for tetrolet coefficients. *arXiv preprint arXiv:1112.4135*, 2011.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2012.

[3] Martin Bokeloh, Alexander Berner, Michael Wand, Hans-Peter Seidel, and Andreas Schilling. Symmetry detection using feature lines. In *Computer Graphics Forum*, volume 28, pages 697–706. Wiley Online Library, 2009.

[4] Simon Brown. Averaging ratios: Characteristics of the error of approximation. *World Applied Programming*, 1(5):288–293, 2011.

[5] Chris Budd, Peng Huang, Martin Klaudiny, and Adrian Hilton. Global non-rigid alignment of surface sequences. *International Journal of Computer Vision*, 102(1-3):256–270, 2013.

[6] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 679–698, 1986.

[7] C.R. Carlson and R.W. Cohen. A simple psychophysical model for predicting the visibility of displayed information. In *Proceedings of the SID*, volume 21, pages 229–246, 1980.

[8] Charilaos Christopoulos, Athanassios Skodras, and Touradj Ebrahimi. The jpeg2000 still image coding system: an overview. *Consumer Electronics, IEEE Transactions on*, 46(4):1103–1127, 2000.

[9] Frederique Crete, Thierry Dolmiere, Patricia Ladret, and Marina Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *Electronic Imaging 2007*, pages 64920I–64920I. International Society for Optics and Photonics, 2007.

[10] Yan Cui, Sebastian Schuon, Derek Chan, Sebastian Thrun, and Christian Theobalt. 3d shape scanning with a time-of-flight camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1173–1180. IEEE, 2010.

[11] Theresa M. Culley, Lisa E. Wallace, Karla M. Gengler-Nowak, and Daniel J. Crawford. A comparison of two methods of calculating gst, a genetic measure of population differentiation. *American Journal of Botany*, 89(3):460–465, 2002.

[12] Dennis P. Curtin. Using your digital camera - a guide to great photographs. `http://www.shortcourses.com/use/using1-10.html`, 2011. [Online; accessed 18-Nov-2014].

[13] Edilson De Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM Transactions on Graphics (TOG)*, 27(3):98, 2008.

[14] Johannes F. De Boer, Barry Cense, B. Hyle Park, Mark C. Pierce, Guillermo J. Tearney, and Brett E. Bouma. Improved signal-to-noise ratio in spectral-domain compared with time-domain optical coherence tomography. *Optics letters*, 28(21):2067–2069, 2003.

[15] Rachel Ehrenberg. Digital image founder smooths out pixels. `http://news.discovery.com/tech/apps/digital-image-pixel.htm`, June 2010. [Online; accessed 25-Apr-2015].

[16] Peter Eisert, Thomas Wiegand, and Bernd Girod. Model-aided coding: A new approach to incorporate facial animation into motion-compensated video coding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 10(3):344–358, 2000.

[17] I.a. Essa and a.P. Pentland. Coding, analysis, interpretation, and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):757–763, July 1997.

[18] Mylene CQ Farias and Sanjit K Mitra. No-reference video quality metric based on artifact measurements. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–141. IEEE, 2005.

[19] John A Flueck and Burt S Holland. Ratio estimators and some inherent problems in their utilization. *Journal of Applied Meteorology*, 15:535–543, 1976.

[20] Robert Forchheimer, Olov Fahlander, and Torbjörn Kronander. Low bit-rate coding through animation. In *Proc. Picture Coding Symposium (PCS)*, pages 113–114, 1983.

[21] Renaud Fortuner and Patrick Quénéhervé. Morphometrical variability in helicotylenchus steiner, 1945. 2: Influence of the host on h. dihystera. *Revue Nématol*, 3(2):291–296, 1980.

[22] Rich Franzen. Kodak lossless true color image suite. `http://r0k.us/graphics/kodak/`, November 1999.

[23] King-Sun Fu and JK Mui. A survey on image segmentation. *Pattern recognition*, 13(1):3–16, 1981.

[24] Eugenia Fuenmayor and Adrian Hilton. Novel scene representations for richer networked media. `http://3d-scene.eu/`, November 2014.

[25] Johannes Furch, Anna Hilsmann, and Peter Eisert. A framework for image-based asset generation and animation. *IEEE International Conference on Image Processing (ICIP)*, 2015.

[26] Franck Galpinab and Luce Morina. Computed 3d models for very low bitrate video coding. *Visual Communications and Image Processing*, 2001.

[27] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real time motion capture using a single time-of-flight camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 755–762. IEEE, 2010.

[28] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[29] Natasha Gelfand and Leonidas J. Guibas. Shape segmentation using local slippage analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 214–223. ACM, 2004.

[30] Lutz Goldmann and Mónica Caballero. Depth enabled workflow for flexible 2d and multiview video production. `http://www.3flex-project.eu/`, April 2015.

[31] Rafael C Gonzalez, Richard Eugene Woods, and Steven L Eddins. *Digital image processing using MATLAB*. Pearson Education India, 2004.

[32] Craig Gotsman, Stefan Gumhold, and Leif Kobbelt. Simplification and compression of 3d meshes. In *Tutorials on Multiresolution in Geometric Modelling*, pages 319–361. Springer, 2002.

[33] Tom Haber, Christian Fuchs, Philippe Bekaer, H-P Seidel, Michael Goesele, and Hendrik Lensch. Relighting objects from image collections. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 627–634. IEEE, 2009.

[34] Thomas Hach, Arvind Amruth, Artur Pappenheim, and Johannes Steurer. Photo-realistic synthesis of cinematic lens bokeh. *Proceedings of the European Optical Society Annual Meeting, EOSAM 2014, Berlin*, 2014.

[35] Thomas Hach and Johannes Steurer. A novel RGB-Z camera for high-quality motion picture applications. In *Proceedings of the 10th European Conference on Visual Media Production*, page 4. ACM, 2013.

[36] Robert T. Held, Emily A. Cooper, James F. Obrien, and Martin S. Banks. Using blur to affect perceived distance and size. *ACM transactions on graphics*, 29(2), 2010.

[37] Carl-Herman Hjortsjö. *Människans ansikte och mimiska språket*. Student-litteratur, 1969.

[38] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 19–26. ACM, 1993.

[39] David I Hoult and R.E. Richards. The signal-to-noise ratio of the nuclear magnetic resonance experiment. *Journal of Magnetic Resonance (1969)*, 24(1):71–85, 1976.

[40] Thomas S. Huang and Li-an Tang. 3D model-based video coding: Computer vision meets computer graphics. *Image Analysis Applications and Computer Graphics*, pages 1–8, 1995.

[41] John Bertrand Johnson. Thermal agitation of electricity in conductors. *Physical review*, 32(1):97, 1928.

[42] Maurice George Kendall. *Rank correlation methods*. Charles Griffin and Company Limited, 1948.

[43] Martin Klaudiny and Adrian Hilton. Cooperative patch-based 3D surface tracking. In *Visual Media Production (CVMP), 2011 Conference for*, pages 67–76. IEEE, 2011.

[44] Debarati Kundu and Brian L Evans. Full-reference visual quality assessment for synthetic images: A subjective study. In *Proc. IEEE Int. Conf. on Image Processing*, 2015.

[45] Jong-Sen Lee and Karl Hoppel. Noise modeling and estimation of remotely-sensed images. In *Geoscience and Remote Sensing Symposium, 1989. IGARSS'89. 12th Canadian Symposium on Remote Sensing., 1989 International*, volume 2, pages 1005–1008. IEEE, 1989.

[46] Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. Real-time lens blur effects and focus control. *ACM Transactions on Graphics (TOG)*, 29(4):65, 2010.

[47] Jin Li. Image compression: The mathematics of JPEG 2000. *Modern Signal Processing*, 46:185–221, 2003.

[48] Hantao Liu, Nick Klomp, and Ingrid Heynderickx. A no-reference metric for perceived ringing artifacts in images. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(4):529–539, 2010.

[49] David Louis. *Wussten Sie schon, dass ...?: Erstaunl. Tatsachen aus allen Wissensgebieten*. Heyne-Bücher. Heyne, 1979.

[50] Kok-Lim Low and Tiow-Seng Tan. Model simplification using vertex-clustering. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 75–ff. ACM, 1997.

[51] David G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[52] Jeffrey Lubin. A human vision system model for objective picture quality measurements. In *Broadcasting Convention, 1997. International*, pages 498–503. IET, 1997.

[53] Jeffrey Lubin and David Fibush. Sarnoff JND vision model, 1997.

[54] Marcus Magnor, Prashant Ramanathan, and Bernd Girod. Multi-view coding for image-based rendering using 3-d scene geometry. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(11):1092–1106, 2003.

[55] Twan Maintz. Digital and medical image processing. *Universiteit Utrecht*, 2005.

[56] Rafal Mantiuk, Scott J Daly, Karol Myszkowski, and Hans-Peter Seidel. Predicting visible differences in high dynamic range images: model and its calibration. In *Electronic Imaging 2005*, pages 204–214. International Society for Optics and Photonics, 2005.

[57] Rafał K Mantiuk, Kil Joong Kim, Allan G Rempel, and Wolfgang Heidrich. Hdr-vdp-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions. In *ACM Transactions on Graphics (TOG)*, volume 30, page 40. ACM, 2011.

[58] Rafał K Mantiuk, Anna Tomaszewska, and Radosław Mantiuk. Comparison of four subjective methods for image quality assessment. In *Computer Graphics Forum*, volume 31, pages 2478–2491. Wiley Online Library, 2012.

[59] Pina Marziliano, Frederic Dufaux, Stefan Winkler, and Touradj Ebrahimi. A no-reference perceptual blur metric. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 3, pages III–57. IEEE, 2002.

[60] Sean McHugh. Tutorials: Depth of field. `http://www.cambridgeincolour.com/tutorials/depth-of-field.htm`, 2014. [Online; accessed 18-Nov-2014].

[61] August F. Möbius. Der barycentrische calcül. *Werke*, 1:1–388, 1827.

[62] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer vision and image understanding*, 81(3):231–268, 2001.

[63] Olaf Munkelt, Christof Ridder, David Hansel, and Walter Hafner. A model driven 3d image interpretation system applied to person detection in video images. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 1, pages 70–73. IEEE, 1998.

[64] Hans Georg Musmann, Michael Hötter, and Jörn Ostermann. Object-oriented analysis-synthesis coding of moving images. *Signal processing: Image communication*, 1(2):117–138, 1989.

[65] David R. Nadeau. Volume scene graphs. In *Proceedings of the 2000 IEEE symposium on Volume visualization*, pages 49–56. ACM, 2000.

[66] Alexandros Neophytou and Adrian Hilton. Shape and pose space deformation for subject specific animation. In *3D Vision-3DV 2013, 2013 International Conference on*, pages 334–341. IEEE, 2013.

[67] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, 2(11), 2005.

[68] Hieu Tat Nguyen, Marcel Worring, and Rein Van Den Boomgaard. Watersnakes: energy-driven watershed segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(3):330–342, 2003.

[69] Harry Nyquist. Thermal agitation of electric charge in conductors. *Physical review*, 32(1):110–113, 1928.

[70] Haldun M Ozaktas and Levent Onural. *Three-dimensional television: capture, transmission, display*. Springer Science & Business Media, 2007.

[71] Frederick Ira Parke. A parametric model for human faces. Technical report, DTIC Document, 1974.

[72] Karl Pearson. Mathematical contributions to the theory of evolution.–on a form of spurious correlation which may arise when indices are used in the measurement of organs. *Proceedings of the Royal Society of London*, 60(359-367):489–498, 1896.

[73] Matt Pharr and Randima Fernando. *Gpu gems 2: programming techniques for high-performance graphics and general-purpose computation*. Addison-Wesley Professional, 2005.

[74] John R. Pierce. *Symbols, Signals and Noise*. Harper modern science series. Hutchinson, 1962.

[75] Nikolay Ponomarenko, Oleg Ieremeiev, Vladimir Lukin, Lina Jin, Karen Egiazarian, Jaakko Astola, Benoit Vozel, Kacem Chehdi, Marco Carli, Federica Battisti, et al. A new color image database TID2013: Innovations and results. In *Advanced Concepts for Intelligent Vision Systems*, pages 402–413. Springer, 2013.

[76] Nikolay Ponomarenko, Vladimir Lukin, Alexander Zelensky, Karen Egiazarian, M Carli, and F Battisti. TID2008 - a database for evaluation of full-reference visual quality assessment metrics. *Advances of Modern Radioelectronics*, 10(4):30–45, 2009.

[77] Hendrik Poorter and Eric Garnier. Plant growth analysis: an evaluation of experimental design and computational methods. *Journal of Experimental Botany*, 47(9):1343–1351, 1996.

[78] Michael Potmesil and Indranil Chakravarty. Modeling motion blur in computer-generated images. *ACM SIGGRAPH Computer Graphics*, 17(3):389–399, 1983.

[79] Majid Rabbani and Rajan Joshi. An overview of the jpeg 2000 still image compression standard. *Signal processing: Image communication*, 17(1):3–48, 2002.

[80] Judith Redi, Hantao Liu, Hani Alers, Rodolfo Zunino, and Ingrid Heynderickx. Comparing subjective image quality measurement methods for the creation of public databases. In *IS&T/SPIE Electronic Imaging*, pages 752903–752903. International Society for Optics and Photonics, 2010.

[81] Matthias Reso, Jörn Jachalsky, Bodo Rosenhahn, and Jörn Ostermann. Temporally consistent superpixels. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 385–392. IEEE, 2013.

[82] Iain E. Richardson. *The H. 264 advanced video compression standard*. John Wiley & Sons, 2011.

[83] Jarek Rossignac and Paul Borrel. *Multi-resolution 3D approximations for rendering complex scenes*. Springer, 1993.

[84] David M Rouse, Romuald Pépion, Patrick Le Callet, and Sheila S Hemami. Tradeoffs in subjective testing methods for image and video quality assessment. In *IS&T/SPIE Electronic Imaging*, pages 75270F–75270F. International Society for Optics and Photonics, 2010.

[85] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

[86] Rida Sadek, Gabriele Facciolo, Pablo Arias, and Vicent Caselles. A variational model for gradient-based video editing. *International journal of computer vision*, 103(1):127–162, 2013.

[87] David Sandberg, Per-Erik Forssen, and Jens Ogniewski. Model-based video coding using colour and depth cameras. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 158–163. IEEE, 2011.

[88] Alexander Schick, Mika Fischer, and Rainer Stiefelhagen. Measuring and evaluating the compactness of superpixels. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 930–934. IEEE, 2012.

[89] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In *ACM Siggraph Computer Graphics*, volume 26, pages 65–70. ACM, 1992.

[90] Leila Shafarenko, Maria Petrou, and Josef Kittler. Automatic watershed segmentation of randomly textured color images. *Image Processing, IEEE Transactions on*, 6(11):1530–1544, 1997.

[91] Hamid R Sheikh, Zhou Wang, Lawrence Cormack, and Alan C Bovik. Live image quality assessment database release 2, 2005.

[92] Eero P Simoncelli and William T Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *Image Processing, International Conference on*, volume 3, pages 3444–3444. IEEE Computer Society, 1995.

[93] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *Signal Processing Magazine, IEEE*, 18(5):36–58, 2001.

[94] Olivier Soligon, Alain le Mehauté, and Christian Roux. Towards 3-D model-based video coding. *Annals of Telecommunications*, 53(5):229–241, 1998.

[95] Henry Sowizral. Scene graphs in the new millennium. *Computer Graphics and Applications, IEEE*, 20(1):56–57, 2000.

[96] Charles Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101, 1904.

[97] Gary J Sullivan, Jens Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1649–1668, 2012.

[98] Alain Tremeau and Nathalie Borel. A region growing and merging algorithm to color segmentation. *Pattern recognition*, 30(7):1191–1203, 1997.

[99] Greg Turk. Re-tiling polygonal surfaces. *ACM SIGGRAPH Computer Graphics*, 26(2):55–64, 1992.

[100] Shivashekharayya Viraktamath and Girish V. Attimarad. Impact of quantization matrix on the performance of JPEG. *International Journal of Future Generation Communication and Networking (IJFGCN), Syst. Rev*, 4(3):107–118, 2011.

[101] T Vlachos. Detection of blocking artifacts in compressed video. *Electronics Letters*, 36(13):1106–1108, 2000.

[102] Gregory K Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[103] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.

[104] Zhou Wang, Hamid R Sheikh, and Alan C Bovik. No-reference perceptual quality assessment of jpeg compressed images. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–477. IEEE, 2002.

[105] Zhou Wang and Eero P Simoncelli. Reduced-reference image quality assessment using a wavelet-domain natural image statistic model. In *Electronic Imaging 2005*, pages 149–159. International Society for Optics and Photonics, 2005.

[106] Turner Whitted. An improved illumination model for shaded display. In *ACM SIGGRAPH Computer Graphics*, volume 13, page 14. ACM, 1979.

[107] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, 2003.

[108] George Wolberg and Siavash Zokai. Robust image registration using log-polar transform. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 1, pages 493–496. IEEE, 2000.

[109] Chenglei Wu, Carsten Stoll, Levi Valgaerts, and Christian Theobalt. Onset performance capture of multiple actors with a stereo camera. *ACM Transactions on Graphics (TOG)*, 32(6):161, 2013.

[110] Jiaze Wu, Changwen Zheng, Xiaohui Hu, Yang Wang, and Liqiang Zhang. Realistic rendering of bokeh effect based on optical aberrations. *The Visual Computer*, 26(6-8):555–563, 2010.

[111] Wufeng Xue and Xuanqin Mou. Reduced reference image quality assessment based on weibull statistics. In *Quality of Multimedia Experience (QoMEX), 2010 Second International Workshop on*, pages 1–6. IEEE, 2010.

[112] Zhengrong Yao. *Model-based Coding - Initialization, Parameter Extraction and Evaluation*. PhD thesis, Umea University, 2005.

[113] Song Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(9):884–900, 1996.

Glossary

AVC Advanced Video Coding, Standard for video compression: H.264 / MPEG-4 AVC. 60, 63–69

AWGN Average White Gaussian Noise, Channel Model with normally distributed noise of uniform power. 124

CGI Computer Generated Imagery, synthetically generated images based on 3D computer graphics. 2

CMOS Complementary Metal-Oxide-Semiconductor, technical solution for image sensors. 7

CTU Coding Tree Units, basic processing units of H.265/HEVC. 62

DC Direct Current, Constant part of Data. 58

DMOS Differential Mean Opinion Score, score for subjective evaluations. 89, 104

ESPL Embedded Signal Processing Laboratory, University of Texas at Austin, origin of the ESPL Image Database for image quality metric development and testing. 92, 93, 97

HD High Definition, increase in visual resolution compared to standard definition (SD). 59, 62

HDR-VDP High Dynamic Range Visible Difference Predictor, visual quality assessment metric. 117, 125, 140–142, 149

HEVC High Efficiency Video Coding, Standard for video compression: H.265. 62–64

IDR Instantaneous Decoder Refresh, coded picture to clear contents of reference picture buffer in AVC and HEVC. 66, 67

JND Just Noticeable Difference, minimal difference that is noticeable for human assessors. 138, 139

JPEG Joined Photographic Experts Group, an image compression scheme for bit-mapped graphics. 57, 91–94, 103, 116, 117, 122

JPEG2000 Joined Photographic Experts Group 2000, an image compression scheme for bit-mapped graphics, enhancement to JPEG. 91, 94, 103, 116, 117, 122, 124

LIVE Laboratory for Image and Video Engineering, origin of the LIVE Image Database for image quality metric development and testing. 91–93, 97

MATLAB Matrix Laboratory, Programming Language and Environment. 93

MBC Model Based Coding, encoding scheme for image and video data. 44, 45

MOS Mean Opinion Score, score for subjective evaluations. 89–92, 104, 110, 116, 117, 119, 127, 141, 149–151

MPEG Moving Picture Experts Group, standardization body for video compression formats. 45, 59, 60, 62

MSE Mean Squared Error, signal quality measure based on signal statistics. 125, 144

NAL Network Abstraction Layer, part of AVC and HEVC. 60, 61, 66

OpenCV Open Source Computer Vision, library for computer vision development in C/C++, Java and Python. 19

OpenGL Open Graphics Library, open source library for 2D and 3D graphics applications. 65

PSNR Peak-Signal-to-Noise Ratio, signal quality measure based on signal statistics. 70, 90, 117, 119, 125, 127, 130, 147, 149

RANSAC Random Sample Consensus, algorithm to estimate and match erroneous data. 51

SC-VQM Synthetic Content - Visual Quality Metric, image quality metric employing a model of the human cognitive system. 143, 144, 149–151

SD Standard Definition, collective term for visual resolution common to analog television. 59

SID Synthetic Image Database, image database for image quality metric development and testing. 119, 149, 150

SLIC Simple Linear Iterative Clustering, image segmentation approach. 27

SNR Signal-to-Noise Ratio, signal quality measure based on signal statistics. 67, 90, 125, 127, 128, 130

SRA Scene Representation Architecture, scene representation designed to merge captured and generated content. 12, 17–21, 40, 71–73, 76, 78, 154, 155

SSIM Structural Similarity, image quality metric employing a model of the human visual system. 117, 140, 141, 147, 149–151

TID Tampere Image Database, image database for image quality metric development and testing. 91–93, 97

ToF Time-of-Flight, approach to measure object distances. 8, 9

VCL Video Coding Layer, part of AVC and HEVC. 60, 61

Index

Absolute Categorical Rating, *see* Single Stimulus
Acel, 13
Acel Coherency, 14

Base Layer, 12
Boundary Recall, 22

Coherency, *see* Acel Coherency
Compactness, 24

Differential Mean Opinion Score, *see* Mean Opinion Score
Director's Layer, 16
Double Stimulus, 102, 104

Forced Choice, 102, 105
Full-Reference Metrics, *see* Image Quality Metrics

Hidden Reference, 113
High Dynamic Range Visible Difference Predictor, 138
Human Visual System, 135

Image Quality, 87
Image Quality Metrics
    Full-Reference, 122
    No-Reference, 118
    Partial-Reference, 121
    Reduced-Reference, 121
Image Segmentation, 22

LIVE Image Quality Assessment Database, 89

Mean Absolute Error, 124
Mean Opinion Score, 102
Mean Squared Error, 124
Mesh Approximation, 48
Model Based Coding, 42
Motion Capture
    Marker-based, 6
    Markerless, 6
Motion Scene Camera, 9
Multifocal, 6

No-Reference Metrics, *see* Image Quality Metrics

Oversegmentation, 23

Partial-Reference Metrics, *see* Image Quality Metrics
Peak Signal to Noise Ratio, 124

Quality Ruler, 102, 106

Reduced-Reference Metrics, *see* Image Quality Metrics
Resampling, 47

Sarnoff Vision Model, 136
Scene Layer, 14
Scene Representation Architecture, 12
Segmentation, *see* Image Segmentation
Signal to Noise Ratio, 124
Similarity Judgment, 102, 105
Single Stimulus, 102, 103, 110
Structural Similarity, 137
Subjective Metric, 102

Sum of Absolute Differences, 124
Sum of Squared Differences, 124
Superpixel, 27
Synthetic Image Database, 95

Tampere Image Database, 89
Time-Of-Flight, 8
Triangulation, 6

Undersegmentation, 23

Vertex Clustering, 44
Vertex Decimation, 45