Embedding Metadata in Computer Graphics for Interaction

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von

Dipl.-Ing. Henry Sonnet

geboren am 28.10.1975 in Magdeburg

Gutachterinnen/Gutachter:

Prof. Dr. Thomas Strothotte
Prof. Dr. Heidrun Schumann
Prof. Dr. Jana Dittmann

Ort und Datum des Promotionskolloquiums:

Magdeburg, 25. April 2007

Abstract

Digital media such as 2D images or 3D visualizations often need to be augmented with information that assists the viewer in understanding what is portrayed or that provides additional knowledge which is in relation to the medium's content. The data representation of such descriptive metadata is a challenging task since digital media can be stored in various forms each of which provides, if at all, its own specific method for integrating its assigned metadata. Those methods are analyzed and classified in this research, and the approach of employing digital watermarking which has emerged as an applicable technique for storing descriptive metadata is discussed. In this regard, several *Illustration Watermarking* techniques are proposed, an end-user tool which was implemented is introduced, and the results of evaluating a selection of those watermarking techniques during an experimental study are presented. Besides the internal representation of descriptive metadata, its visual representation is another aspect addressed in this research. Since display space is limited so that descriptive metadata is typically not shown in the original view but on explicit user request, both techniques for guiding the viewer to those regions which are associated with the metadata sought and techniques for eventually presenting the metadata are necessary. Adapting the appearance of the mouse cursor or integrating a down-scaled focus+context view in the original visualization are two techniques which are proposed to provide information that facilitates exploring augmented images. As to metadata presentation, the particular application scenario of integrating textual information in interactive 3D visualizations is dealt with for instance by introducing a specific technique that attaches text labels directly to their corresponding scene objects and by evaluating a selection of techniques in the course of a user study which was conducted.

Zusammenfassung

Digitale Medien wie zum Beispiel 2D-Graphiken oder 3D-Visualisierungen werden häufig mit zusätzlichen Informationen angereichert, die zu einem besseren Verständnis des Bildinhaltes beitragen oder die weitere Details bezüglich der Darstellung vermitteln. Die interne Repräsentation derartiger Metadaten ist mit zahlreichen Schwierigkeiten verbunden, da kein einheitliches Format zur Speicherung digitaler Medien existiert. Darüber hinaus verfügen nur wenige Spezifikationen über geeignete Strukturen, die eine Integration von Metadaten ermöglichen. In der vorliegenden Arbeit wurden Techniken untersucht und klassifiziert, die das Speichern von mit zusätzlichen Informationen angereicherten Medien erlauben. Dazu zählen auch Techniken zum Speichern digitaler Wasserzeichen, obwohl deren Hauptanwendungen auf dem Gebiet des Urheberschutzes zu finden sind. Es stellte sich jedoch heraus, dass diese Techniken durchaus geeignet sind, auch umfangreichere Metadaten direkt im Medium zu speichern. Ein Teil der Arbeit beinhaltet daher zahlreiche sogenannte *Illustrations-Wasserzeichen* Techniken, die entwickelt und zum Teil während einer Benutzer-Studie evaluiert wurden sowie ein Endbenutzer-Werkzeug, das in diesem Zusammenhang implementiert wurde. Neben der internen Repräsentation der Daten ist die visuelle Präsentation von Metadaten ein weiterer in der vorliegenden Arbeit behandelter Aspekt. Da Metadaten aufgrund der begrenzten Darstellungsfläche im Allgemeinen zunächst ausgeblendet werden, sind geeignete Techniken erforderlich, die dem Betrachter signalisieren, wo sich die gesuchten Metadaten befinden und mit denen diese letztendlich auch dargestellt werden können. Eine gezielte Veränderung des Mauszeigers zur Unterstützung der Navigation sowie die Integration einer verkleinerten “Fokus+Kontext” Ansicht sind zwei entwickelte Techniken, mit denen die Erkundung von mit Metadaten angereicherten Medien unterstützt werden kann. Bezüglich der Darstellung von Metadaten wird insbesondere eine Technik vorgestellt, mit der Annotationen direkt mit Objekten einer 3D-Visualisierung verknüpft werden können. Desweiteren wurden verschiedene Techniken zur Integration von textuellen Informationen in interaktiven 3D-Visualisierungen im Rahmen einer Benutzerstudie untersucht, deren Ergebnisse in dieser Arbeit vorgestellt werden.

Acknowledgments

Most of this research was done at the Department of Simulation and Graphics at the Otto-von-Guericke University of Magdeburg during the past nearly four years. During this time, I was supervised by Prof. Dr. Thomas Strothotte to whom I am particularly grateful. We had fruitful discussions, got in touch with companies and other research institutes, and he encouraged me to stay at the Department of Computer Science at the University of Calgary in Canada for a few months which was a great experience for me. In this regard, I would like to thank Dr. Sheelagh Carpendale for making possible this stay and for the helpful discussions we had concerning the research that I did.

I would also like to thank my additional reviewers Prof. Dr. Heidrun Schumann and Prof. Dr. Jana Dittmann for immediately agreeing to examine my research.

I am very grateful to Dr. Tobias Isenberg for our joint work and for the many advices he gave me throughout the work on the dissertation. I would also like to thank Lothar Schlesier for his great support during the work on the *Smage* project. My special thanks also go to Andrea Unger, Silvio Lange, Nguyen Dinh Quyen, and Michael Specht for their significant contributions to my work. Parts of this thesis have been published in collaboration with Dr. Sheelagh Carpendale, Dr. Wallace Chigona, Prof. Dr. Jana Dittmann, Dr. Knut Hartmann, Dr. Tobias Isenberg, Silvio Lange, Dr. Felix Ritter, Lothar Schlesier, Prof. Dr. Thomas Strothotte, Andrea Unger, and Thomas Vogel, who therefore deserve my sincere thanks.

I would like to thank Timo Götzelmann, Niklas Röber, and Martin Spindler who shared the office with me and with whom I had interesting discussions and who were the first contacts I could ask when I had problems regarding my work. I should not forget Joachim Böttger with whom I had interesting discussions as well and who helped me in many situations. I also wish to thank Anand Agarawala, Michael Boyle, Carman Neustaedter, Stacey Scott, Anthony Tang, and the other members of the Interactions Research Lab at the University of Calgary. We had a great time at work and after work.

I am indebted to Petra Schumann, Petra Specht, and Beate Traoré for the many questions they have answered and the many problems they have solved. I would also like to express my gratitude to Heiko Dorwarth, Dr. Volkmar Hinz, and Thomas Rosenburg for always giving their best to keep the technique running.

Last but not least I would like to thank my girlfriend Anja, my brother Christian, and my parents for their support whenever I needed it and for reminding me that there is much more in life than doing research and writing a thesis.

Contents

1 Introduction

A statement by Bertin [Ber83] hints that any medium can serve a purpose that we are not able to think of at a certain point in time or that we are not capable to perceive:

> "One had to wait until the fourteenth century to suspect, at Oxford, and until the eighteenth century to confirm [...] that the two dimensions of a sheet of paper could usefully represent something other than visible space. This amounts to a transition from a simple representation to a 'sign-system' that is complete, independent, and possesses its own laws ...".

Now we are amidst, or better, at the beginning of the computer era and have only a vague imagination of what is possible with such technology. Providing virtual information is only one issue amongst many others in this context. But only a fraction of the information available can eventually be shown on a display device since the display space is limited. A viewer has thus often no idea about the whole data which is available for display. Assistance in visual data exploration is hence required. Also, to empower a viewer to explore the data, which can consist of various components, it must be stored in a way which allows handling its components separately while the link between these components is maintained.

The research at hand deals with both aspects of appropriately storing the data that is to be displayed and aspects that enable a viewer to visualize parts of that data selectively. These two important areas in computer science, which divide this work in two parts, are referred to as *Data Representation* and *Visual Representation* (compare with [Car99]).

Data Representation

According to Marr [Mar82], representation is a formal system that covers certain types of entities as well as the manner in which those entities are interpreted. The number thirty-seven can serve as an example: it can be represented, for example, in the Arabic ($3 \times 10^1 + 7 \times 10^0$), binary (00100101), or Roman numeral system ($XXXVII$).

With respect to computer technology, the binary numeral system is used virtually by every hardware component. However, the manner in which binary data is eventually interpreted can provide diversified information. It can, for instance, be interpreted as color information which, in the form of a picture, can provide information such as a snapshot of the real world. Or it can be interpreted as *ASCII* code which represents text. And even applying different interpretation techniques to the same binary data can yield meaningful information, as this work will demonstrate.

Visual Representation

According to the interpretation technique applied, binary data can be visualized so that the information represented by the binary data can be adequately communicated (see Fig. 1.1). Visual representation offers hence "a method for seeing the unseen" [MDB87]. Since the "unseen" typically exceeds the display space which is available, techniques are necessary that help a viewer to (1) navigate through the "unseen" and (2) to manage the content which is eventually displayed. Those aspects will be addressed in the second part of this work.

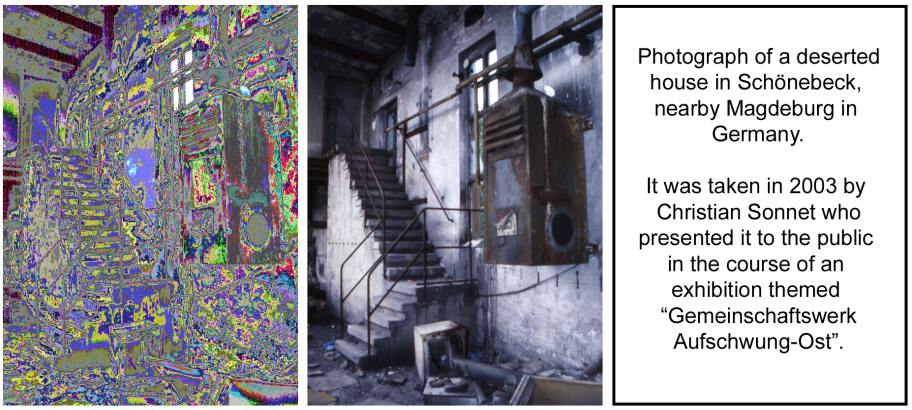


Figure 1.1: *Left: Binary data can be visualized as color values. Center: When the binary data is interpreted in a different way, that is, the first four bits in each color channel are exchanged with the last four bits, the represented photograph appears. Right: When specific bits in each color channel are then interpreted as ASCII code, an encoded text can be extracted.*

Interrelation between Data and Visual Representation

Data representation can vary in its complexity. Displaying simple text, for example, requires only a sequence of bytes which can be interpreted as characters. If the text, on the other hand, contains words in different styles and colors, data representation is far more complex and text display requires more specific software. The same applies to images for which information beside their colors is available (e. g., image captions or annotations). There is hence a strong correlation between data complexity and the requirements to the software which serves to visualize the data. The challenge in this regard is to represent the data such that common software enables an end-user to explore the data in its entirety.

1.1 Aim of this Work

In the field of data and visual representation, numerous, diversified techniques are already in use. Even though some of those techniques have more or less weak points, the aim of this work is not to present the *one* new technique that is to replace other existing techniques.

This work rather aims at suggesting solutions which can coexist with others which all have their assets and drawbacks.

Some of the issues addressed in this regard can be associated with Figure 1.1. The second image, for example, is linked with a description. But image and text are differently represented. The image typically consists of format-specific notations and a matrix of *RGB* color values whereas the text is basically composed of a sequence of characters. The question is how can these two types of data be appropriately combined? This question and aspects such as what is meant by "appropriately" and what role does digital watermarking play in this regard will be addressed in the course of this work.

Besides aspects of data representation, Figure 1.1 also indicates some of the challenges associated with the visual representation of data. The second image shows a work of art. But even though there is text available that describes what is depicted, no hints for its availability are provided to the viewer. A second problem that becomes obvious is that to visualize the description eventually, a separate image is employed, namely the text view in Figure 1.1(right). But display space is limited in most situations. Adequate solutions for integrating several visualizations of data within the same display area are thus necessary as well as techniques for assisting a viewer in exploring the data.

1.2 Problem Analysis and Concept Formation

Adequate data representation requires to analyze all the components which are part of the medium and which can be visualized at a particular time with or without the need of end-user interaction. In this regard, two types of data layers can be identified which all media that can be explored have in common: the ***Background Layer*** and the ***Illustration Layer***. The *Background Layer* includes the data which is naturally displayed. It can be augmented with auxiliary data included in the *Illustration Layer*, which is typically caused by the end-user.

Various techniques exist which can be used to store *Background* and *Illustration Layer* data. Depending on whether they distribute the data across multiple data files, the techniques can be divided in two basic classes: techniques for ***Integrated Data Storage*** and techniques for ***Separated Data Storage***. However, each of these techniques has not only advantages but is also associated with problems such as inadequate variety of data types which are supported, intricate data management, or missing standardization of storage facilities.

Illustration Watermarking is introduced as an alternative concept for layer data storage. It follows the idea of digital watermarking in that it exploits the data structures of the original medium (*Background Layer*) to embed the data of the *Illustration Layer* after its appropriate preparation. To improve efficiency, the *Background Layer* is analyzed and a ***Capacity Map*** is generated which indicates for each part of the *Background Layer* to what extent it can be modified for data insertion.

Data representation is one aspect. The other aspect concerns providing the information which is available but not necessarily visible at the moment. This is due to the display space being limited, which causes visual representation to be a challenging task. Amongst others,

it comprises assisting the viewer in exploring the unseen information of the *Illustration Layer*. To this end, ***Metadata-Related Information*** is shown. It indicates which regions of the *Background Layer* are associated with data of the *Illustration Layer*. Furthermore, it provides hints about type, content, and amount of the information which is available. Providing *Metadata-Related Information* can hence be regarded as navigation and decision guidance before the actual information sought is requested and eventually displayed.

1.3 Overview of Results

Results of this work can basically be attributed to three categories: overview and classification of existing work, introduction of new concepts and techniques, and evaluation of a selection of techniques by experimental studies.

- ***Layer subdivision of a graphic's content.*** *Background* and *Illustration Layer* are defined as the elementary components of a computer graphic that provides auxiliary information which is typically invisible at a particular time, but which can be requested by the viewer at will.

- ***Classification and evaluation of techniques for interrelated data storage.*** Various techniques for *Integrated* or *Separated Data Storage* exist which can be employed to represent a medium and the information which semantically correlates with that medium. Those techniques are classified and evaluated with regard to basic requirements for appropriate layer data storage. The evaluation yields that each technique has assets and drawbacks, which strongly depends on the application.

- ***Illustration Watermarking for data storage.*** The concept of *Illustration Watermarking* is introduced as an alternative approach to store a medium and its associated information. *Illustration Watermarking* is compared with other techniques and its assets and drawbacks are discussed.

- ***Illustration Watermarking techniques for vector graphics and 3D data.*** Various techniques for embedding additional information in vector graphics and in 3D model data were proposed and published ([SIDS03, SL05]). It is shown that those techniques are capable of imperceivably storing considerable amounts of data, which makes them applicable for *Illustration Watermarking*.

- ***Introduction and study of Illustration Watermarking techniques for raster graphics.*** The proposed techniques range from traditional watermarking techniques which were adapted for the purposes of this work to new techniques. *Capacity Map* computation and its inclusion during the watermarking results in more effective data insertion, which was verified by a user study (see [SUS+06]).

- ***Smage—a tool for annotating images.*** An end-user tool, which is called *Smage*, was devised and made available for public download (see Appendix C). This tool allows to select arbitrary regions in an image and to assign information such as texts,

images, or dictionary keywords to those regions. Each information item is encoded into its corresponding image region as *Illustration Watermark*.

- ***Tools for assisting a viewer in information exploration.*** One of the prototypes implemented in this regard is the *Meta-Previewer*. It guides a viewer to those regions in an image which have additional information associated before it provides further ***Metadata-Related Information***. In another approach, the mouse cursor is adapted so that it points into the directions of interest while its shape and color reveal further information.

- ***Integration of 3D data and text: approaches and study.*** Text positioning in 3D scenes that can be explored by moving the camera is challenging with respect to clear object-text correlations, text readability, and scene occlusions. One approach that addresses those aspects attaches text labels directly to those scene objects the text refers to (see [SCS04]). The usability of this approach and related approaches was evaluated in the course of a user study (see [SCS05]).

1.4 Thesis Structure

In this section, each of this work's chapters is outlined in terms of its objectives and how these were eventually realized.

Chapter 2: Linking Media with Descriptive Metadata

This research deals with computer graphics which are associated with auxiliary information which is naturally invisible. To enable a viewer to request this information, computer graphic and auxiliary information must be stored in a way such that both components can be handled separately while the link between them is maintained. Several techniques exist which can serve this purpose of linked data storage. But what are the differences between those techniques and in which situations are they not applicable?

The results of this chapter are twofold. First, explorable computer graphics are analyzed and it is revealed that these typically consist of two basic types of layers: *Background Layer* and *Illustration Layer*. It then analyzes and compares techniques which can be exploited to store these components. In this regard, the concept of *Illustration Watermarking* as a sub-category of digital watermarking is introduced and discussed.

Chapter 3: Illustration Watermarks for Geometrical Data

Depending on the application of a computer graphic, it can be stored in different ways. One option is to represent the graphic by a sequence of descriptions such as regarding its geometry and color. With respect to *Illustration Watermarking*, the auxiliary information can thus only be embedded in those descriptions.

The computer graphics addressed in this chapter are divided into vector graphics and 3D model data. For both types, features of their descriptive representations are analyzed as to whether they are capable of storing parts of the whole watermark data. In case of vector graphics, considerable amounts of data can, for instance, be stored by varying the lengths of their line segments. For 3D model data, an approach is proposed that inserts the data by watermarking specific *Carrier Objects* which are then placed inside the original model.

Chapter 4: Illustration Watermarks for Raster Graphics

Raster graphics, which are represented by a 2D matrix of pixels, can be watermarked by modifying the color channels of a pixel. To avoid introducing perceivable changes while considerable data amounts are encoded, aspects of human perception must be considered and image texture analysis is necessary.

This chapter proposes various watermarking techniques that operate in different color spaces or that transform the image into its wavelet representation before it is watermarked. Also, *Capacity Maps* are generated and included that suggest data encoding rates for each pixel's color channels.

Chapter 5: Evaluation of Illustration Watermarking Techniques for Raster Graphics

The bulk of research in the field of digital watermarking targets raster graphics. However, the effectiveness of proposed techniques with regard to perceptibility is, in most cases, solely demonstrated based on theoretic considerations. To determine the limits of imperceivably modifying different types of images, experimental studies are essential.

This chapter describes a user study with 112 participants which was conducted to identify those limits with respect to image type and to evaluate a selection of the techniques proposed in this work. The study confirmed, for instance, that when image texture analysis was included in the watermarking process, the changes introduced during watermarking were less perceivable.

Chapter 6: Interaction with Embedded Data

The viewer should be enabled to explore the available information in its entirety. To this end, attention must be attracted in certain situations and the information demanded must be appropriately presented. What techniques do exist and how can the viewer be assisted to gather the information sought are questions which have to be answered in this regard.

In this chapter, techniques for attracting a viewer's attention are introduced and approaches for providing ***Metadata-Related Information*** are proposed. After this, the visual integration of 3D data and text is analyzed. In this regard, requirements are itemized, existing techniques are surveyed, and a new approach is proposed. In addition, a user study with 36 participants that was conducted is described.

Chapter 7: Application Scenarios

The main part of this research deals with *Illustration Watermarking* techniques which can be employed to store auxiliary information, which can be demanded by the viewer, in different types of computer graphics. But what are potential application scenarios for those techniques?

This chapter describes, in particular, an end-user tool which can be used to enrich arbitrary images with information that correlates with the image's content. It also describes scenarios in which parts of the image content can be animated or controlled by the viewer (e. g., *Gameable Images*). In both cases, the whole information required is encoded as *Illustration Watermark*.

Finally, ***Chapter 8*** summarizes the contents of this work and contains concluding remarks.

2 Linking Media with Descriptive Metadata

Digital media such as images are often exploited to communicate issues which cannot be adequately expressed in words. But even though it is said that a picture is worth more than a thousand words, the words which are essential to communicate a certain message can be missing. This is due to the fact that the message that a viewer eventually receives strongly depends on how the viewer interpretes the content of the image. To avoid potential ambiguities, images are often enriched with information that facilitates communicating the intended details.

Besides displaying information, another aspect, which is dealt with in this chapter, concerns the representation of this information. In this regard, Section 2.1 focuses the basic requirements which are necessary so that a viewer can explore a medium and its associated additional information. Section 2.2 then addresses techniques which can be used to store explorable media. This includes digital watermarking (Sect. 2.3), a technique which has its main application in the field of copyright protection. Hence, the concept of *Illustration Watermarking* is introduced in Section 2.4. Section 2.5 then compares the discussed techniques, before Section 2.6 concludes this chapter.

2.1 Illustrative Media and their Components

The term *Digital Medium*, as it is used in this work, refers to a digital format in the computer graphics domain that represents certain information. The manner in which this information is eventually provided to the viewer strongly depends on the graphics format chosen. In this regard, a computer graphics medium can, for example, be represented as a matrix of pixel values (e. g., 2D raster graphic) or as a sequence of descriptions (e. g., 3D geometric models and 2D vector graphics).

2.1.1 Digital Media as Communication Tools

In many situations, computer graphics are intended to represent information visually with the objective of communicating a certain message. However, since the information displayed often does not suffice to adequately communicate the whole message, the inclusion of auxiliary information is required. That way, arbitrary computer graphics can be converted into *Illustrative Media*.

What is characteristic for illustrative media?

As an example, an article about medicinal herbs in which the dandelion is one of the herbs focused may help to answer this question. When an image of a dandelion such

as the image in Figure 2.1 is included in this article, the image automatically becomes an illustration since it helps the reader to picture dandelions. However, the image alone is solely a snapshot of the nature. According to Walther [Wal96], there is interaction between texts and images (illustrations) in which images emphasize the expressiveness of a text and set priorities. The author hence regards illustrations as an addition to texts. In this work, however, an illustration's scope of application is expanded.

Figure 2.1: *Photograph of a dandelion.*

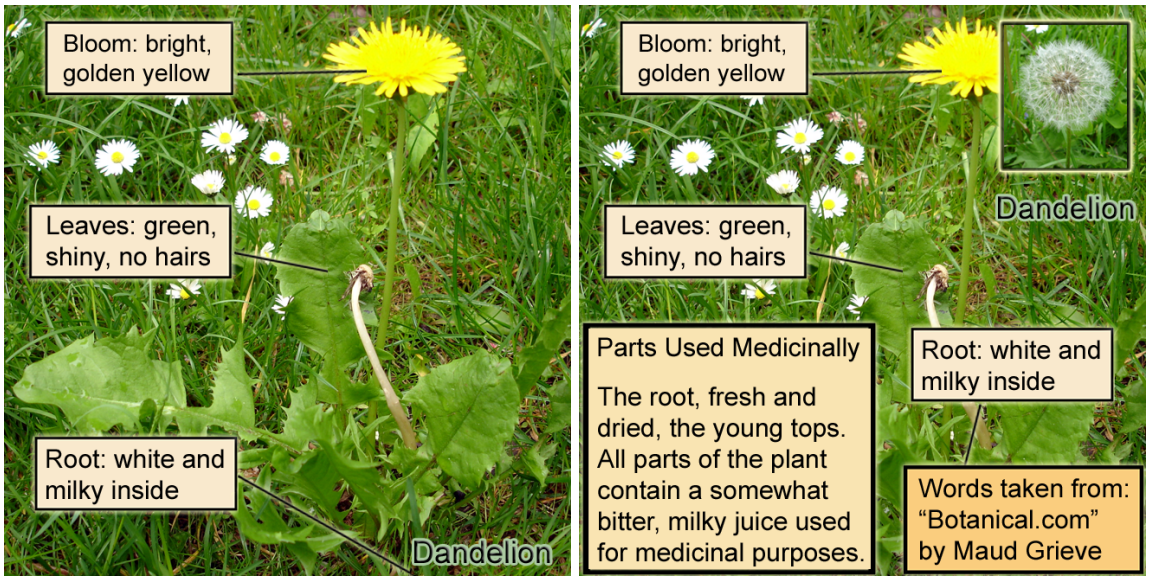


Figure 2.2: *Left: A photograph augmented with textual elements that correlate with the photograph's content becomes an illustration. Right: Too many inserted details, on the other hand, result in information overflow and involve considerable occlusion.*

Besides including a graphic in text by which the non-illustrative graphic becomes illus-

trative, the conversion between illustrative and non-illustrative media can be provoked by other aspects, as well. The images in Figure 2.2 illustrate this issue. Here, alone the inclusion of text phrases into the image in Figure 2.1, which is an ordinary photograph, signalizes that the author intends to illustrate something. As a result, the photograph is converted into an illustration. This observation allows to deduce the following characteristics of an illustration:

- A certain kind of information shall be unambiguously communicated. For this purpose, a specific graphic (e. g., draft, drawing, etc.) is designed or an existing graphic is augmented with data that contributes to adequately communicating the information.
- The illustrative elements can be clearly identified and, typically, correlate with the content of the original graphic.

Descriptive Metadata

A basic component that distinguishes an illustration from a non-illustrative graphic is the data that was added to assist in the communication process. Such data can be classified as a subcategory of *Metadata* which is data that describes other data to enhance its usefulness [Mar98] or that provides additional information about the contents, creation, or uses of a medium [Dig00]. According to Brown and Shepherd [BS95], metadata is categorized into interpretation data, attribute data, and documentation data.

However, the term that henceforth is used to refer to the data which is specific to illustrations was introduced by Baeza-Yates and Ribeiro-Neto [BYRN99]. They term metadata, which is associated with text and which typically includes the author, the date of publication, the source of publication, the document length, and the document genre, as *Descriptive Metadata*. In the research at hand, the scope of descriptive metadata is broadened such that it not only applies to text but to any computer graphic that will be addressed.

2.1.2 The Layers of an Illustrative Medium

The image in Figure 2.2(right) is an example for both a graphic which was converted into an illustration and a graphic that is overloaded with descriptive metadata. To avoid the second aspect, passing the graphic from the "dead image" to a "living image" can be a solution [Ber83]. This can be achieved by dynamically showing or hiding illustrative elements, which requires to handle original graphic and descriptive metadata separately.

Background and Illustration Layer

The content of an illustrative medium can be divided in two types of layers: the *Background Layer* and the *Illustration Layer* (see Fig. 2.3). The *Background Layer* is the information which is represented by a graphic when no descriptive metadata is present. The *Illustration Layer*, on the other hand, includes the descriptive metadata. It also includes *Structural Metadata* [Wen99] which links *Background* and *Illustration Layer* components and which provides information about the manner in which descriptive metadata is to be displayed.

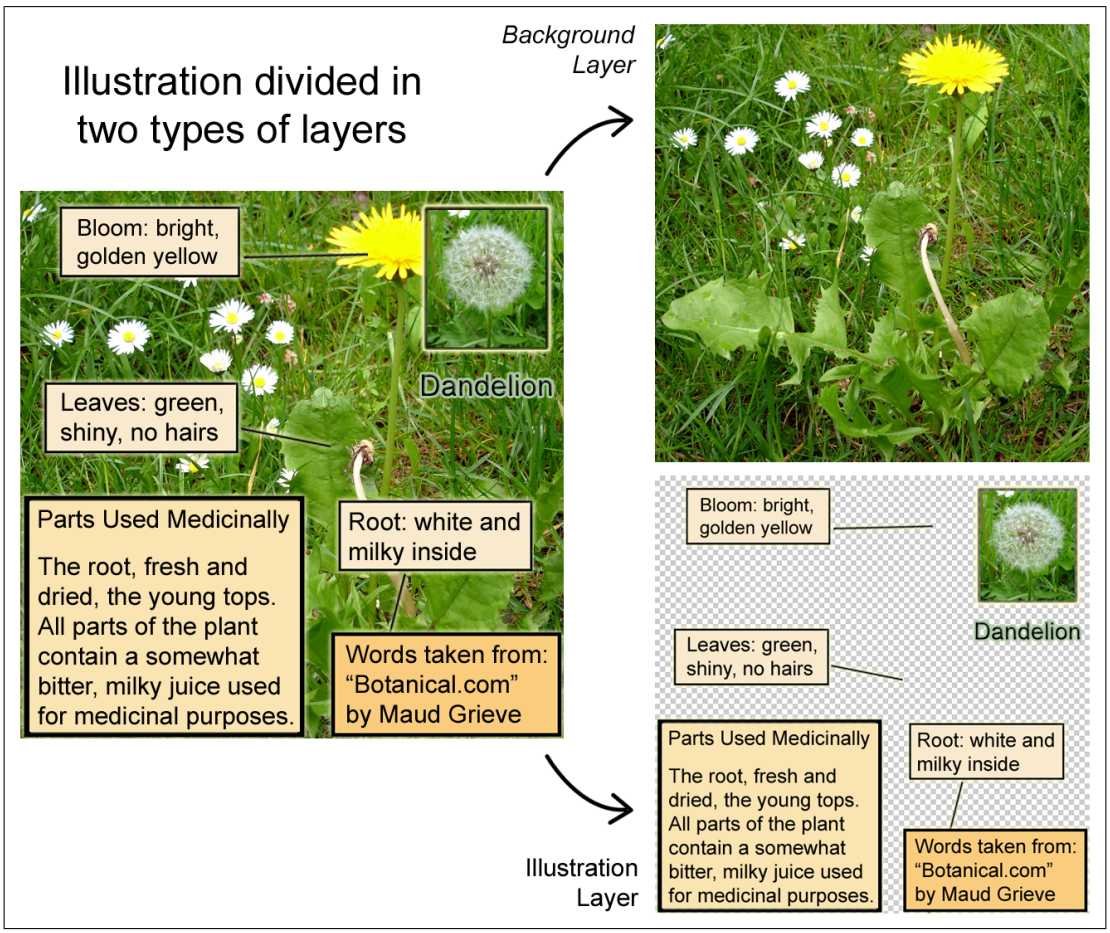


Figure 2.3: *An illustration is composed of two layers. The* Background Layer *is the pure image whereas the* Illustration Layer *contains data that provides further information.*

The two layers concept can generally be applied to every kind of computer graphic that has descriptive metadata. A 3D geometric model, for example, can be rendered according to its geometric descriptions and material properties. The rasterized image which results from rendering the model has no descriptive metadata and hence, can be considered as the *Background Layer*. The *Illustration Layer* then consists of all the data that was added to the model to explain what is displayed by the image. Examples are textual descriptions, descriptive images, and even auditory data.

With respect to images, Wu et al. [WKLH00] identified four levels which can be compared with the concept of the two layers. The first, which is the *Image Level*, can be associated with the *Background Layer*. The two levels, which can be considered as a subcategory of the *Illustration Layer*, are the *Segmented Image Level* and the *Descriptions & Measures Level*. Those levels are interconnected with each other by labels that link a segmented image region with its descriptions and feature measures. The fourth level, which comprises previous knowledge that allows to interprete the image, is the *Interpretation Level*. Needless

to say that this level is a precondition for interpreting *Background* and *Illustration Layer*, as well.

2.1.3 Layer Subdivision in Related Work

Even though digital-only media are addressed in this work, the idea of linking a medium with descriptive metadata is pursued in other domains, as well, and analyzing them may hence help to gain further insights.

Hybrid approaches are an example. "Hybrid" means in this context: a medium—present in real life—is in some way branded with markers that link the medium with information which is digitally available. Perhaps, one- or two-dimensional barcodes on consumer products are the most popular example for metadata carriers in this domain.

Dymetman and Copperman [DC98] itemize a number of reasons for why it may be necessary to juggle real life media and virtual data. They try to bridge the gap between these two worlds with their *Intelligent Paper* approach. Sheets of intelligent paper are standard paper sheets with marks imprinted. Those marks, which are visible only to specific pointers, represent unique IDs and local coordinates. Once a pointer is connected to the Web, arbitrary information assigned by the paper's author becomes available.

Another approach, which is called *Paper++* (e. g., [NS02, NS05]), also reflects the layer concept. Here, various virtual layers—arranged on top of a virtual copy of a printout—include areas which are linked to database items. To query the database, a user can point at a printout with a wand device that reads information from imprinted barcodes.

An alternative to traditional barcodes are so-called *DataGlyphs* [Hec01]. *DataGlyphs* are machine-readable stroke patterns imprinted on paper documents. A stroke's orientation indicates whether a bit is set or not. To extract the data encoded, *Xerox PARC* developed the *Glyph-O-Scope* system which records and processes the printed document. Thereafter, it displays the encoded information, digitally overlaid on the document.

2.2 The Challenge of Layer Representation

The previous section revealed that creating a "living" illustration involves subdividing the illustration's content into layers. The challenge now consists in finding an appropriate technique to store *Background* and *Illustration Layer*. To this end, it is essential to determine each layer's potential data components.

Background Layer data

The *Background Layer* consists of the data that is to be illustrated. This can be a 2D matrix of pixels (raster graphic) or a description whose interpretation yields a graphic which is then displayed.

Illustration Layer data

The *Illustration Layer* data can be rather complex, which requires to be careful structuring it. Aspects to be considered are, for example:

- The *Illustration Layer* can consist of several components. In Figure 2.2, for example, each text box is an individual *Illustration Layer* component. To allow for handling those components separately, a sublayer for each component is required.
- The *Illustration Layer* components can be diversified types of data. These can range from textual descriptions and raster graphics (as in Figure 2.2(right)) over instructions (e. g., when to display a component) to auditory data.
- If the *Background Layer* is partitioned, each partition must be linked with its associated *Illustration Layer* component.

2.2.1 Basic Requirements for Layer Storage

According to Leung and Sutanto [LS99], a "good data model must take into account the semantic richness of the underlying information, and should be applicable to a wide range of application domains".

This particularly applies to *Illustration Layer* data for which a wide range of storage techniques exists. However, not all of them fulfill criteria that make them applicable to serve the purpose addressed in this work. The most important of those criteria are:

- ***Accessibility.*** Typically, a certain message shall be communicated with an illustration, which is facilitated by the descriptive metadata included in the *Illustration Layer*. The layer with its sublayers must hence be stored so that it can be used in its entirety and at will.
- ***Compactness.*** With an increasing number of sublayers, the diversity and amount of the data to be handled increases, as well. Reducing the number of layers to the essential and compactly storing them eases their handling.
- ***Low Storage Costs.*** What can be assumed when adding an *Illustration Layer* to its *Background Layer* is that the illustration's storage costs increase proportional to the size of the *Illustration Layer*. However, techniques which yield lower storage costs than those which would result from adding the two layers are preferred.
- ***Compatibility.*** Allowing to explore any illustration using common software is hard to accomplish, even though this is desired. However, alone *Background Layer* data compatibility with established graphics file formats is better than an illustration's complete incompatibility.

The perfect "illustration file format" that satisfies all these criteria does not exist. However, there exists software with which very complex and interactive illustrations can be generated. But to enable end-users to explore those illustrations, the installation of specific software is required.

On the other hand, graphics file formats established that allow for storing descriptive metadata. Unfortunately, the diversity of the metadata supported is only a fraction of the diversity an *Illustration Layer* may include. And as soon as the metadata's complexity increases, specific software for its interpretation is necessary.

2.2.2 Survey and Classification of Storage Techniques

This paragraph addresses existing techniques which can be employed to store *Background* and *Illustration Layer*. The aspect of eventually enabling an end-user to interact with the illustration is disregarded at the moment because it basically depends on the software system that handles the illustration.

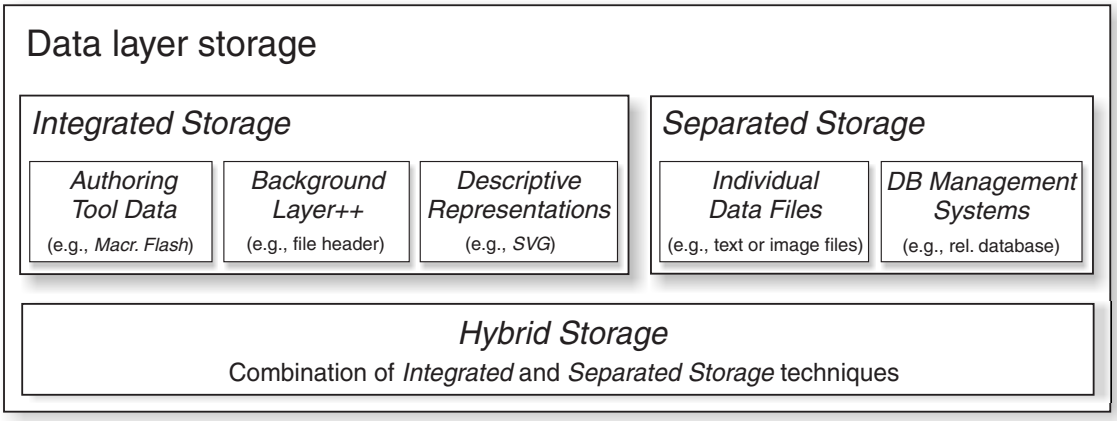


Figure 2.4: *Classification of layer storage techniques.*

Depending on whether the individual layers are stored separately or included within the same data file structure, the two alternatives, which are distinguished in this work, are *Integrated Data Storage* and *Separated Data Storage*. A third alternative referred to as *Hybrid Data Storage* uses elements of both *Integrated* and *Separated Data Storage* (see Fig. 2.4).

Integrated Data Storage

Illustrations whose entire data is included within a single data file are classified in this category. Further subdivision yields illustrations that were created using *Authoring Tools*, illustrations whose *Illustration Layers* become part of their *Background Layers* (referred to as *Background Layer++*), and illustrations the contents of which are *Descriptively Represented*.

- ***Authoring Tool Data.*** Software such as *Macromedia Flash* (e. g., [RD04]), *ToolBook* (e. g., [Sum04]), or *MS PowerPoint* (e. g., [Wem03]) can be employed to create interactive illustrations which are highly complex. The final illustration can then be stored as a single data file (*Macromedia Flash: file.swf*, *MS PowerPoint: file.ppt*, *ToolBook: file.tbk*). However, displaying and interacting with those illustrations requires specific software.
- ***Background Layer++.*** The concept of *Background Layer++* refers to storage techniques that integrate *Background* and *Illustration Layer*. In this regard, a number

of graphics format specifications provide a certain amount of space for metadata storage. The graphics file format *JPEG2000*, for example, allows to include metadata elements (*XML boxes*) in *JPX* files (e. g., [Joi04a, JW04]). Other file formats provide *private tags* (*TIFF*, described in [Ado92]) or *private text chunks* (*PNG*, specified in [Joi04b]). In the medical domain, the *DICOM* standard [DIC04] is widely-used for handling, storing, and transmitting patient-relevant data.

- ***Descriptive Representations.*** The data of illustrations classified in this category is basically stored as a sequence of well-confined textual descriptions. The *Extensible Markup Language (XML)* can be considered as the most popular standard underlying those descriptions. Examples are *Scalable Vector Graphics (SVG)* [Wor03] as a specification for handling 2D graphics and graphical applications using XML, or *Extensible 3D (X3D)* (e. g., [GC04]) as a 3D file format that facilitates delivering 3D data and other information.

  In contrast to the concept of *Background Layer++*, descriptive languages focus descriptions in words rather than pixel representations. However, even raster graphics can be formed so that they can be included in *Descriptive Representations* (e. g., [AT04]).

Separated Data Storage

Separated Data Storage refers to storage techniques for illustrations whose layers are not included within a single data file. Besides the basic principle of handling *Individual Data Files*, the *Background* and *Illustration Layer* can also be represented in *Database Management Systems*.

- ***Individual Data Files.*** A simple example are Web pages which include text and image data, and which were generated using *Dynamic HTML* (e. g., [Goo02]) or *Synchronized Multimedia Integration Language (SMIL)* [Wor05]. These languages not only allow to structure the data appropriately, they also provide concepts for interaction.

  However, the difference to *Descriptive Representations* is that *DHTML* or *SMIL* were not designed to include graphic data as such. Instead, they include links that point at data files which are stored separately.

- ***DB Management Systems.*** An illustration's layer data can also be regarded as a collection of data items which are in relation to each other. Those data items and their interrelations can be managed by a *Database Management System (DBMS)* (e. g., [GSG+02, Hal01]). Employing *DBMSs*, the entire data can be efficiently processed and compactly stored on a device. Examples for *DBMSs* are *Microsoft Access*, *DB2*, *Oracle*, or *SQL Server*.

Hybrid Data Storage

Image data and textual descriptions are typical elements of an illustration. When those elements are included within the same data file, the storage technique is said to be an *Integrated Storage* technique. However, other data associated with that illustration can be media such as auditory or movie data which is included in separate files. Since such an illustration comprises elements of both *Integrated* and *Separated Storage*, its method for storing the layers is called *Hybrid Data Storage*.

2.2.3 Layer Storage Exemplified

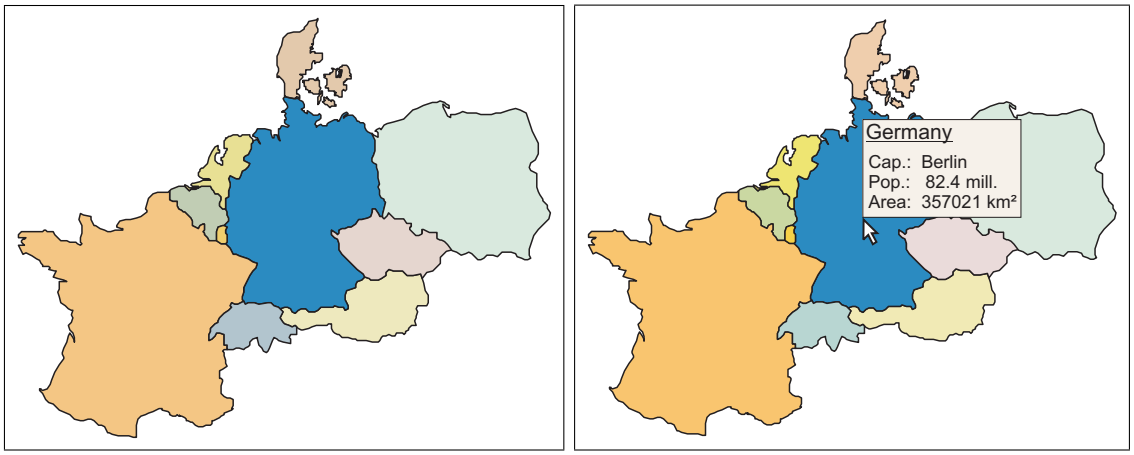


Figure 2.5: *Map of Germany and its neighboring countries. The image to the right hand side illustrates the display of an* Illustration Layer *component, triggered by the mouse cursor position.*

Illustration example

A common example for an explorable illustration is an image which is divided into regions. For each region, descriptive metadata can be requested which is invisible by nature. Such an example is depicted in Figure 2.5. With no interaction on user side, solely differently colored regions are displayed which represent Germany and its neighboring countries. However, once the mouse cursor reaches a region, the associated metadata is displayed.

In terms of layers, Figure 2.5(left) shows the *Background Layer* whereas a single *Illustration Layer* component is included in Figure 2.5(right), in addition to the *Background Layer*. The *Illustration Layer* consists of the regions' contour coordinates, the textual data to be displayed, interrelations between contours and texts, and interaction details.

Storage techniques

To illustrate a first technique, the example's *Illustration Layer* data was structured according to the *Relational Database* example dealt with in [RSV02] (see Fig. 2.6). The

technique hence serves as an example for a *DB Management System* which can be queried for metadata display (for more details, refer to [RSV02]).

Country				
Name	**Capital**	**Population**	**Boundary**	**Area**
Germany	Berlin	78.5	B1	357021
France	Paris	58	B2	543965
...	...	...	...	...

Boundary	
ID	**Contour**
B1	C1
B2	C2
B2	C3
...	...

Contour		
ID	**Order**	**Point**
C1	2	P1
C1	1	P2
C1	3	P3
C1	...	...
C2	1	P4
C2	...	...
...	...	...

Point		
ID	**x**	**y**
P1	452	1000
P2	365	875
P3	386	985
P4	296	825
P5	589	189
...	...	...

Figure 2.6: *Relational representation of descriptive metadata such as names of countries, their boundaries, and statistical data.*

Illustration Layer data can also be expressed according to *XML* specifications, as is illustrated in Figure 2.7. Further formatting as to syntax specifications defined by image data files such as *JPEG2000* then allows to include the metadata within the image representation.

A last example is illustrated in Figure 2.8. It shows the creation of the illustration by exploiting the authoring tool *Macromedia Flash*. Upon background image setup, sensitive regions (invisible buttons) need to be generated, their behaviors must be defined, and it is necessary to specify what should happen in certain events. Finally, the illustration can be stored in various formats, for example, the executable file that displays what can be seen in Figure 2.8(right).

2.2.4 Concluding Remarks

The techniques addressed so far were chosen to demonstrate the diversity of methods which are available to store *Background* and *Illustration Layer*. However, they are still a subset of all existing techniques.

A technique not addressed so far is digital watermarking. Associated with the concept of *Background Layer++*, it can be classified as *Integrated Storage* technique (refer to Fig. 2.4).

```
<Point ID='P1'>                    <Contour ID='C1'>
  <x>452</x> <y>1000</y>             <Points>P2 P1 P3 ...</Points>
</Point>                           </Contour>
<Point ID='P2'>                    <Contour ID='C2'>
  <x>365</x> <y>875</y>              <Points>P4 P5 ...</Points>
</Point>                           </Contour>
<Point ID='P3'>                    <Contour ID='C3'>
  <x>386</x> <y>985</y>              <Points>P5 ...</Points>
</Point>                           </Contour>
  ...                                ...

<Boundary ID='B1'>                 <Country>
  <Contours>C1</Contours>            <Name>Germany</Name>
</Boundary>                          <Capital>Berlin</Capital>
<Boundary ID='B2'>                   <Population>78.5</Population>
  <Contours>C2 C3</Contours>         <Boundary>B1</Boundary>
</Boundary>                          <Area>357021</Area>
<Boundary ID='B3'>                 </Country>
  <Contours>C4 C5</Contours>       <Country>
</Boundary>                          <Name>France</Name>
<Boundary ID='B4'>                   <Capital>Paris</Capital>
  ...                                ...
```

Figure 2.7: *The illustration's descriptive metadata expressed using XML syntax.*

Besides analyzing traditional watermarking techniques, the subsequent sections will propose novel watermarking techniques that serve the purpose of storing the two layers of which an illustration consists.

2.3 Digital Watermarking

Digital Watermarking, as it was defined by Cox et al. [CMB02], "is the practice of hiding a message about an image, audio clip, video clip, or other work of media within that work itself". As a consequence of message (*Watermark*) encoding, this work is imperceptibly altered.

The first time at which electronic watermarking was mentioned has been dated back to the year 1954 [CM02]. Since then, more than 50 years have past, and many fields of electronic (digital) watermarking are researched these days. However, interest in digital watermarking rapidly increased not until the 1990's. This basically resulted from copyright concerns which were induced by the spread of computer technology and the increasing popularity of the *World Wide Web*.

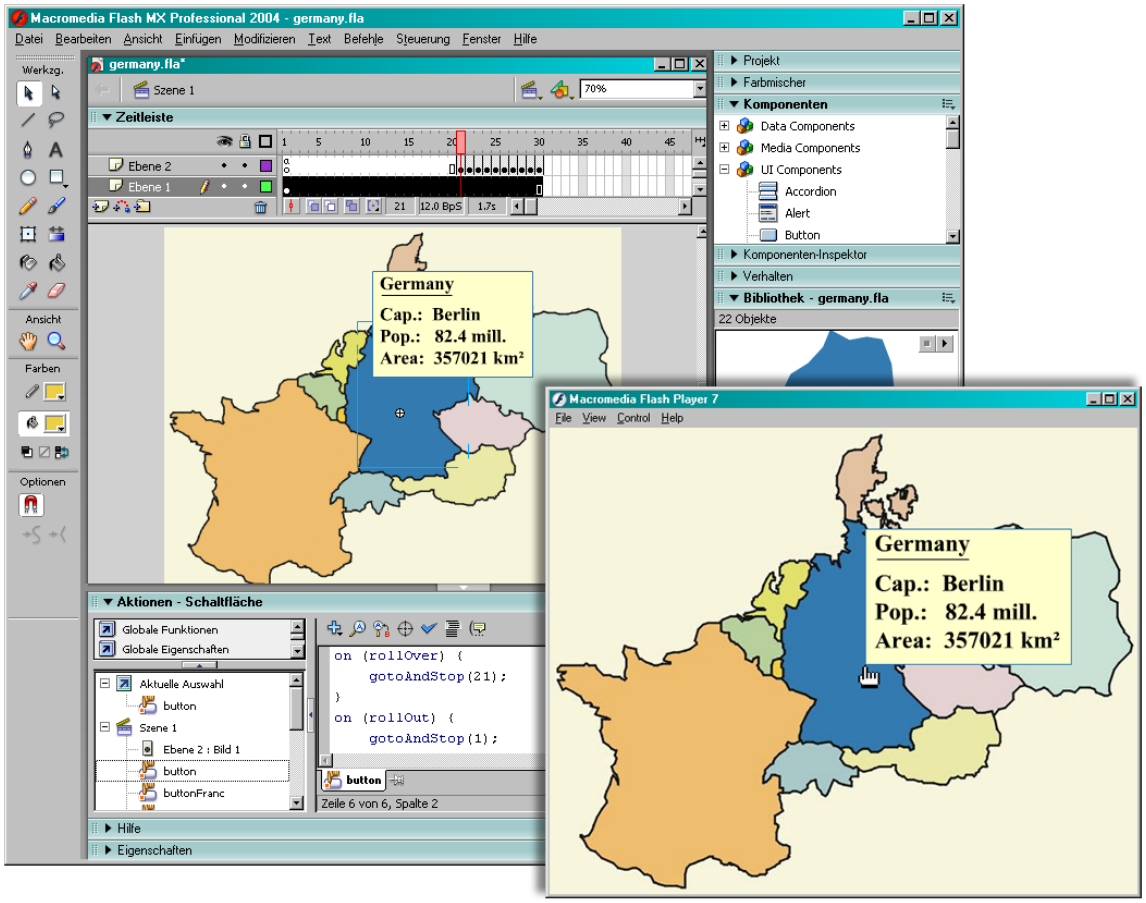


Figure 2.8: *The authoring tool Macromedia Flash provides a wide range of functionality to create explorable media.*

2.3.1 Digital Watermarking—A Subdiscipline of Information Hiding

Information Hiding is "a general term encompassing a wide range of problems beyond that of embedding messages in content" [CMB02]. According to Bender et al. [BGML96], it is a class of processes employed to embed data into various forms of media with a minimum amount of perceivable degradation added to the media.

The diagram in Figure 2.9 shows that the discipline of information hiding, in fact, encompasses many different subdisciplines, even though the diagram includes only a part of the original diagram presented by Petitcolas et al. [PAK99]. The diagram's key subdisciplines are *Covert Channels*, *Steganography*, and *Copyright Marking*.

Covert Channels

The term *Covert Channels* refers to "communication paths that were neither designed nor intended to transfer information at all" [PAK99]. Typically, there is no relationship between the channels and the messages sent through those channels. No person other than

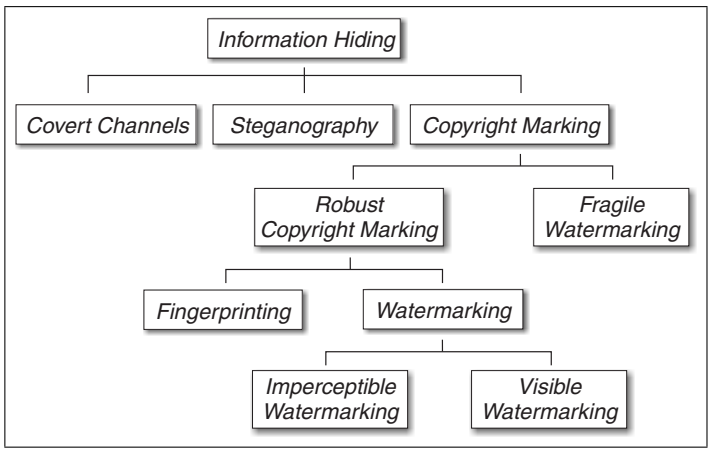


Figure 2.9: *Information hiding overview. Digital watermarking is one of many different subdisciplines.*

the sender and the recipient are aware that a covert message is being communicated.

Steganography

Steganography, derived from the Greek words *steganos* (covered) and *graphia* (writing), is "the art of concealed communication" [CMB02]. The existence of messages communicated is secret to persons other than sender and recipient. But in contrast to covert channels, steganographic messages are in relationship with the channels they are sent through. More detailed information about steganography can be found, for example, in [AP98, JJ98, MLC00].

Copyright Marking

The term *Copyright Marking* refers to techniques that add copyright messages, unique serial numbers (*Fingerprinting*), or other verification notices to media. Key requirements for copyright marking are that the embedded data is secure and withstands potential attacks. However, in contrast to covert channels and steganography, the existence of communicated messages is not necessarily concealed.

The diagram in Figure 2.10 categorizes subdisciplines of information hiding according to whether a message is communicated secretly or overtly and whether a message is in relation with its communication channel.

Apparently, covert channels and steganography are in no relation to the work at hand since those techniques primarily aim at secret message communication. The difference between the remaining techniques, that non-secretly embed data, consists in whether there is a relationship between the message to be communicated and the *Cover Medium* being the medium into which the message is encoded. However, since *Overt Embedded Communication* is defined as "known transmission of auxiliary, hidden information that is unrelated

to the signal in which it is embedded" [CMB02], digital watermarking is the subcategory that best conforms with what the work at hand aims at.

Nevertheless, even digital watermarking has a number of subcategories which are characterized by their properties specific to certain applications. The following paragraph itemizes a selection of those applications and properties.

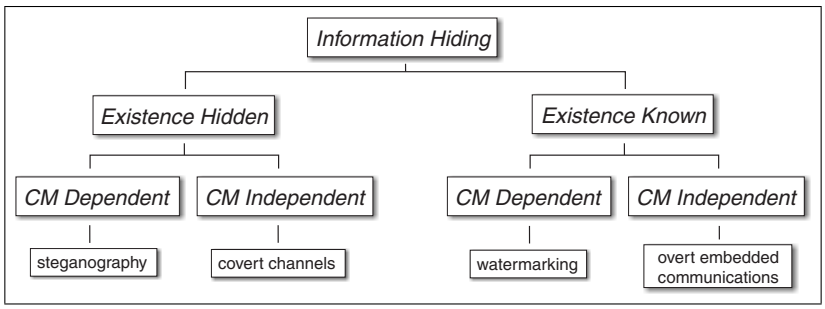


Figure 2.10: *Four categories of information hiding, according to* Cox *et al. [CMB02]. Information hiding strategies are differentiated by means of communication secrecy and the relationship between the* Cover Medium *(CM) and the communicated message.*

2.3.2 Applications and Properties of Digital Watermarking

Three main application fields of digital watermarking, identified by Kutter and Petitcolas [KP99], are: (1) *data monitoring*, (2) *copyright protection*, and (3) *data authentication*. Each of these application fields requires embedded watermark messages to withstand criminally intended attacks. An unauthorized person, for example, must neither be able to remove a watermark from its cover medium nor must the person be able to alter the watermark so that its recovery fails.

Beside those security applications, a fourth application field comprises *watermarks that convey object-specific information or captions* [MB99]. These watermarks add value to media from which a consumer benefits. Hence, security aspects can be rendered almost unnecessary.

Watermarking properties

Basically, watermarking properties concern the process of watermark embedding and detection. The most important properties with respect to this work are introduced in the following, characterized according to Cox et al. [CMB02].

- ***Fidelity*** refers to the perceptual similarity between the original medium and the medium with the watermark embedded. In this regard, *imperceivable* or *invisible* means that a consumer notices neither the presence of embedded data nor any modifications introduced to the medium because of watermarking. Those watermarking techniques are also referred to as *transparent* watermarking techniques.

- ***Data Payload*** or ***Capacity*** indicates the amount of data encoded as watermark within the cover medium. For a medium, the capacity can range from a few bits to megabytes, which depends on the application.
- ***Blind Detection*** means that the original, unwatermarked medium is not required for watermark recovery. Systems that implement blind detection are also referred to as *public watermarking systems*. ***Informed Detection***, on the other hand, is realized by *private watermarking systems*. Here, the original medium is essential for watermark recovery.
- ***Robustness*** refers to whether an embedded watermark can be detected and decoded after applying signal processing operations. Such operations can be, for example, spatial filtering, lossy compression, printing and scanning, and geometric distortions such as rotation, translation, cropping, or scaling.

  In contrast to robust watermarking, *fragile watermarking* (see Fig. 2.9) particularly aims at data authentication. Here, the damage or loss of an embedded watermark signalizes data corruption.

Another property relevant to this work is the ***Efficiency*** of watermark encoding and decoding. Also, for some applications, it may be needful to completely remove embedded watermarks and to restore the original medium (e. g., [FGD02]), which is referred to as ***Invertibility***.

The three basic requirements—high fidelity, robustness, and high capacity—demanded by many applications [CM02] compete with each other. When the energy of a watermark is increased to improve its robustness, the watermark's transparency decreases, and vice versa. Similarly, high capacity conflicts with high fidelity and robustness. Hence, in most cases, a tradeoff is searched that still complies with the application demands.

2.3.3 Digital Watermarking Techniques

The basic concept behind watermarking can be described using Costa's words [Cos83]: "[...] imagine a sheet of paper covered with independent dirt spots of normally distributed intensity. [...] The writer knows the location and intensity of the dirt spots, but the reader cannot distinguish them from the ink marks applied by the writer."

But what disables the reader to detect the dirt spots? The *Human Visual System (HVS)* is an imperfect signal detector [SKT98]. This means that only a fraction of the entirety of signals that arrive at the *HVS* are perceived by a person. In this regard, the effect of *Masking* plays a prominent role since it is a reason for why weaker signals are invisible to humans when other signals with certain characteristics are present (for further reading, refer to, e. g., [DVML02, RBH99]).

Digital watermarking from a distance

Wu and Liu [WL03] consider digital watermarking as a form of communication by which the watermark is the signal to be transmitted. According to Cox et al. [CMB02], the signal

is communicated through a transmission channel of which the cover medium is one part. Once the watermark message (signal) has been converted into a form which is compatible with the data structure of the cover medium, it can be inserted into that medium. The basic system components for watermark insertion and recovery are illustrated in Figure 2.11.

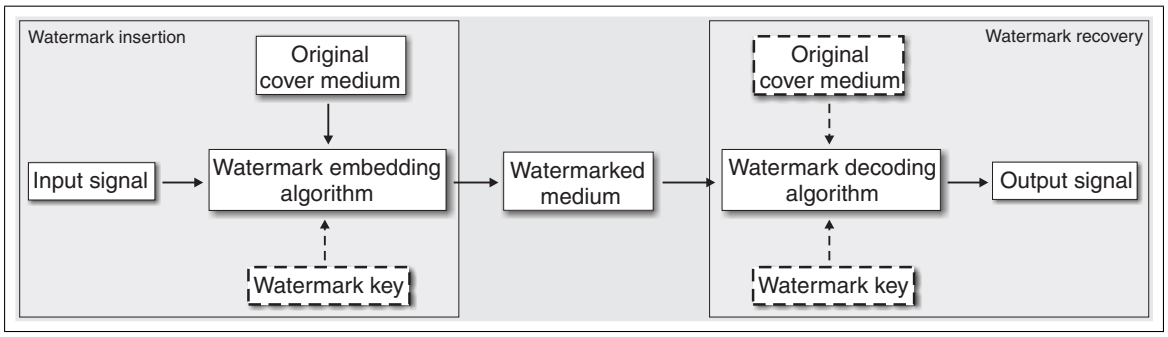


Figure 2.11: *General watermarking system. The input signal is inserted into the cover medium using a specific algorithm and, optionally, a watermark key. For watermark recovery, the watermarked medium as well as the two optional components—key and original medium—serve as input.*

Generally, a watermark message is inserted by slightly modifying the original cover medium, which is done with respect to the watermark itself. To retrieve the embedded message, the modifications introduced to the original cover medium need to be detected and appropriately interpreted, with or without involvement of the original medium or watermark keys.

Watermarking techniques can be classified in techniques that operate in two domains: the *Spatial Domain* and the *Transformed Domain*. In the *Spatial Domain*, the watermark message is embedded directly into the data representation of the cover medium. *Transformed Domain* techniques, on the other hand, transform the representation of the original cover medium prior to the watermark insertion. Such a transformation can, for example, result from applying a *Discrete Cosine Transform (DCT)* or a *Discrete Wavelet Transform (DWT)*. After watermark insertion, the medium is transformed back to its initial representation.

The subsequent paragraphs aim at surveying spatial and transformed domain techniques for 2D vector graphics, 3D model data, and 2D raster graphics.

Watermarking for 2D Vector Graphics

Vector Graphics are images which are represented by geometric primitives such as points, lines, or curves. Typically, they feature low storage costs, lossless geometric transformations, and high quality printouts.

These are exactly those features demanded by *Geographic Information Systems (GIS)*. *GIS* data facilitates representing real world objects such as roads, rivers, or land use by means of geographical coordinates, topology, and other characteristics. Huber [Hub02] describes a number of techniques for embedding messages in *GIS* data. One of these tech-

niques adds points to the graphic, which are positioned along the graphic's line segments. Message data is then represented by means of distances between subsequent points. Another approach for encoding data in *GIS* data was introduced by Voigt and Busch [VB02]. They modify geographical coordinates according to *Pseudo-Noise*-sequences that represent the watermark bits. A more recent approach by Voigt et al. [VYB04] suggests to embed the watermark by altering or leaving certain coefficients which result from applying the *DCT*.

Text documents can also be treated as vector graphics. Concerning document watermarking, Brassil et al. [BLMO95] proposed a technique which can be applied to both the rasterized representation of a document as well as the document's format file that includes content and layout. The authors suggest to embed watermark messages by vertical line shiftings, horizontal word shiftings, or by changing features such as the lengths of character strokes.

Solachidis and Pitas [SP04b] embed watermarks in lines, representing contours in an image, by slightly modifying their vertex coordinates. To enhance robustness, the watermark is encoded in the magnitude of the lines' *Fourier* descriptors. Another transformed domain approach was introduced by Ohbuchi et al. [OUE03]. They first compute a vertex mesh by treating all vertices as connected point set. The watermark is then encoded by altering low-frequency coefficients which result from applying a mesh spectral analysis to rectangular submeshes. This approach indicates that watermarking techniques, which were developed to be applied to 2D vector graphics, can be employed for 3D geometric data, as well.

Watermarking for 3D Geometric Models

3D Geometric Models are data representations of objects which are intended to be rendered as images. According to Foley et al. [FvDFH96], such models can contain a spatial layout, shape information (i. e., the geometry), other attributes that affect the appearance of the model components, information about connectivity (i. e., the topology), and application-specific data such as descriptive texts.

Ohbuchi et al. [OMA97] were the first to present their work in the particular field of watermarking 3D data. In [OMA98], they itemize various attributes which can be exploited for watermark insertion. These include, for example, the shape (polygonal mesh topology and geometry), shape attributes such as colors and texture coordinates, textures, and animation parameters. One of their proposed techniques first searches for sets of four adjacent triangles (so-called *MEPs*). The watermark is then embedded by slightly modifying the ratios between edges and heights for each triangle of a *MEP*.

Benedens [Ben99] proposed a technique that primarily aims at robustness to point randomization, mesh altering operations, and polygon simplification. To this end, similar surface patch normals are grouped into distinct sets (so-called *bins*). Thereafter, a single data bit can be encoded into one *bin* by moving the *bin's* mean normal within a certain range. Another approach that directly modifies the geometry of a model was introduced by Harte and Bors [HB02]. They compute non-intersecting bounding volumes and embed the watermark message by moving vertices with respect to their enclosing volumes. Mao

et al. [MSI01] proposed a technique that does not affect the appearance of the original mesh. To this end, they add new vertices to triangles, one vertex per edge. Watermark encoding is realized by adapting the ratio between the length of the original edge and the length of the edge segment which is defined by a triangle vertex and the added vertex. There are numbers of other approaches which can be classified in the spatial domain (e. g., [ADME02, ME04]).

Kanai et al. [KDK98] were the first to propose a technique which is based on decomposing the original mesh by applying a wavelet transform. They encode data by modifying wavelet coefficient vectors. To increase the robustness of embedded watermarks, Praun et al. [PHF99] suggested to convert the original mesh into a multiresolution representation which consists of a coarse base mesh and a sequence of refinement operations. For those refinement operations that affect the model most, basis functions are defined which serve for watermark insertion. An improved technique was introduced by Yin et al. [YPSZ01]. They employ a multiresolution analysis to generate *Burt-Adelson* mesh pyramids. The watermark message is embedded in the coarser levels of those pyramids. In doing so, a watermark is resistant to multiresolution operations. There are many other approaches that operate in the transformed domain (e. g., [OMT02, UCB04]).

A completely different approach was presented by Garcia and Dugelay [GD03]. They encode a watermark in the texture of a 3D model. With their technique, they aim at protecting images synthesized from the model. For watermark recovery, applied render operations are reversed.

Watermarking for 2D Raster Graphics

A *2D Raster Graphic* is represented as matrix of pixel values each of which defines a color value at a specific position within the matrix. Typically, watermarks are embedded by directly modifying pixel values (spatial domain techniques) or by modifying an image's representation after a certain transformation (e. g., wavelet transform) was applied (transformed domain techniques). Furthermore, techniques can be classified according to whether watermark encoding affects the whole image or only certain regions, and according to whether perceptual models were included.

There exist various techniques that directly modify the *Least Significant Bits (LSBs)* of an image's color channels. Tirkel et al. [TRvS$^+$93] were amongst the first to propose a technique for embedding a watermark into a gray tone image. An earlier approach by Kurak and McHugh [KM92] can also be regarded as a technique that directly modifies the *LSBs* in an image. They hide an image (the watermark) within the cover image by interchanging the cover image's least significant bits and the watermark image's most significant bits. Another approach that directly modifies the bits of the blue color channel was presented by Kutter et al. [KJB97]. Here, the position of the pixel to be changed depends on a secret key. The modification introduced to that pixel is—with respect to the pixel's luminance—either additive or subtractive, which depends on the bit to be embedded. Lie and Chang [LC99] proposed a technique that also embeds a watermark by modifying the *LSBs* of a pixel, but they include the *HVS*. The number of *LSBs* per pixel that are to be modified is determined by a mapping function which was designed according to human

visual sensitivity to contrast.

To enhance robustness, the watermark should resemble the image to be protected [RDB96]. The assumption is that damaging the watermark would also damage the image, perceivably. Hence, the location of the watermark should correlate with the most important features of the image. In this regard, Kutter et al. [KBE99] refer to watermarking techniques that consider perceptually significant features in images as *Second Generation Watermarking Schemes*. These are in contrast to those schemes that embed data within the whole image, irrespectively of certain features such as edges, corners, or textured areas. In this context, the phenomenon that image regions of non-regular and highly changing luminance are able to mask other signals—which is referred to as *Contrast*, *Noise*, and *Entropy Masking* [WBT97]—can be exploited. Entropy, as a measure of a signal's information content, was included in various watermarking techniques (e. g., [vDD02, KS04]). Van Droogenbroeck and Delvaux [vDD02], for example, subdivide an image into equally sized blocks, compute the entropy of each block, and embed the data according to the determined entropies.

Transformed domain techniques aim at focusing perceptually significant features in images, as well. The majority of them embeds a watermark after applying a *Discrete Cosine Transform* (e. g., [KZ95, CKLS97, HAPG00]) or a *Discrete Wavelet Transform* (e. g., [XBA98, BBP01, MU01]). As a side-effect, the watermark's robustness to image compression operations is also increased since those operations include such transformations (e. g., *JPEG* [Wal92] or *JPEG2000* compression [SCE01, TM02]).

With regard to the capacity of a watermark, Barni et al. [BBRP99] analyzed how many bits can be reliably hidden in an image using a transformed domain technique. Security aspects focused, they found that the capacity strongly depends on the image energy (i. e., the variance of the transformed domain coefficients) and the watermark strength. Gorodetski et al. [GPSS01] addressed both high capacity and robustness to common image distortions. They use a *Singular Value Decomposition (SVD)* transform to compute a *SVD* image layer in which the *singular values* correspond to the image's luminance. The watermark is then embedded by slightly modifying the *singular values*, which has no perceivable effect. With their technique, the authors can encode one bit in each RGB color channel of a block which results from image segmentation.

Watermarking for 2D Raster Graphics confined to regions

Basically, there are two reasons for why watermark encoding is confined to certain regions: (1) to increase robustness and fidelity (*Feature-Based Techniques*) and (2) to assign a certain amount of data which is in semantic relation to that particular region (*Content-Based Techniques*).

Concerning *Feature-Based Techniques*, Su et al. [SWK99] state that a watermark is more protected when it is embedded within the most important regions of an image referred to as the *Regions Of Interest (ROI)*. With their proposed technique, they can encode a small amount of data by modifying wavelet coefficients, which is done with respect to the afore selected *ROI*. Nikolaidis and Pitas [NP01] discuss a watermarking technique that embeds watermarks so that they withstand attacks such as compression, filtering, or cropping.

To this end, regions—identified according to salient spatial features—are approximated by ellipsoids whose bounding rectangles are then employed for watermark encoding. A technique that focuses data encoding in images with spacious regions homogeneously colored (e. g., cartoon images or maps) was introduced by Masry [Mas05]. The author embeds watermark messages into the boundaries that divide adjacent regions.

As regards *Content-Based Techniques*, Boulgouris et al. [BKM+02] presented a technique with which they aim at indexing images to be retrieved from databases. To this end, two types of watermarks are embedded, one that comprises segmentation data, the other contains indexing data. The concept introduced by Rytsar et al. [RVEP04] is focused on providing metadata (short text labels) for specific image regions. After interactive region selection, the regions can be augmented with the metadata.

2.4 The Concept of Illustration Watermarking

The general purpose of watermark encoding is to prevent a medium's intellectual content from being copied, distributed, or modified without permission. On the other hand, already a number of approaches exist that focus providing descriptive metadata that adds value to a medium, as the previous section revealed (e. g., [BKM+02, RVEP04]).

This section introduces the concept of *Illustration Watermarking* which is related to the idea behind Alattar's *Smart Images* [Ala00]. *Smart Images* are printouts of digital images which were augmented with data such as Web pointers or contact information using a watermarking technique. These printouts are intended for distribution. The encoded data can then be extracted by digitizing (scanning) a copy and applying the decoding method. However, in contrast to *Smart Images*, which can be associated with the class of hybrid techniques (see Sect. 2.1.3), *Illustration Watermarking* is intended to be applied to digital-only media.

2.4.1 Digital Watermarks as Carriers of Descriptive Metadata

In Section 2.2.2, several techniques were surveyed which can be used to store a computer graphic such that its descriptive metadata can be displayed on demand. To this end, *Background* and *Illustration Layer* must be separated and appropriately stored. The technique of encoding data in media by means of digital watermarks lends itself to store *Background* and *Illustration Layer* because of the following characteristics:

- Watermarks are capable of storing a certain amount of data.
- Watermarks are imperceivable to the naked eye when transparent techniques are employed.
- The watermarked medium already includes *Background* and *Illustration Layer*, which indicates compact data storage.
- Watermarks can be locally confined so that *Illustration Layer* components can be linked with certain regions of the cover medium.

Illustration Watermark definition

The term *Illustration Watermark* refers to a digital watermark with certain properties. Its purpose is to add value to a medium from which an end-user benefits. In this regard, *Illustration Watermarks* are intended to provide information that enables end-user interaction with the watermarked medium (see [SIDS03]). Both the information eventually displayed and the information that provides guidelines for interaction are represented as *Illustration Watermarks* and are defined by the author of the watermarked medium. The procedure of embedding *Illustration Watermarks* into a cover medium is referred to as *Illustration Watermarking*.

2.4.2 Properties of Illustration Watermarks

Illustration Watermarks can be considered as traditional digital watermarks with specific properties. On an abstract level, a traditional watermark is a portion of data which is, in some way, encoded within its cover medium. The same applies to *Illustration Watermarks*, except the demand that they must be in semantic relation with the content of the medium. The purpose of traditional watermarks, on the other hand, is basically to prevent the watermarked medium from being affected by criminal activities. Thus, those watermarks must withstand criminally intended attacks, an aspect which can be disregarded in case of *Illustration Watermarking*.

Section 2.3.2 addressed various properties associated with digital watermarking. In the following, those properties which are most relevant to the concept of *Illustration Watermarking* are itemized:

- ***High Fidelity.*** *Illustration Watermarks* are imperceivable and hence, encoded using transparent watermarking techniques.
- ***High Data Payload.*** The data amount to be embedded depends on the size of the *Illustration Layer*.

  High fidelity and embedding the *Illustration Layer* in its entirety take the highest priorities, if need be, at the cost of the watermark's robustness.
- ***Robustness.*** Robustness to transformations such as compression or geometric distortions is desired (see [VD05]). However, due to the challenge of balancing the opposed properties (high fidelity, high data payload, and robustness), robustness is only hard to achieve.
- ***Blind Detection.*** *Illustration Watermarks* are recovered without the need of the original unwatermarked medium.

Besides those properties, *Illustration Watermarks* are generally embedded into those parts of the cover medium with which they are associated. This approach saves to store information regarding the interrelation between cover medium parts and their corresponding watermarks. Furthermore, a part of the cover medium and its watermark are closely connected with each other. Thus, the watermark remains attached, even when the cover medium part is being separated from the original medium.

2.4.3 Classification of Illustration Watermarking

How can *Illustration Watermarking* be classified in the context of other information hiding techniques?

The diagram in Figure 2.9 is not applicable for classification since *Illustration Watermarking*—as a subdiscipline of digital watermarking—is in no relation with *Copyright Marking* as the diagram would imply.

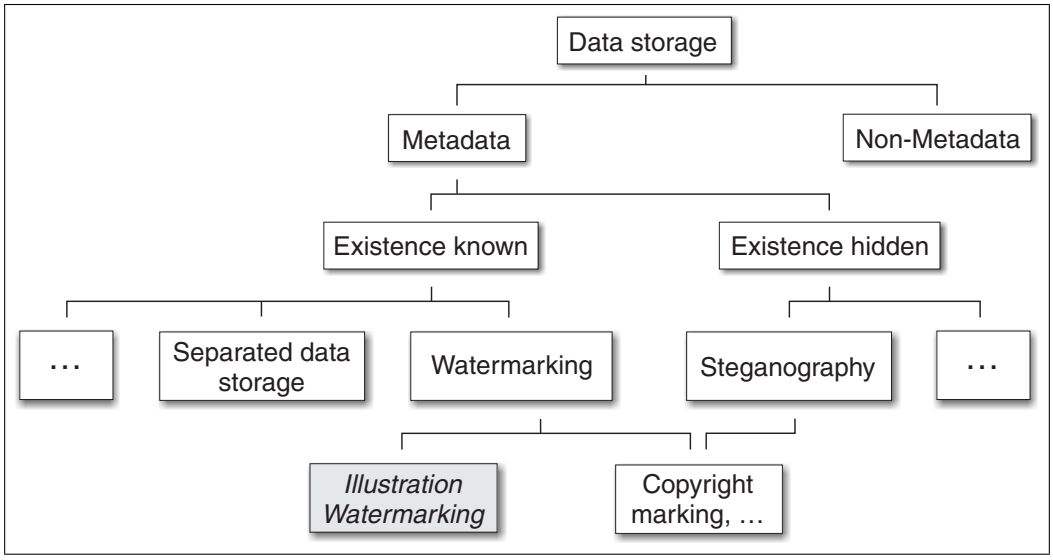


Figure 2.12: *Classification of* Illustration Watermarking *in the context of other data storage techniques.*

The diagram in Figure 2.10, on the other hand, characterizes *Watermarking* in a way that conforms better with the characteristics of *Illustration Watermarking*. It hence served as pattern to design the diagram in Figure 2.12 which shows *Illustration Watermarking* in the context of other data storage techniques.

2.5 Comparison of Techniques for Data Layer Storage

Various techniques were introduced which can be exploited to store the two layers of which an illustration consists. Those techniques are in part highly diversified, which however is in accord with the diversity of application fields that exist.

Illustrations, for example, can be augmented with short text labels or they can be accompanied by complex documents or auditory data. Other application-specific aspects concern the manner in which an end-user can interact with an illustration or who is to be addressed (e. g., Web community or single student).

This section aims at comparing the introduced techniques. In this regard, the proposed concept behind *Illustration Watermarking* will be related to the established techniques.

2.5.1 Integrated and Separated Data Storage

Needless to say that all concepts addressed can be exploited to generate illustrations with dynamic contents. However, *Authoring Tools* are by far more complex than the other concepts because they already include both a user interface for illustration design as well as methods for interpreting and displaying the data of an illustration. In this section, solely the data representation is considered. In case that aspects beyond data representation are addressed, it is assumed that appropriate software exists that, for example, can interprete and display the data of an illustration.

Authoring Tools

Authoring Tools can be used to generate highly complex illustrations. However, in most cases, the data file format they use is very specific and requires appropriate software.

+ Data file format allows for storing complex illustrations.
+ Various types of data can be associated with an illustration.
+ Compact data storage allows for close connection of the data layers.
− Tool-specific data file formats.
− The higher the complexity, the more specific software is necessary.
− Often commercial products high in price.

Background Layer++

Including *Illustration Layer* data in the *Background Layer* serves the purpose of closely linking the two layers. However, due to the demand for adapting the *Illustration Layer* data so that it conforms to the format specifications defined by the *Background Layer*, the *Illustration Layer* complexity may be limited.

+ Compact data storage allows for close connection of the data layers.
+ Established and widely-used data file formats can be employed.
+ *Background Layer* data can be visualized even if the *Illustration Layer* data cannot be extracted.
− Potential complexity limitations imposed by *Background Layer* data specifications.
− Lack of standardized definitions for *Illustration Layer* storage (file formats have their own specifications).
− Interpretation of included *Illustration Layer* data requires specific software.

Descriptive Representations

Descriptive Representations allow for straightforward editing. In principle, the respective format specifications are standardized. However, similar to the concept of *Background Layer++*, the underlying file structure defines the complexity of expressions. The direct inclusion of rasterized data is typically not supported.

+ Compact data storage allows for close connection of the data layers.
+ Established and widely-used data file formats can be employed.
+ Data file specifications are standardized. Some formats even provide concepts for specifying interaction rules.
+ Straightforward layer editing.
− *Illustration Layer* data is limited to data that can be expressed in terms of standardized text.
− Not designed for the inclusion of raster graphics or auditory data.
− Standardization often involves large data files with much overhead.

Individual Data Files

When illustration components are distributed across multiple data files, they can be shared by other illustrations. Thus, modifying a single component affects all illustrations that include this component, which can save work. On the other hand, a modified component may now have become invalid for a particular illustration. Also, managing several data files can be laborious and their propagation is typically user-unfriendly. Besides, coordinating separately stored components can be intricate and hard to realize. For example: *show the text (text file X) when the mouse cursor enters the image region (image file Y) characterized by a particular polygon (defined in file Z).*

+ Any digital medium can be associated with the illustration.
+ Illustration components can be shared by other illustrations.
− Managing several data files is more laborious than handling a single data file.
− Component coordination with the objective of defining interrelations and enabling interactivity is intricate.
− Illustration components may get lost.

DB Management Systems

Database Management Systems are efficient in extracting specific pieces of data from huge data amounts. Those systems are also excellently qualified for handling interrelations between data items. Potential problems concern the access to the database itself. For instance, the database server can be shut down or temporarily inaccessible due to other reasons. Rights management can also be an issue in this regard.

+ Any digital medium can be associated with the illustration.
+ Illustration components can be shared by other illustrations.
+ Efficient access to data and interrelations according to well-formulated queries.
− Problems as to accessibility can occur.
− Illustrations cannot be distributed as compact media.
− *DB Management Systems* require considerable administration effort.

The brief evaluations of the differing data storage concepts reveal that each concept has its individual characteristics some of which make them particularly applicable for certain tasks.

For example, illustrations augmented with many different types of media, interrelated and demanding high storage capacities, are potential applications for *Database Management Systems*. However, many illustrations are solely accompanied by short text labels and descriptions, which makes *Integrated Storage* techniques applicable. In case that highly complex interaction tasks are necessary to explore an illustration, *Authoring Tools* can be the best alternative. Nevertheless, as long as *Illustration Layer* components are compatible with *Background Layer* structures provided for metadata storage, concepts such as *Background Layer++* and *Descriptive Representations* have numerous advantages.

2.5.2 Illustration Watermarking—One Concept of Many

Illustration Watermarking follows the concept of *Background Layer++* and shares most of its characteristics. However, distinctions must be made with regard to aspects such as the robustness of assigned metadata or the types of metadata that can be included. In case of *Illustration Watermarking*, for example, embedded metadata can currently get lost when the watermarked medium is transformed in some way. On the other hand, *Illustration Watermarking* allows to encode any type of binary data, which is not necessarily facilitated by other *Background Layer++* concepts.

Table 2.1 shows *Illustration Watermarking* in comparison with the concepts discussed in the previous sections. The first part includes properties (accessibility, compactness, storage costs, and compatibility) which were addressed in Section 2.2.1 that dealt with basic requirements for data layer storage. The second part is particularly important with regard to *Illustration Watermarking*.

	Author. Tools	Backgr. Layer++	Descr. Repres.	Indiv. Files	DBMS	Illustr. Waterm.
Accessibility	++	++	++	+	+	++
Compactness	++	++	++	−−	−	++
Storage costs	++	+	−	○	+	+
Compatibility	−−	+	++	+	○	○
Fidelity	++	+	++	++	++	+
Capacity	++	○	++	++	++	−
Robustness	○	○	++	++	++	−−
Data variety	++	○	−	++	++	++
Data mapping	++	+	−	−−	−−	++
Splitting	−−	○	−	−	−	+
++ highly applicable ○ neutral −− not applicable						

Table 2.1: *Concepts for data layer storage in comparison with* Illustration Watermarking.

Discussion of the table's first part

Several properties were already addressed in the previous Section 2.5.1. In case of *Individual Data Files* and *DBMSs*, problems regarding ***accessibility*** can arise when single data files were displaced or access to the database is denied. The positive and negative ratings in Table 2.1 as to an illustration's ***compactness*** can be attributed to the manner in which *Illustration Layer* components are stored, which significantly differs between *Integrated* and *Separated Storage* techniques.

Regarding ***storage costs***, *Descriptive Representations* were negatively rated because (1) it is costly to represent, for example, pixel data in terms of text and (2) XML-based structures can be used to express many different issues, but it also takes many words. The latter aspect also applies to techniques with which event handling is specified and stored in words.

Compatibility can be characterized as follows: the more *Illustration Layer* components can be extracted without the need for specific interpretation software, the higher is the compatibility of a technique. Hence, techniques such as *SVG* (*Descriptive Representation*) which are based on standardized specifications for expressing diversified kinds of information are of high compatibility. On the other hand, *Authoring Tools* generally create very specific data structures which require very specific software for handling.

Discussion of the table's second part

Regarding the first property (***fidelity***), none of the concepts except for *Illustration Watermarking* affects the view of the illustration because of enriching the illustration with

descriptive metadata. In case of *Illustration Watermarking*, ***fidelity*** and ***capacity*** are strongly interrelated to each other. The more data is to be embedded, the higher is the risk of introducing perceivable modifications.

A major disadvantage of *Illustration Watermarking* is its lack of ***robustness*** to modifications introduced to a watermarked medium. Since most of the other concepts (except for *Authoring Tools*) clearly separate *Background* and *Illustration Layer*, modifying the *Background Layer* has no influence on metadata extraction.

Data variety refers to the types of information items which can be associated with an illustration. For *Descriptive Representations*, for example, metadata is limited to data which can be expressed by means of standard text. *Illustration Watermarking*, on the other hand, sets no limits because any data is embedded according to its binary representation.

It should be possible to link parts of the *Background Layer* with their corresponding *Illustration Layer* components (***data mapping***). *Authoring Tools* allow for assigning data to specific parts of the illustration. The same applies to *Illustration Watermarking* because metadata is directly embedded within those parts of the *Background Layer* with which it is associated. The remaining concepts allow for linking *Background* and *Illustration Layer* components, as well, but indirectly. Parts of the *Background Layer* must be explicitly defined and stored before they can be assigned with metadata.

In this regard, problems can emerge when parts of the *Background Layer* which are associated with metadata are to be copied to a different location (***splitting***) since region information must be copied, as well. Only specific software can handle both splitting of the illustration and preserving the descriptive metadata. Illustrations, on the other hand, which have their descriptive metadata embedded as *Illustration Watermarks* can be split—without the loss of metadata—as long as a watermarked region of the *Background Layer* is selected entirely.

2.6 Summary

To store illustrations with dynamic contents which can be controlled by end-users, the components which allow for such interaction must be identified. In this regard, this chapter started with dividing illustrative media in two layers: the *Background Layer* and the *Illustration Layer*. The *Illustration Layer* contains the whole data which is necessary for interaction. This includes the information to be displayed on demand as well as the manner in which this information can be provided to the end-user. After identifying those components, requirements for their appropriate data representation were itemized and discussed. Various existing techniques were introduced and analyzed in this regard. In addition, a new concept (*Illustration Watermarking*) was introduced which is related to digital watermarking and which is intended to coexist with other techniques. Finally, the presented techniques were compared and evaluated in terms of their assets and drawbacks.

3 Illustration Watermarks for Geometric Data

The work at hand proposes various *Illustration Watermarking* techniques for both geometric data as well as for rasterized data. This chapter addresses techniques which can be applied to geometric data. Geometric data is represented as a sequence of descriptive expressions. The manner in which the data can eventually be visualized allows geometric data to be subdivided into two classes. The first class comprises 2D vector graphics which are displayed as static images. 3D geometric models, on the other hand, which are associated with the second class are designed such that varying views are facilitated. This is enabled by adding the third dimension to the geometry representation as well as by including display variables such as camera parameters (e. g., varying points of view) or illumination models (e. g., varying numbers, shapes, and positions of light sources).

In this regard, Section 3.1 analyzes vector graphics and proposes techniques for encoding watermarks into those graphics. Thereafter, 3D geometric models are explored and potential watermarking techniques are introduced in Section 3.2. Finally, Section 3.3 contains concluding remarks.

3.1 Illustration Watermarks for Vector Graphics

Vector graphics, briefly introduced in Section 2.3.3, are typically displayed as static images. Their view is defined by the data representation. It includes expressions that specify how shapes (e. g., lines, polygons, or texts) are to be drawn in two-dimensional space.

```
<svg xmlns=''http://www.w3.org/2000/svg''
 xmlns:xlink=''http://www.w3.org/1999/xlink''>
 <g>
  <polygon points=''50,0 97,35 79,91 20,91 2,35''
   stroke=''black'' fill=''blue''/>
  <polygon points=''50,0 79,91 2,35 97,35 20,91''
   stroke=''black'' fill=''yellow''/>
 </g>
</svg>
```

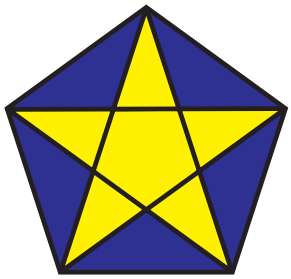


Figure 3.1: *Example for a SVG document taken from [Cag02]. Its interpretation yields the graphic on the right hand side.*

To enable discussing watermarking techniques, the basic structure of the data represen-

tation of a vector graphic must be analyzed. In so doing, those features of a vector graphic can be revealed which can be exploited for watermark insertion.

3.1.1 Basic Structure of Vector Graphic Representations

Vector graphics are represented descriptively which means all the elements of a graphic that are to be drawn are specified as a sequence of expressions. The basic structure is similar for all graphics, irrespective of the data file format. Generally, it starts with format specific notes. The subsequent part then includes expressions concerning the drawing, for example, colors, line styles, fill options, and geometries.

Figure 3.1 shows the data representation of a *SVG* document [Wor03] the interpretation of which is displayed in the figure's right hand side. The same result can be achieved by employing the stack-based *PostScript* language [Ado99]. The corresponding data representation is shown in Figure 3.2. Even though it seems to differ considerably from the *SVG* representation, its core information is the same.

```
%!PS-Adobe
 newpath
    50 91 moveto
    97 56 lineto
    79 0 lineto
    20 0 lineto
    2 56 lineto
 closepath
 gsave
    0 0 1 setrgbcolor fill
 grestore
    0 0 0 setrgbcolor stroke
    ...
```

```
    ...
 newpath
    50 91 moveto
    79 0 lineto
    2 56 lineto
    97 56 lineto
    20 0 lineto
 closepath
 gsave
    1 1 0 setrgbcolor fill
 grestore
    0 0 0 setrgbcolor stroke
showpage
```

Figure 3.2: *Example for a PostScript document the visualization of which equals the graphic shown in Fig. 3.1.*

To develop *Illustration Watermarking* techniques, characteristic data representation features must be identified which are shared by established vector graphic specifications. Embedding watermarks in those features prevents watermark data damage or loss, particularly with regard to potential file format conversions after a graphic has been watermarked.

The two examples for data representation depicted in Figures 3.1 and 3.2 reveal two characteristic features which can be considered for watermark insertion. These are color values, split in their RGB components, and point coordinates of line segments. Those features are essential for representing a vector graphic and thus, are part of any vector graphic specification.

The watermarking approaches which are described in the following section are based on modifying those data representation features.

3.1.2 Approaches for Illustration Watermarking

The proposed techniques addressed in this section were introduced in [SIDS03]. Basically, those techniques were applied to line-drawings—a class of vector graphics—which are a very common drawing technique for illustrations (e. g., [Hod03]). To enhance communicating their contents, those drawings are often augmented with text labels and captions. They are thus a major application for *Illustration Watermarking*.

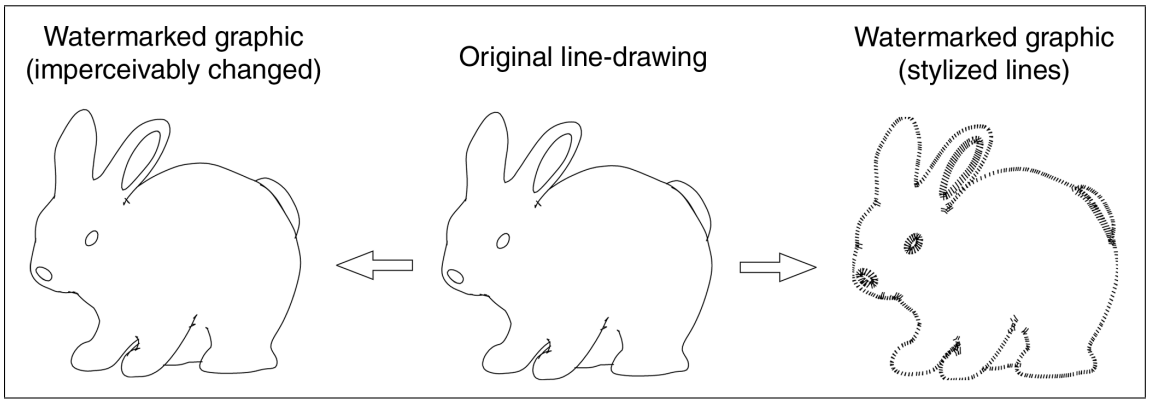


Figure 3.3: *Embedding watermarks into a silhouette that is computed from a 3D model according to the selected view. The techniques used for watermarking produce either graphics that are undistinguishable from their original pendants or graphics visibly modified (e. g., stylized lines).*

The line-drawings being focused consist of line segments which can be stored according to vector-oriented file format specifications such as the *PDF* file format.

Figure 3.3 shows a line-drawing and two watermarked versions of it. Those watermarked line-drawings result from applying two classes of algorithms: (1) algorithms that do not introduce perceivable changes to the cover graphic and (2) algorithms that introduce changes which appear as certain styles assigned to the line segments.

Imperceivable Illustration Watermarks

A selection of existing watermarking techniques for vector graphics was overviewed in Section 2.3.3. The two techniques which are described in this paragraph roughly follow the concept presented by Huber [Hub02]. They were developed with the objective of visually leaving the original graphic unchanged. To this end, new vertices are inserted which are positioned along the line segments of a graphic. The techniques differ in their manner of selecting the positions for the vertices to be inserted.

(I) Vertex insertion

The first technique subdivides line segments before a new vertex is inserted in each subsegment.

- ***Subdivision.*** The graphic's line segments are subdivided according to the number

of watermark bits to be embedded. The shorter the subsegments which result, the greater can be the amount of encodable data.

- ***Insertion.*** For each computed subsegment, a new vertex is inserted the position of which is chosen relative to the start point of the subsegment. The distance between start point and new vertex indicates which bit of the current byte of the watermark message is set.

Figure 3.4(b) illustrates the subdivision after which each line segment L_j ($j = \{1, ..., M\}$ with M = number of line segments) consists of N_{L_j} vertices. The magnification shows potential positions for the new vertex. Each of these positions, from which only one will be used, represents one bit of the current byte.

$\tilde{P}_{i,j}$ is defined as vertex to be inserted between $P_{i,j}$ and $P_{i+1,j}$ ($i = \{1, ..., N_{L_j}\}$). When $\tilde{P}_{i,j}$ equals $P_{i,j}$, bit 0 is set; when $\tilde{P}_{i,j}$ is positioned in the middle between $P_{i,j}$ and $P_{i+1,j}$, it sets bit 4. The following equation specifies how the position of $\tilde{P}_{i,j}$ can be computed. Note that *bit* ranges from 0 to 7 for bit representation. Since solely set bits are encoded, the start of the subsequent byte is signalized by setting *bit* to 8. An example for encoding the character $'a'$ is illustrated in Figure 3.4(c).

$$\tilde{P}_{i,j} = P_{i,j} + \frac{bit}{8} \cdot (P_{i+1,j} - P_{i,j}) \quad (0 \leq bit \leq 8) \tag{3.1}$$

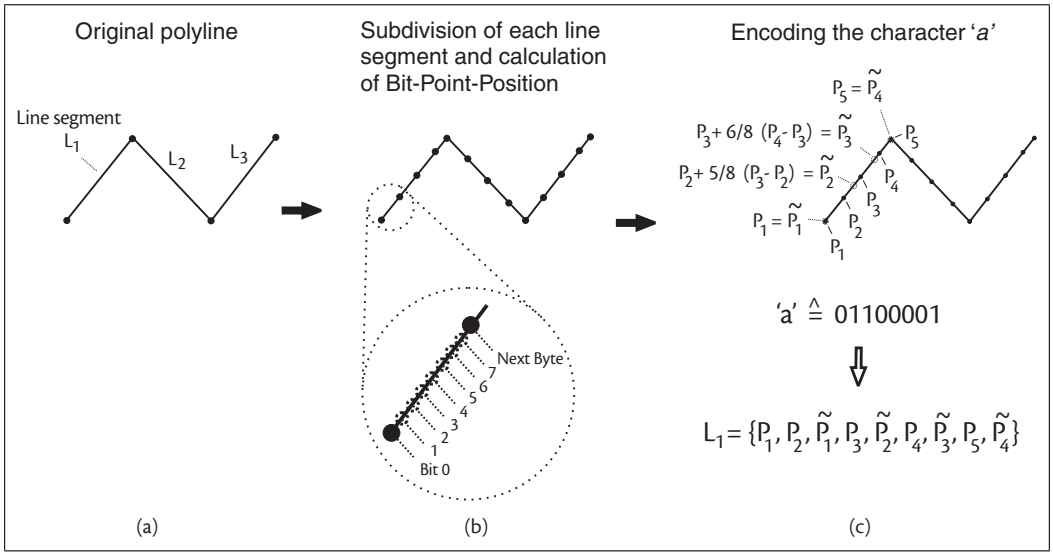


Figure 3.4: *The line segments of the original polyline (a) are subdivided, which results in line subsegments (b). In between a subsegment's end points a new vertex is inserted the position of which relative to the subsegment's start point defines a bit of the data stream to be encoded. As an example, the character 'a' is encoded (c).*

The proposed technique is a blind watermarking technique. Hence, neither the original graphic nor any encoding details are necessary for watermark recovery.

For decoding, the vertices of each line segment are traversed and analyzed. When a vertex is—according to its coordinates—located between consecutive points, but follows them in the sequence of a line segment's point representations, it encodes watermark information. Determining the distance to the corresponding start point can then be used to extract the encoded bit information. To this end, the coefficient c ($c = bit/8$) needs to be computed:

$$c = \frac{\tilde{P} - P_{i,j}}{P_{i+1,j} - P_{i,j}} \tag{3.2}$$

$$bit = floor(c \cdot 8 + 0.5) \quad \text{(floor: greatest integral value)} \tag{3.3}$$

Figure 3.5(b) shows an example for applying this technique to a graphic. The magnification demonstrates that the watermarked graphic cannot be visually distinguished from its original depicted in Figure 3.5(a).

(II) Line segment length

The second technique encodes data by varying the lengths of line segments. In contrast to the first technique, line subdivision does not serve for increasing the number of subsegments available for the data insertion within the second step, it rather serves for direct data insertion.

According to a certain base length l, which should be at least about a magnitude smaller than the length of the shortest line segment present, subdivision proceeds as follows:

1. The length l_{L_j} of the current line segment is determined, and its first vertex is added to F_P. F_P is the array that will contain all vertices that result from line subdivision. The current length of the subdivided line segment l_{cur} equals 0.

2. The new length $\tilde{l}$ is calculated according to the bit position to be encoded (only set bits are considered):

$$\tilde{l} = \frac{1}{8} \cdot (bit + 1) \cdot l \quad (0 \leq bit \leq 7) \tag{3.4}$$

$$\tilde{l} = l + \frac{1}{8} \cdot l \quad \text{(next byte)} \tag{3.5}$$

   If $\tilde{l} + l_{cur}$ is less or equal to l_{L_j}, a new vertex is added to F_P. Its position on the original line segment has distance $\tilde{l}$ from the last point of F_P. Thereafter, $\tilde{l}$ is added to l_{cur} and Step 2 is repeated.

   As soon as $l_{cur} + \tilde{l}$ is greater than the segment's total length l_{L_j}, no further data can be embedded into this segment.

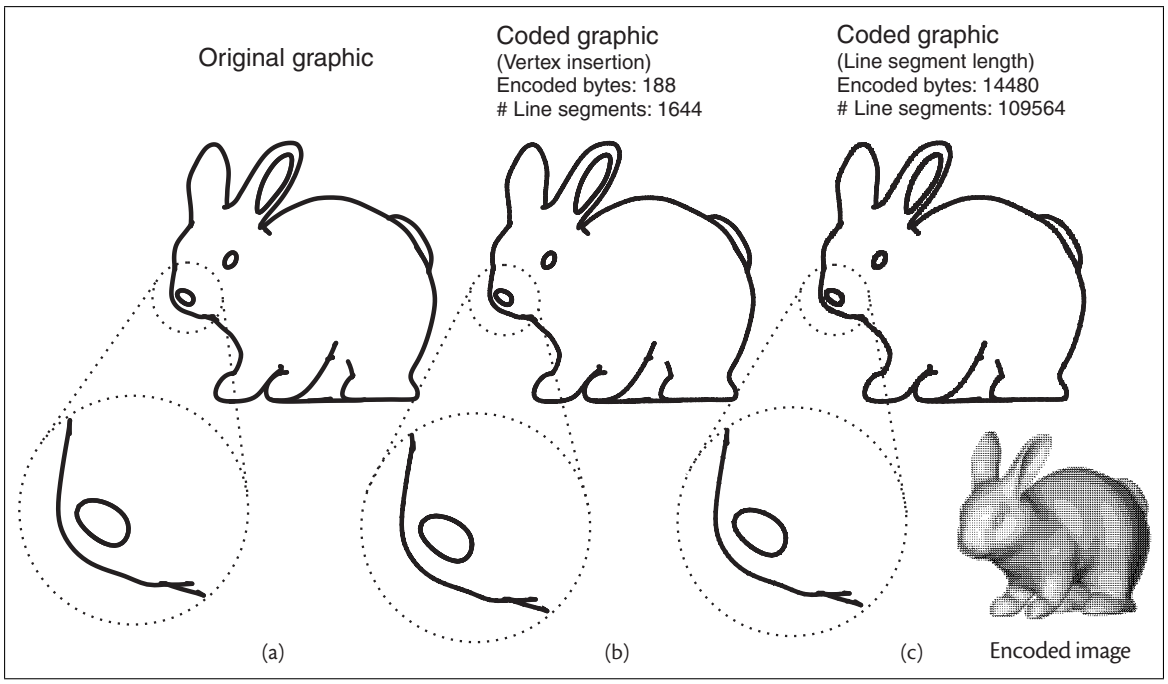


Figure 3.5: *Two examples for line graphics into which data was encoded. The graphic in (b) was coded by adding vertices in between line subsegments. The graphic in (c) was coded by adapting the subsegments' lengths. Here, the number of line subsegments was enough to encode the dithered image in the corner down right (255x247 pixels).*

3. Finally, the end point of the original line segment is added to F_P and Steps 1 to 3 are iterated for subsequent line segments.

The graphic in Figure 3.5(c) was watermarked using this technique. It contains, amongst other data, an image (1 bit depth) of size 255 x 247 pixels. The large number of line subsegments enables embedding such high data rates.

When decoding the embedded data, the base length l has to be known, which is a disadvantage. Therefore, the subdivision step was slightly modified. The length l is now encoded within the first line segment by setting its length to l. Extracting encoded bits can then proceed after the length $\tilde{l}$ of each subsequent line segment has been determined:

$$bit = floor\left(\frac{\tilde{l} \cdot 8}{l} + 0.5\right) - 1 \tag{3.6}$$

Perceivable Illustration Watermarks

The techniques which are described in this paragraph introduce perceivable changes to a graphic. Besides data insertion, applying those techniques can result in interesting drawing styles that give new meaning to a graphic.

(III) Angled lines

Replacing line segments by strokes with specific characteristics can both conceal information and assign certain attributes to a graphic.

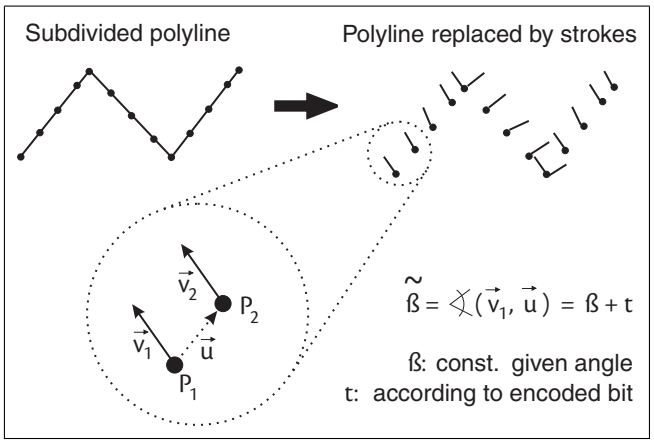


Figure 3.6: *After subdividing the original polyline, its line subsegments are replaced by strokes whose orientations vary according to the information they carry. By changing the attributes of these strokes, new characteristics can be assigned to the graphic.*

The proposed technique starts with subdividing the original line segments. But in contrast to the previous techniques that replaced a line segment by subsegments, the end points of subsegments now serve as start points of strokes differently aligned. To encode the watermark message, the angles $\tilde{\beta}$ between strokes and the original line segment are adapted, which is illustrated in Figure 3.6.

Given an angle β, the new angle $\tilde{\beta}$ is computed by adding a bit-specific amount to β. After determining $\tilde{\beta}$ and $\vec{u}$, which is the vector between P_i and P_{i+1}, the vector that defines the direction of the stroke (e. g., $\vec{v_1}$ in Figure 3.6) can be computed as follows:

$$\begin{pmatrix} v_{i,j_x} \\ v_{i,j_y} \end{pmatrix} = \begin{pmatrix} \cos\tilde{\beta} \cdot \vec{u_{j_x}} & - & \sin\tilde{\beta} \cdot \vec{u_{j_y}} \\ \sin\tilde{\beta} \cdot \vec{u_{j_x}} & + & \cos\tilde{\beta} \cdot \vec{u_{j_y}} \end{pmatrix} \tag{3.7}$$

Adjustable attributes which can be assigned to strokes and which thus affect the appearance of the whole graphic, are the length of a stroke and the space between adjacent strokes (see Fig. 3.7). Furthermore, the angle β can be assigned an arbitrary value.

The technique becomes a blind watermarking technique as soon as β, the angle which is essential for data decoding, is encoded as the orientation of the first stroke. The vectors $\vec{u_j}$ can be quickly computed by subtracting the start points of subsequent strokes. Calculating the angles between $\vec{u_j}$ and $v_{i,j}$ can then be used to extract the encoded data.

(IV) Line attribute variations

Watermarks can also be embedded by modifying line attributes such as color or line width (see Fig. 3.8). In gray-level graphics, for example, the gray tones can be adjusted with

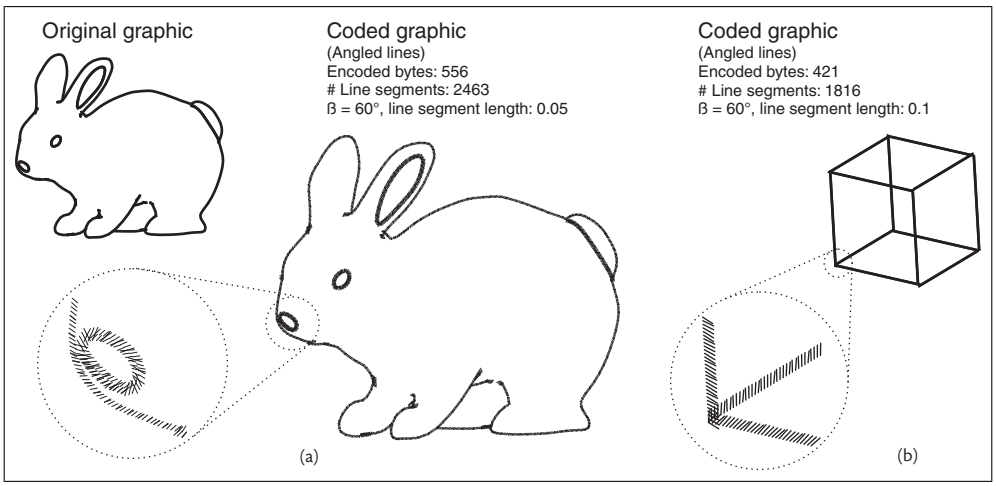


Figure 3.7: *Two examples for graphics that are composed of strokes instead of line segments. Besides carrying information, the strokes also influence the appearance of the graphic according to their attributes chosen. The viewer gets the impression that an artistic line style was assigned, which distracts from the real purpose: embedding watermarks.*

respect to the bits of the watermark message. As an example, a new gray tone $\tilde{G}$ (G is the base gray tone) can be computed as follows:

$$\tilde{G} = a \cdot 1/8 \cdot (bit + 1) + G - b \quad (0.0 \leq G \leq 1.0) \tag{3.8}$$

Parameter a affects the amount to which new values diverge from G and parameter b shifts the new gray tone such that it does not exceed its thresholds ($0 \leq \tilde{G} \leq 1$). These parameters should be chosen such that the difference between $\tilde{G}$ and G is minimal, but not zero. The graphic in Figure 3.8(a), for example, was generated using Equation 3.9. The start of a new byte can be indicated by setting $\tilde{G}$ equal to G.

$$\tilde{G} = 0.5 \cdot 1/8 \cdot (bit + 1) + G - 0.2 \quad (G = 0.5) \tag{3.9}$$

3.1.3 Evaluation of the Proposed Techniques

Even though the proposed techniques are only a fraction of what is feasible with regard to metadata insertion, already an evaluation of these few reveals significant differences.

The most important aspects analyzed concern the amount of data that can be embedded, the storage costs required, perceptibility, and whether the watermarked graphics are robust to transformations such as translation, rotation, or scaling.

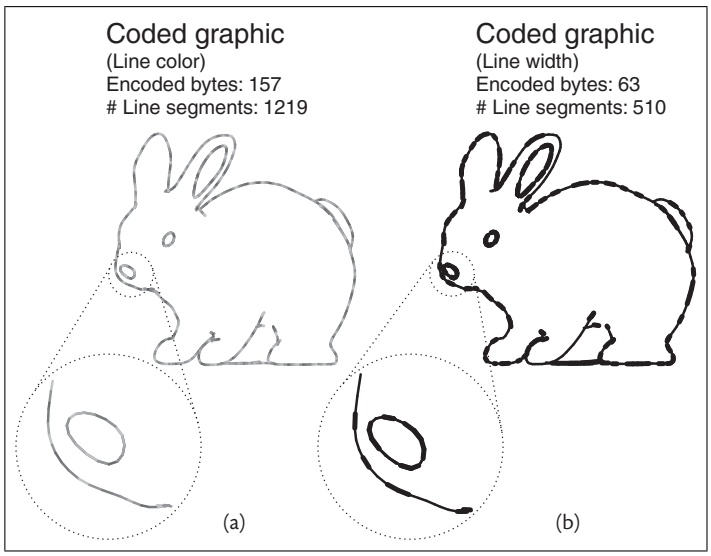


Figure 3.8: *Data is encoded by modifying line attributes. In (a), the color values of the line segments are distributed around the original color according to the bits to be encoded. In (b), the line widths of the line segments vary. Note that in both graphics the attribute variations are overstated for better illustration.*

Data payload and storage costs

The encodable data amount depends significantly on the number of line segments available. Line subdivision can thus enhance data payload. However, at the same time, it has to be noted that the insertion of new vertices during subdivision also raises the storage costs.

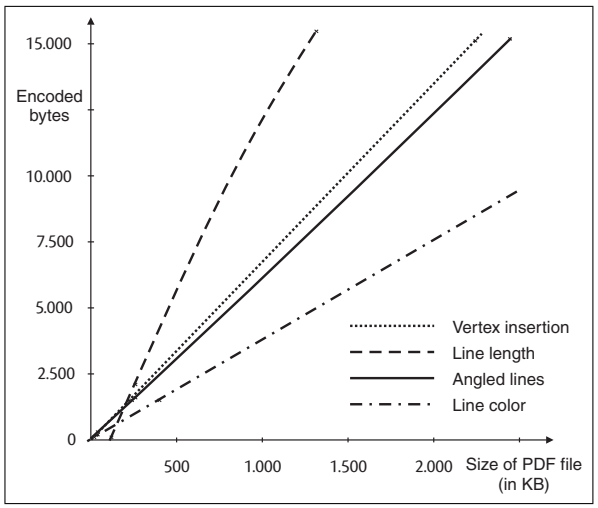


Figure 3.9: *Storage costs related to the encodable data amount.*

Table 3.1 presents measured values for each proposed technique. Besides the number of line segments before and after subdivision, other important values are the numbers of

encoded bytes and the resulting data file sizes.

It is evident that an increasing number of line subsegments yields an increase in both data payload and storage costs. The highest data amount can be embedded using the *Line length* technique. This is due to the fact that bit levels define subsegment lengths. In consequence, encoding low-level bits (less significant bits) involves generating quite short line subsegments.

The diagram in Figure 3.9 illustrates that the *Line length* technique is also the most efficient technique, whereas the *Line color* technique yields the largest PDF files. Furthermore, for each technique, a nearly linear correlation between the embedded amount of data and the resulting file size can be observed.

Regarding the *Line length* technique, a disparity can be observed at the origin of the diagram. In comparison with the other techniques, the *Line length* technique creates more subsegments, but the encodable data amount remains the same. This can be explained with the insertion of very short line segments which are shorter than the subsegment that would encode the lowest bit. This is necessary when the next bit to be encoded demands a subsegment length that would exceed the length of the original line segment.

Algorithm	***# Lines***	***# Lines (subdivided)***	***Encoded bytes***	***Size of coded PDF file***	***bytes / file size (%)***
Vertex insertion	71	1226	157	35.0 KB	0.44
	76	7135	1490	236.9 KB	0.61
	79	67358	15113	2.2 MB	0.66
	68	633878	143255	21.1 MB	0.65
Line length	71	7437	134	129.1 KB	0.10
	76	15537	2127	267.5 KB	0.78
	79	104010	22200	1.7 MB	1.25
	68	938895	211037	15.7 MB	1.28
Angled lines	71	1226	276	45.4 KB	0.59
	76	7135	1614	259.2 KB	0.61
	79	67358	15236	2.4 MB	0.61
	68	633878	143376	22.4 MB	0.61
Line color	71	1226	157	42.7 KB	0.36
	76	7135	1490	394.1 KB	0.37
	79	67358	15113	3.9 MB	0.37
	68	633878	143255	36.8 MB	0.37

Table 3.1: *Comparison of the proposed techniques.*

Perceptibility

The proposed techniques were classified in whether or not they may visually affect the view of the graphic to which they were applied. However, the only technique that definitively changes the appearance of a graphic is the *Angled lines* technique. The technique that aims

at altering the attributes of line segments (e. g., color and line width) does not necessarily cause perceivable changes when it is parameterized accordingly.

Even though introduced changes are imperceivable when visualizing a graphic, they can easily be detected by analyzing the data representation of the graphic.

Robustness

To evaluate the robustness of the introduced techniques, several geometric transformations were applied. The basic procedure in this regard started with importing a graphic, represented as *PDF* file, into a software tool such as *CorelDRAW*. The graphic was then transformed in terms of translation, scaling, and rotation, before it was exported according to *EPS* and *PS* file specifications. Finally, the result was converted back into the *PDF* file format.

The data which was encoded using the *Angled lines* and *Line color* algorithms could be decoded after applying the aforementioned transformations.

The two remaining techniques had problems during decoding. The first problem was that the latter of two identical consecutive vertices was removed by *CorelDRAW*'s export function. This problem could be solved by multiplying the vertex with a factor, for example 0.9999, so that a nearly exact copy of the preceding vertex results. Numerical roundoff errors caused by internal system conversions in *CorelDRAW* were a second problem. Including small error tolerances during decoding could solve this problem, as well.

3.2 Illustration Watermarks for 3D Geometric Data

3D geometric data (3D models), introduced in Section 2.3.3, can be regarded as extension to 2D vector graphics dealt with in the previous section. Such data also consists of mathematical expressions and other descriptive elements that define how the data is to be visualized. But in addition, 3D models typically include data such as camera parameters and coordinates in three dimensions which allow for interacting in 3D with what results from interpreting the information provided. Thus, the *Background Layer* no longer appears as a static image, but as an image that permanently changes its view with respect to certain instructions triggered, for example, by end-user input. Adjusting the camera, for instance, enables an end-user to view the model data from different directions or to move through a virtual scene composed of individual 3D models each of which can consist of various 3D objects (e. g., the components of an engine).

However, displaying model components by interpreting geometric and material descriptions is one aspect. Another aspect concerns the display of information that cannot be extracted from such a visualization alone. Textual descriptions for particular 3D objects, auditory explanations, or images that indicate how the virtual data may look like in reality are only a few examples for descriptive metadata that can enrich 3D model data (see Fig. 3.10).

This section addresses techniques which can serve the purpose of integrating descriptive metadata into the data representation of a 3D model.

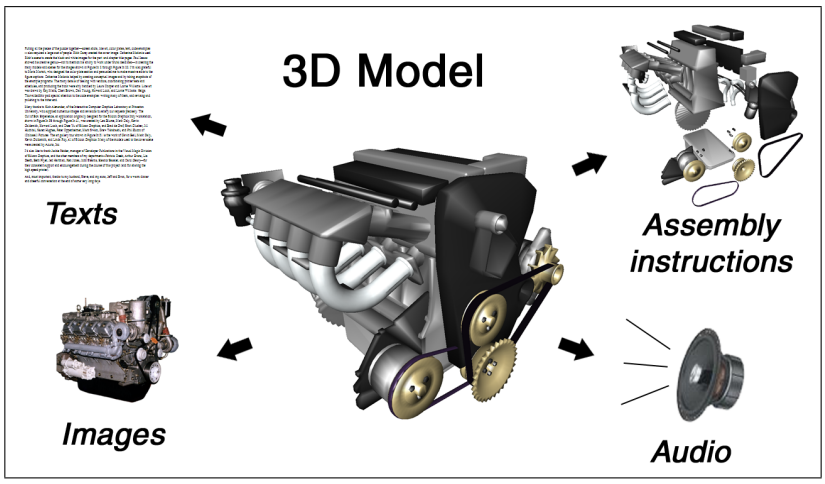


Figure 3.10: *3D model data can be augmented with descriptive metadata such as texts, images, or auditory data. Such metadata can reveal information beyond the information provided by the visualization of the model.*

3.2.1 Basic Structure of 3D Models

Many 3D file format specifications exist that allow for including data whose interpretation yields, in the final stage, rasterized images. The interpretation is done by so-called *3D render engines*. Examples for established file formats are *3ds* (*3D Studio MAX*), *iv* (*Open Inventor*), *obj* (*Alias Wavefront*), or *vrml* (*Virtual Reality Modeling Language*).

The basic data components which are necessary to render (rasterize) a 3D model are the same for all file formats. These comprise information about the model's geometry and material as well as information regarding light sources and viewing in terms of camera parameters. Since those data components are shared by the majority of file formats, they should be addressed by techniques for watermark insertion.

In the following, the *Open Inventor* file format serves as example for illustrating potential watermarking techniques.

3D toolkit *Open Inventor*

Open Inventor is a platform independent 3D toolkit designed by *Silicon Graphics, Inc.* that facilitates the development of interactive applications for 3D model data [Wer94].

In *Open Inventor*, 3D scenes are represented as directed, acyclic graphs which can be stored as data files. The structure of those scene graphs—the elements of which are called *nodes*—is hierarchic, that is, specific *group nodes* can have one or more other nodes subordinated. Scene graphs are traversed top-down and left-to-right, starting from the *root node*. Figure 3.11(left) shows an example for a scene graph the interpretation of which can yield an image such as the one depicted in Figure 3.11(right).

Open Inventor enables the inclusion of metadata by inserting a specific node (*SoInfo*) into the scene graph. In Figure 3.11, *Metadata* is such a node. However, including metadata that way has disadvantages:

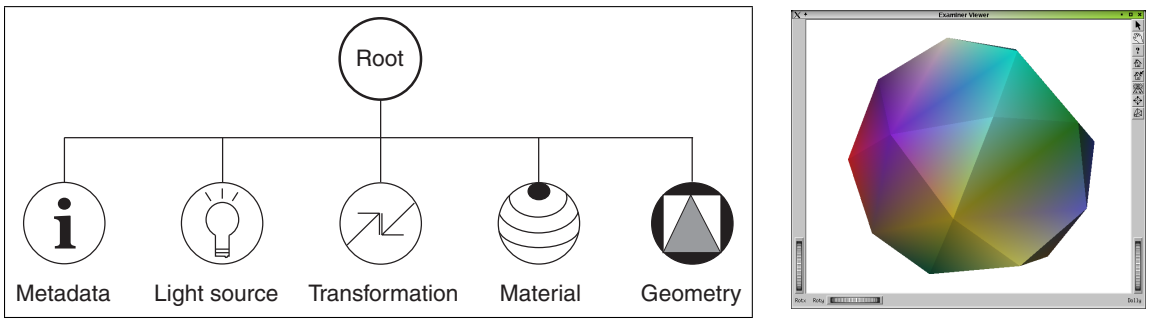


Figure 3.11: Open Inventor *scene graph (left) that includes basic elements which are required for rendering (right). In addition, it has an element that can store metadata as ASCII text.*

1. Only textual information can be specified using the *SoInfo* node. Other data such as images and auditory data can solely be referenced by means of links.

2. The concept behind nodes that can store metadata is very specific to *Open Inventor*. File format conversions may thus yield metadata loss.

Nevertheless, the scene graph in Figure 3.11 has other components which are more qualified for data insertion. These are material properties (e. g., colors, textures, shininess, etc.) and geometric data (e. g., coordinates of vertices, textures, and normals). The following sections propose watermarking techniques that make use of those model components.

3.2.2 Textures with Information Embedded

Each model component can be assigned at least one image as texture that affects the appearance of the rendered model. In case that a rendition shall appear with no texture effects, a texture can be set transparent. Hence, the texture becomes part of the model, but no part of the visualization.

In this context, a watermark message can be stored in two ways:

1. ***Watermarked Texture.*** Typically, a texture is stored as 2D raster graphic. Watermarks can thus be embedded by applying a watermarking technique to the raster graphic. Examples were given in Section 2.3.3 and will be given in Chapter 4.

2. ***Data Texture.*** A watermark message can also be represented directly as the colors of a texture. Since a *Data Texture* is generally set transparent, it can consist of arbitrarily colored pixels. Due to potential data rates of 24 bits per pixel, *Data Textures* are a very efficient approach for storing descriptive metadata.

In terms of *Open Inventor's* scene graph, texture images can be included straightforwardly at any position in the graph. To assign a texture to a particular model component, the texture image must be positioned before the component's geometric specifications.

Figure 3.12 shows two textures. The first texture (a) (*Watermarked Texture*) was assigned to a component of the model and hence, influences its appearance. The second

texture (b) (*Data Texture*) is a transparent texture, which means its alpha channel was set accordingly. This texture represents a sound file (*mp3* file of size 553 KB). To this end, the sound file's binary data was converted into the RGB color values of which the texture consists.

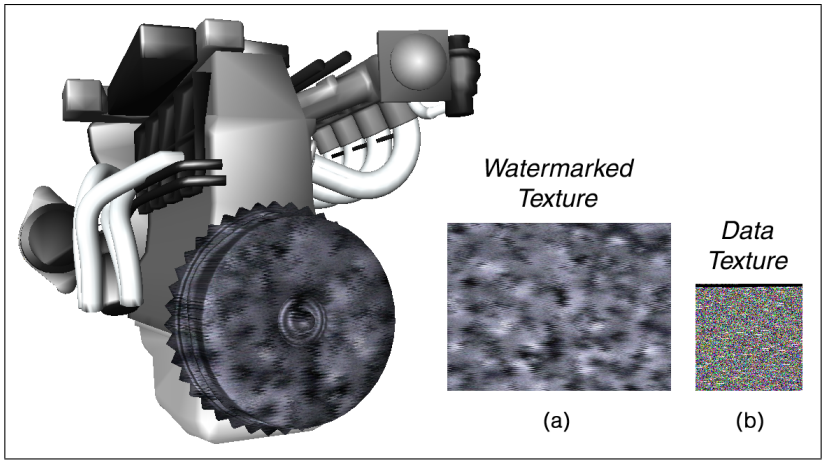


Figure 3.12: *Embedding* Illustration Watermarks *in a model's textures: (a) texture that represents the material of a model component; it was watermarked by employing a watermarking technique for 2D raster graphics; (b) texture, generated according to specific metadata, does not affect the appearance of the rendered model.*

Drawback: separation of model and metadata

Many 3D file formats support the inclusion of one or more (transparent) texture images. Thus, embedding data in textures is not only a powerful technique to augment a 3D model with descriptive metadata, it is also compatible with various data format specifications.

However, in most cases, texture images are stored separately from the model description. Interconnections between model components and textures are realized by links. In terms of the classification introduced in Section 2.2.2, the proposed technique is a technique for *Separated Data Storage* (*Individual Data Files*) which is associated with several drawbacks itemized in the section.

3.2.3 Data Storage via Carrier Objects

The approach addressed in this section results from collaboration with Silvio Lange [SL05]. The proposed technique embeds data by watermarking specific structures which are then smoothly integrated with the cover model. Thus, model and descriptive metadata become an entity instead of separated data items.

The basic idea behind this approach is to insert *Carrier Objects* (3D objects that carry the data to be embedded) into the 3D model that consists of polygonal surfaces. The two main steps in this regard are: (1) *Carrier Objects* are generated with respect to the

data to be encoded and (2) the *Carrier Objects* are placed inside the original model using cross-sections. Figure 3.13 outlines the basic steps.

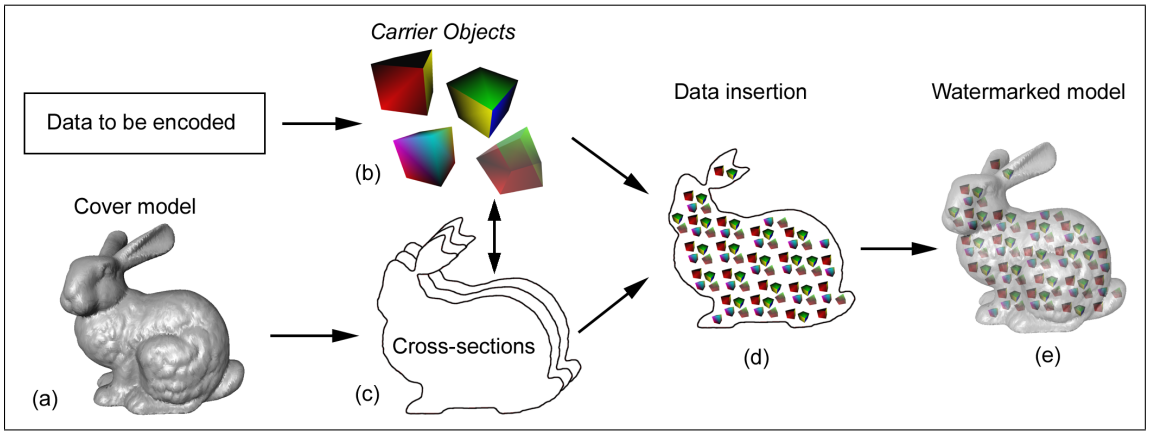


Figure 3.13: *Starting from a model or a component of it and a data stream to be encoded (a),* Carrier Objects *are generated and watermarked (b). At the same time, cross-sections are computed (c) which are used for placing the* Carrier Objects *inside the model (d). The (transparently) rendered 3D model that contains the (strongly magnified)* Carrier Objects *is shown in (e).*

Carrier Objects as Illustration Watermarks

Carrier Objects are 3D polygonal objects that will contain the descriptive metadata specified. To this end, their various geometric and topological properties can be modified.

Each *Carrier Object* can hold data whose amount depends on the complexity of the *Carrier Object* (e. g., number of its polygons). In Figure 3.14, four examples for *Carrier Objects* are illustrated. The *Carrier Object* in Figure 3.14(a) is the *Primary Carrier Object* (*PCO*). All *Consecutive Carrier Objects* (*CCO*) (Figure 3.14(b)-(d)) are positioned, transformed, and assigned attributes with respect to the *PCO*.

Characteristics of the Primary Carrier Object

The *Primary Carrier Object* does not store any portion of the watermark data stream, but it is of high importance with regard to the *Consecutive Carrier Objects*. Its geometry and topology serve as patterns when *Consecutive Carrier Objects* are encoded and decoded.

To facilitate data recovery, the primary object features specific characteristics. It has, for example, a single triangle T_P that does not share any of its edges with another triangle (see Fig 3.14(a)). In addition, the ambient colors of the triangle's vertices are coded: their least significant bits of the blue color channels, which are unset by default, are set. Furthermore, T_P has exactly one edge E_P whose end points V_P^* and $V_P^{\sim}$ are specifically coded: the second bits of their ambient colors' blue channels are set. V_P^* is the origin of the local coordinate system. To ease its detection during the decoding, it is coded, as well.

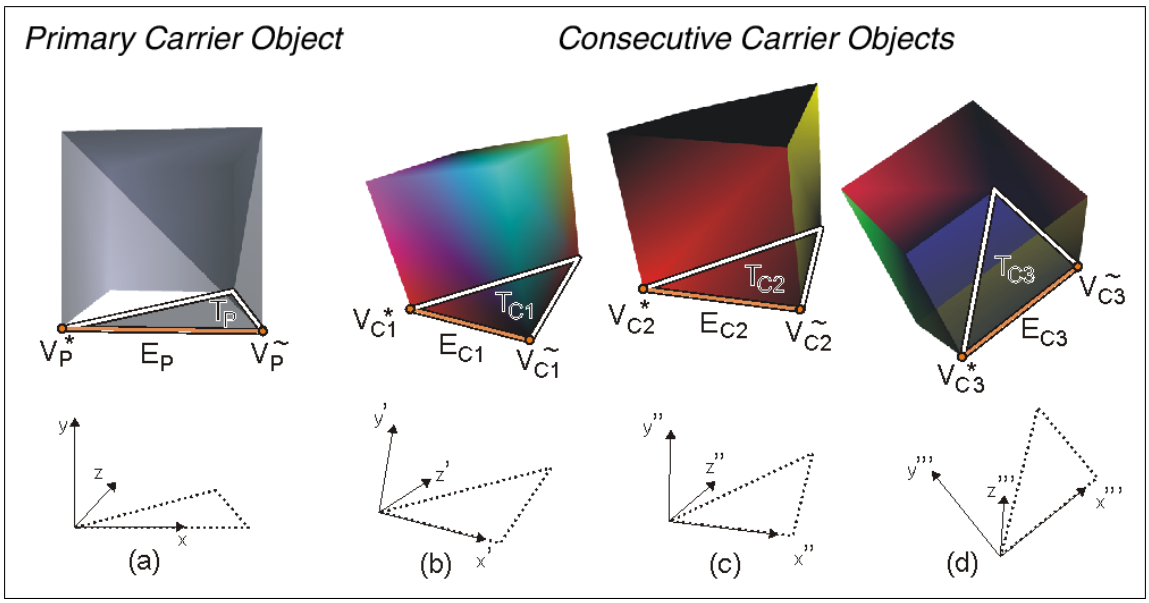


Figure 3.14: Primary *(a) and* Consecutive Carrier Objects *(b)-(d). The* Consecutive Carrier Objects *were watermarked by modifying certain material properties and by applying various transformations (e. g., scaling, rotation, vertex displacements). The objects' corresponding local coordinate systems are shown in the bottom row.*

The Consecutive Carrier Objects

Ohbuchi et al. [OMA98] itemized several embedding targets for 3D polygonal models, e. g., shape, shape attributes, or animation parameters. Each of these targets can be used for watermarking a single 3D object. For the approach described in this section, a subset of those targets is exploited for watermarking a single 3D object—a *Carrier Object*—as well. However, the difference is that the 3D model to be watermarked will, likely, contain numbers of *Carrier Objects*.

Comparable to what Ohbuchi et al. suggested, a *Carrier object's* embedding targets are: its transformation, geometry, topology, and material. In this regard, it is differentiated between attributes which are applied to the entire *Carrier Object*, its triangles, or its vertices. Hence, some attributes depend on the number of vertices of which an object consists (the object's complexity).

Embedding target: (1) Transformation

The transformation of a *Carrier Object* affects the entire object. Transformations include rotation, translation, and scaling (see Fig. 3.14(b)-(d)).

- *Rotation*: An angle is typically measured in radian (0-2π). Any floating-point number less than 5.24288 and in the range of 0-2π (with 5 decimal places at most) can be expressed using 19 bits. Those 19 bits can be used for data embedding (difference between primary object's orientation and current object's orientation). Carried to the 3 axes, **57 bits** can be stored by an object's rotation.

- *Translation*: Using 5 places, a non-negative number ($uvwxy$) less than 65536 can be expressed using 16 bits. When the position of the *Carrier Object i* (pos_{Ci}) is given, its new position along axis x can, for example, be computed by multiplying pos_{Ci_x} with $1.0uvwxy$. Hence, for each axis, a total number of **48 bits** can be coded.

- *Scaling*: *Consecutive Carrier Objects* can be scaled down with regard to the primary object. Hence, values between 0 and 1 using 5 decimal places can be specified, which allows a total of **48 bits** to be coded.

Embedding target: (2) Geometry

Operations such as modifying vertex positions are geometric transformations. The amount of data which can be inserted per object depends on the number of vertices being available. Similar to object translation, a total number of **48 bits** can be embedded per vertex.

In Figure 3.14, for each *Carrier Object* a triangle is marked whose vertices may not be altered. The reason for this limitation is that these triangles will be compared to the primary object's triangle during data recovery. Hence, for watermarking, a number of 3 must be subtracted from the total number of *Carrier Object* vertices.

Embedding target: (3) Topology

Depending on whether the vertices of a triangle are sorted clockwise or counter-clockwise, **1 bit** of data can be embedded per triangle. Generally, the vertex order affects the orientation of a triangle's normal and hence, its shading. However, since *Carrier Objects* are not intended to influence the appearance of the rendered model, this aspect can be disregarded.

Embedding target: (4) Material

Specific material properties can be applied to every single vertex. These include ambient, diffuse, specular, and emissive colors as well as the shininess of an object. Each of the color properties can hold 32 bits ($RGBA$), the shininess property is specified using 8 bits. Thus, a total number of 136 bits can be coded per vertex. However, since the three least significant bits of the ambient colors' blue channel are reserved for triangle, edge, and origin identification, the total number of codable bits per vertex is **133 bits**. Note that in case of a transparent model the materials' alpha channels should be set to 0 (translucent *Carrier Objects*), which decreases the encodable data amount accordingly.

Overview: data payload

According to the numbers of bits which can be coded per attribute (see Table 3.2), the total number of bits (n_{bits}) which can be inserted into a *Carrier Object* can be determined using Equation 3.10. It depends on the *Carrier Object's* number of vertices (n_{vert}) and triangles (n_{tria}).

$$n_{bits} = \underbrace{153}_{(1)} + \underbrace{(n_{vert} - 3) * 48}_{(2)} + \underbrace{n_{tria}}_{(3)} + \underbrace{n_{vert} * 133}_{(4)} \tag{3.10}$$

The number of *Carrier Objects* (n_{objs}) which are required for encoding a specific amount of data (n_{data}, in bits) can be determined using Equation 3.11. To encode, for example, 1000 bytes (8000 bits) by exploiting a simple cube as *Carrier Object* (8 vertices, 12 triangles), the equation yields a number of 7 *Carrier Objects*.

$$n_{objs} = \underbrace{ceil\left(\frac{n_{data}}{n_{bits}}\right)}_{\text{CCO}} + \underbrace{1}_{\text{PCO}} \tag{3.11}$$

Attribute	Applicable to	Capacity
Transformation (1)	whole object	153 bits per object
Rotation		57 bits per object
Translation		48 bits per object
Scaling		48 bits per object
Geometry (2)	vertices	48 bits per vertex
Topology (3)	triangles	1 bit per triangle
Material (4)	vertices	133 bits per vertex
colors		125 bits per vertex
shininess		8 bits per vertex

Table 3.2: *Amount of data which can be encoded within a* Carrier Object.

Placing Carrier Objects inside the model

Carrier Objects are geometric objects composed of triangles which are to be merged with the original 3D model so that model and *Carrier Objects* form an entity. In this regard, heed must be taken of placing the *Carrier Objects* because they may not intersect with the surface of the model. It is assumed that the model is a hole-less surface model that consists of triangles.

Carrier Object positioning basically comprises two steps: (1) the model is successively intersected with a plane, which yields model vertices that will form a connected area within the plane, and (2) using a scanline technique, appropriate positions within the planes' connected areas are searched.

Computation of connected areas

Figure 3.15(left) illustrates a single cross-section (plane) in x-y direction at position z_P. The small (red) dots on the right hand side symbolize intersections of model edges and plane.

Once the intersections have been determined, the vertices need to be connected appropriately. To this end, two vertices V_i and V_j are connected if and only if the edges E_i and

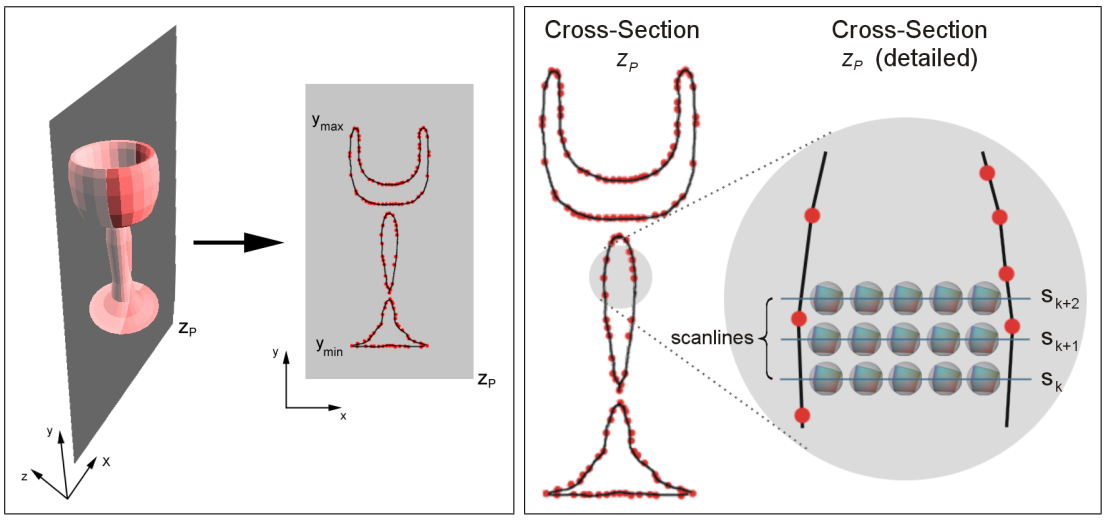


Figure 3.15: *Left: Cross-section of a model at position z_P and its intersections. Right: Positioning of* Carrier Objects *within a cross-section by exploiting scanlines.*

E_j belong to one and the same triangle in the original model. E_i and E_j are the edges whose intersections with the cross-section yielded V_i and V_j.

Placing Carrier Objects *within a cross-section*

For placing *Carrier Objects* within the connected area of a cross-section, a scanline technique is used (see Figure 3.15(right)). The increment by which a scanline is moved as well as the distance to be chosen between adjacent *Carrier Objects* located on the same scanline depends on the *Carrier Objects'* sizes.

Carrier Objects may not intersect with the polygonal mesh of the model. To verify this important aspect, each object's bounding sphere is tested with regard to potential intersections. Figure 3.15(right) illustrates the positioning of *Carrier Objects* for three scanlines.

Data recovery

Data recovery takes place in two steps: (1) the *Primary* and *Consecutive Carrier Objects* are searched and (2) the embedded data is decoded.

The unique characteristics of the *Primary Carrier Object* addressed before facilitate its detection. These include the triangle T_P, its normal, and the edge E_P (see Fig. 3.14). Those characteristics can then be used to compute the local coordinate system which, afterwards, serves for detecting the *Consecutive Carrier Objects* and for decoding the embedded data (see [SL05] for more details).

Figure 3.16 illustrates an example for which a watermark stream was embedded within a 3D model (vase of flowers). The watermark stream consists of further model components (water, flower) which can be displayed at will after decoding.

Figure 3.16: *Example for a watermarked 3D polygonal model. The data is coded inside the rim of the vase. The first two illustrations show the mesh of the vase with and without* Carrier Objects *included. The image on the right hand side shows the data encoded (models of water and flower, 2.4 MB).*

3.2.4 Discussion

Two *Illustration Watermarking* techniques were introduced which are different from the bulk of traditional watermarking techniques most of which directly modify the polygonal mesh (e. g., [Ben99, HB02, MSI01], see Sect. 2.3.3). The proposed techniques do not alter the original mesh.

The first technique exploits the textures associated with a model. Both *Watermarked* and *Data Textures* provide considerable space for descriptive metadata storage, in particular, when considering that each model component can be assigned at least one texture. However, the separation of model and textures can be associated with disadvantages.

The second technique adds *Carrier Objects* to the original model. Since those objects are structured the same as the model, model and objects form an entity after their integration. Also, due to the inclusion of each *Carrier Object's* local coordinate system during data recovery, the watermarked model is robust to common geometric transformations (translation, rotation, scaling). A drawback is that with an increasing amount of data to be encoded, the number of *Carrier Objects* increases, as well, which may raise the time required for model rendering. For comparison, an amount of 1 MB of encoded data yielded a performance of 8 fps on a PC with 900 MHz and 256 MB RAM (100,000 bytes of coded data yielded 70 fps). However, varying the complexity of *Carrier Objects*—increase in their complexity yields a decrease in their number—revealed interesting results. When embedding 1,000 bytes, less complex *Carrier Objects* (8 vertices) yielded lower rendering costs than complex *Carrier Objects* (100 vertices); but when embedding an amount of 100,000 bytes, rendering with complex objects was more than twice as fast as rendering

with less complex objects. An effect which can be attributed to internal hard- and software optimizations.

3.3 Concluding Remarks

This chapter addressed *Illustration Watermarking* techniques for graphics descriptively represented. In contrast to traditional watermarking techniques, which are still not robust to all imaginable attacks [Her02], *Illustration Watermarking* aims at transparently embedding the highest possible amount of data while robustness to criminally intended attacks is disregarded.

For most of the techniques suggested, transparency can be verified by analyzing the changes which are introduced to the cover medium in consequence of data insertion. A line as part of a vector graphic, for example, is still rendered the same regardless of whether or not the line was subdivided into line segments. Also, the *Carrier Objects* inside a 3D geometric model remain invisible to a viewer. Watermarks embedded within the texture of a 3D model, on the other hand, can introduce detectable changes to the texture which is a 2D raster graphic. The extent of those changes strongly depends on the technique which was chosen and the amount of data which was encoded. The subsequent chapters will shed light on those aspects.

4 Illustration Watermarks for Raster Graphics

The computer graphics dealt with in this chapter are—in contrast to the graphics addressed in Chapter 3—directly represented as matrices of pixel values each defines a color value at a specific position in its matrix. In this work, those graphics are referred to as *Raster Graphics*. Raster graphics are widely-used in the digital world, even though they are linked with several drawbacks such as lacking scalability or high storage costs in case of high quality. Often, raster graphics are generated by directly assigning colors to pixel matrices [BS95]. Examples are images taken by digital cameras, images digitized from paper documents, or even images which were created by directly painting on a digital device.

This chapter first analyzes the general structure of raster graphics and potential limitations associated with that structure in Section 4.1. It then introduces the basic procedure of embedding *Illustration Watermarks* in those graphics in Section 4.2. In Section 4.3, novel watermarking techniques are introduced that analyze image features for the watermark insertion. After this, Section 4.4 describes a technique that operates in the Wavelet domain, before Section 4.5 summarizes this chapter.

4.1 Raster Graphics—Their Characteristics and Limitations

According to Watson [Wat93], raster graphics are the most conventional method of representing images digitally. Using this technique, an image is stored in a 2D array of pixels each represents the intensity (or color) of the image at a point.

Mathematically, a pixel P is a tuple (S_P, I_P) with S_P being the extent of P in the screen plane (position and shape) and I_P being the color intensity [Fiu89]. The shape of a pixel is typically square (computer monitors) or rectangular (televisions). A pixel's color is specified according to the color space in use. Examples are the *RGB color space* (three bytes per pixel, one for red, green, and blue, respectively) or the *HSV color space* (Hue, Saturation, and Value).

Image quality

Besides the number of pixels of which an image consists (*image resolution*), the image quality strongly depends on the storage capacity which is available for each pixel. The higher the number of bits per pixel, the more color nuances can be stored and hence, the higher can be the quality of the image. In terms of images showing real world scenarios, high quality images appear more realistic than images of lower qualities.

The colors of an image are mostly represented in the RGB color space which requires a color depth of 24 bits per pixel. Decreasing the color depth of an image may involve image quality loss, even though not necessarily visible. Gray level images are typically represented by 8 bits per pixel, which allows to express 256 different gray scales for each pixel.

Image compression

The storage capacity demanded by an image can exceed the data space which is available on a device, in particular when the image is of high resolution. There exist various techniques with which an image's memory requirements can be reduced, which results in image compression. In general, image compression techniques can be divided in lossy (transformation is not invertible, e. g., *JPEG* compression) and lossless compression (transformation is invertible, e. g., *Huffman coding*). The latter is enabled due to redundancies in certain representations.

Color Quantization (e. g., [GP90, Hec82]) is an example for a lossy compression technique by which a set of representative colors is chosen from the pool of all colors an image contains. For each pixel in the image, a color from the reduced set of colors is then selected that best fits the pixel's original color. Typically, the set of colors is referred to as *Color Look-up Table* which contains a maximum of 256 RGB colors. Hence, each pixel's new color can be represented by an 8 bit pointer that refers to an item in the color table. *Color Cell Compression* [CDF+86] is a technique that goes even one step further. It allows to represent a pixel's 24 bits color value by no more than approximately 2 bits.

Why are image quality and image compression important with regard to *Illustration Watermarking*?

The *Human Visual System (HVS)* is extremely sensitive to color variations and able to distinguish about 10 million colors under optimal viewing conditions [ST97]. Those abilities call for digital images of high qualities, which, on the other hand, involves high memory requirements. However, the more bits are available per pixel, the more color shades can be stored and hence, slight color changes introduced to individual pixels can be less easily detected by a viewer, which is a precondition for watermark encoding.

But often, high storage capacities are undesired, which results in compression techniques appliance. However, in conjunction with digital watermarking, lossy compression can damage the data encoded as *Illustration Watermark*. Lossless compression, on the other hand, ensures the preservation of the encoded data, which is at the expense of high compression rates. But even the latter class of compression techniques allows to considerably reduce an image's memory requirements.

4.1.1 File Structure of Raster Graphics

The data representation of raster graphics typically includes two main parts: the *File Header* and the actual *Image Content* in terms of color data (see Fig. 4.1).

The file header consists of information which is necessary to identify the image file format and to interpret the color data of the image so that it can be displayed correctly. This comprises, for example, the image dimensions, the color model, or information about the compression technique that was applied (see [Dig00]). Furthermore, the file header can include information about the image (descriptive metadata).

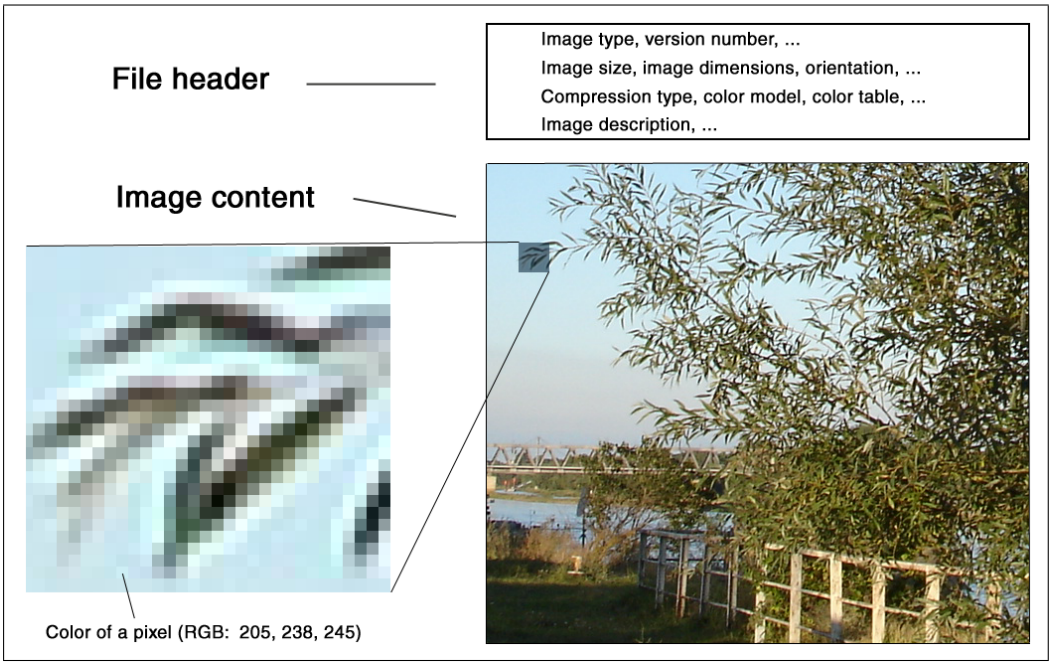


Figure 4.1: *Basically, the data representation of a raster graphic comprises a file header and the image content which is represented by a matrix that consists of pixels with specific color values assigned.*

Typically, file header data is directly followed by the color data defining the content of the image. It consists of an array of color values which are arranged as a 2D matrix according to the information specified in the file header. In case of the RGB color model, each color value is represented by three bytes, one for each color channel: red, green, and blue.

Widely-used raster graphic file formats are *JPEG* (*Joint Photographic Experts Group*), *PNG* (*Portable Network Graphics*), *TIFF* (*Tagged Image File Format*), *BMP* (*BitMaP*), or *GIF* (*Graphics Interchange Format*). Further reading on graphics file formats and their data representations can be found, for example, in [BS95, MvR96, Mia99].

As to *Illustration Watermarking*, the second part of a raster graphic's data representation—the information specified in terms of color data—is exploited for data encoding. In contrast to file header data, the color data of an image does not vary between different image file formats that use the same color model. Hence, encoded data survives file format conversions as long as the color information is not reduced or modified in any other way during the conversion.

4.1.2 Static Raster Graphics Made Dynamic

A raster graphic is, in principle, displayed as a static image. In this regard, it cannot be distinguished from images in printed media. However, the fact that a user can utilize a device such as a computer mouse and its displayed cursor to trigger events affecting the actual visualization is definitively an extension to what printed media permit. The information which is necessary to turn a static raster graphic into a dynamic medium can be stored as *Illustration Watermark* within the graphic itself.

Interaction with raster graphics that may have dynamic elements can serve various purposes. For example, images can be augmented with information (*image enrichment*) which is hidden by nature but can be requested on explicit user demand. Images with such dynamic contents can range from images which are interactively explorable to images in which layered elements can serve purposes such as dynamic highlighting or motion.

The author of a dynamic image may also assign diversified information elements which are solely intended for specific end-users or which are shown only within specific presentation contexts (*user or context dependent visualizations*). An example for context dependent visualizations are images of an electronic consumer product which must provide other information for production than for advertising. By exploiting the technique of *Illustration Watermarking*, these presentation styles can be included in the same image.

4.1.3 Concluding Remarks

Raster graphics are optimized for presentation on raster displays. They are not optimized for interacting with them. The necessary information to identify image regions that represent image objects or to visualize image objects, which are occluded by others, in their entireties does not exist.

Even though a few image format specifications allow for the inclusion of descriptive metadata, often limited in extent and form, the manner of how the data is represented considerably differs between those specifications. This complicates metadata handling for software that deals with varying image file formats.

However, applications for interactive or dynamic raster graphics exist, as selective examples had shown and will show in the course of this work. *Illustration Watermarking* is a powerful technique which is capable of providing the information necessary to overcome the limits associated with raster graphics being static in their original purpose of use. In the following, spatial as well as transformed domain techniques (see Sect. 2.3.3) are proposed that allow encoding considerable data rates imperceivably to turn static raster graphics into dynamic media.

4.2 Content-Based Spatial Domain Watermarking

Illustration Watermarking is a content-based digital watermarking technique, which means data is encoded only into those image regions with which it is semantically associated. Hence, the data remains attached to its image region even if the region gets separated from the image.

This section describes the basic components necessary to create images with dynamic contents by exploiting *Illustration Watermarks*. To this end, one of the most frequently used watermarking techniques is analyzed and its implementation for the purpose of *Illustration Watermarking* is discussed.

4.2.1 The Watermarking Pipeline

The basic components which are necessary for *Illustration Watermarking* with regard to raster graphics are:

- The ***Original Cover Image*** (Fig. 4.2(a)) being the image into which the data will be encoded. The image which results from encoding (watermarked image) will slightly differ from the original image.
- The ***ID-Buffer*** (Fig. 4.2(b)) is a 2D raster graphic which has the same size as the cover image. Its depth of 8 bits per pixel allows to specify up to 256 different regions, each with a unique color. One of these colors is reserved as background color. The individual regions, which must not intersect one another, mark those regions within the cover image to be watermarked.
- A ***Watermark Stream*** (Fig. 4.2(c)) is the data to be encoded into a specific region of the cover image. Each image region which is associated with a marked region within the ID-buffer has its own watermark stream. The watermark stream can include binary data items of varying types (e. g., text data item, image data item, etc.). Its size is limited by the number of pixels available in the corresponding image region.

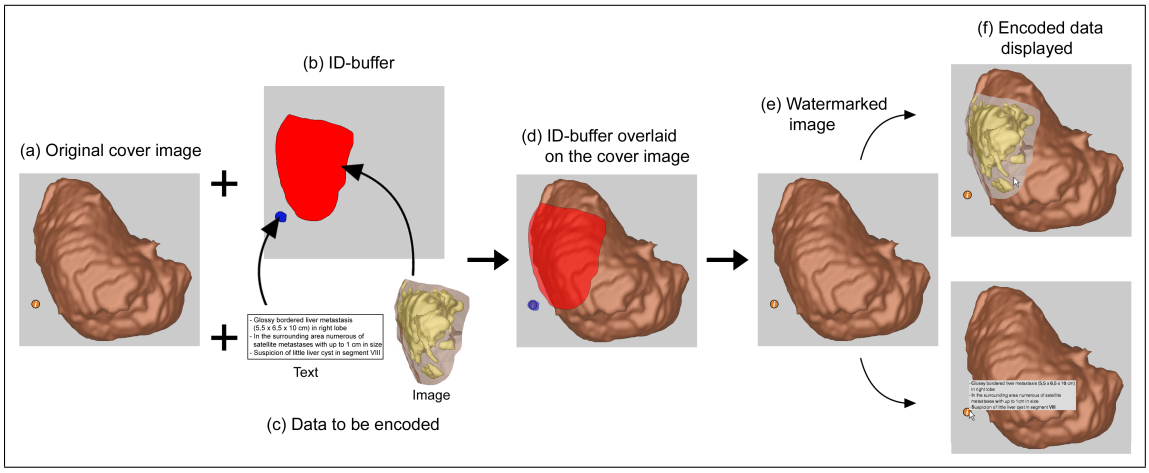


Figure 4.2: *Data can be encoded into specific regions of the original image by using an ID-buffer which serves as reference for identifying those regions. The watermarked image, which results from applying a watermarking technique with adapted parameters, can be explored by mouse cursor movement.*

Figure 4.2 illustrates *Illustration Watermarking* with an example image. Image (a) shows a rendering of a model of a human liver and a small circular region labeled with an *i* indicating that further information is available. The ID-buffer (b) has two regions differentiated by color. Overlaid on the original image (d), the ID-buffer regions correlate with a part of the liver and the information symbol. The regions of the original image marked by the ID-buffer are intended to be modified for data insertion.

Two types of data (c) are to be inserted: a text message and a part of another image (part of the liver transparently rendered). These data items are each included in separate watermark streams linked with regions in the ID-buffer. Before data insertion starts, it is verified that the data amount does not exceed the space a region provides. If verification succeeds, data encoding can commence by choosing a watermarking technique that modifies the colors of the cover image without introducing perceivable artifacts.

Figure 4.2(e) shows the watermarked image that allows for exploration. The transparent rendering that appears once the mouse cursor reaches the according region enables a viewer to look inside the liver, which reveals an otherwise occluded tumor. Moving the cursor over the information symbol triggers a text box to be displayed.

4.2.2 Images Scaled Down to Bit Level

In many digital watermarking approaches, data is embedded by modifying specific color bits, often the least significant bit (*LSB*) of a color channel. Since the human visual system is least sensitive to the blue color—assuming that only red, green, and blue components are present (RGB color model)—this color channel is mostly targeted for data insertion. However, several approaches include other color channels, as well, and exploit more bits than only the LSB.

Why is it possible to alter individual bits without the introduced changes being perceivable to a viewer?

When a regular image (e. g., a photograph with structured information) is split into its bit-planes (see the ape's face in Fig. 4.3), it oftentimes can be observed that the lower bit-planes (in Fig. 4.3 bit-planes 1 to 3) contain unstructured information which is perceived as noise. Replacing noise by watermark stream data, which is typically also perceived as noise, has no influence on the appearance of the image.

But how about images such as cartoon images which have large evenly colored areas? Their lower bit-planes appear less noisy than those of photographs. An extreme example is illustrated in Figure 4.3. The original image has eight areas with different gray levels on its right hand side (the numbers denote the levels). These areas appear uniform within the bit-plane images, which is in contrast to the noisy regions in the rest of these images.

What would be the effect of encoding a watermark stream that introduces noise into evenly colored regions?

The images in Figure 4.4 aim at illustrating the effect of encoding data into gray level images. The image into which data was encoded consists of two types of subimages each

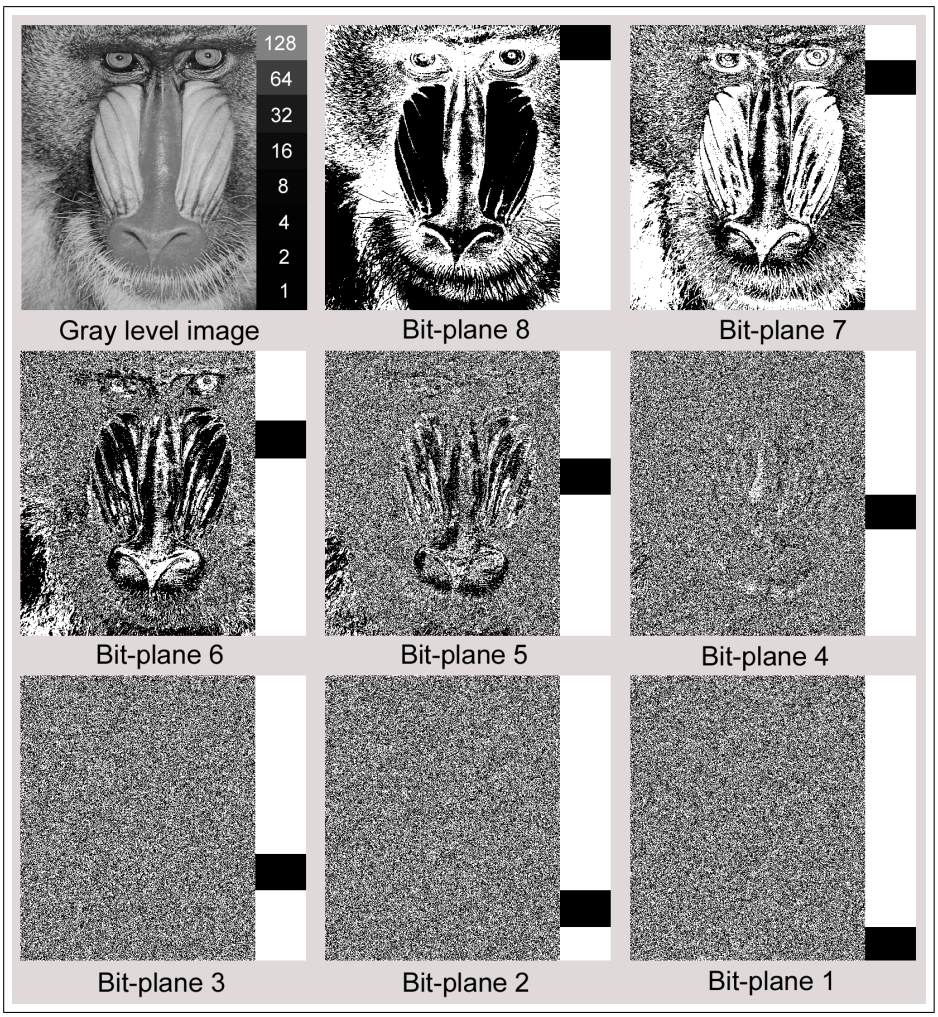


Figure 4.3: *A gray level image, composed of an ape's face and eight different gray levels, is split into its bit-planes. Black indicates that a single bit is set at a specific position within a bit-plane.*

of which is shown three times. The subimage types are in high contrast to each other: (1) eye section of the ape in Figure 4.3 and (2) white image (RGB: 255, 255, 255).

Into each copy of the two subimage types, differently structured watermark streams (Fig. 4.4(a) to (c)) were encoded, limited to areas marked by the word "DATA". The watermark streams consist of: (a) a sequence of the number "51" expressed as bytes, (b) a sequence of the string "data", and (c) a sequence of random characters. The watermark streams were only encoded into the first four bits (bit-planes 1–4) of each pixel located within the marked regions. Hence, number "51", for example, induced the according bits in bit-planes 1 and 2 to be set whereas the bits in bit-planes 3 and 4 were unset.

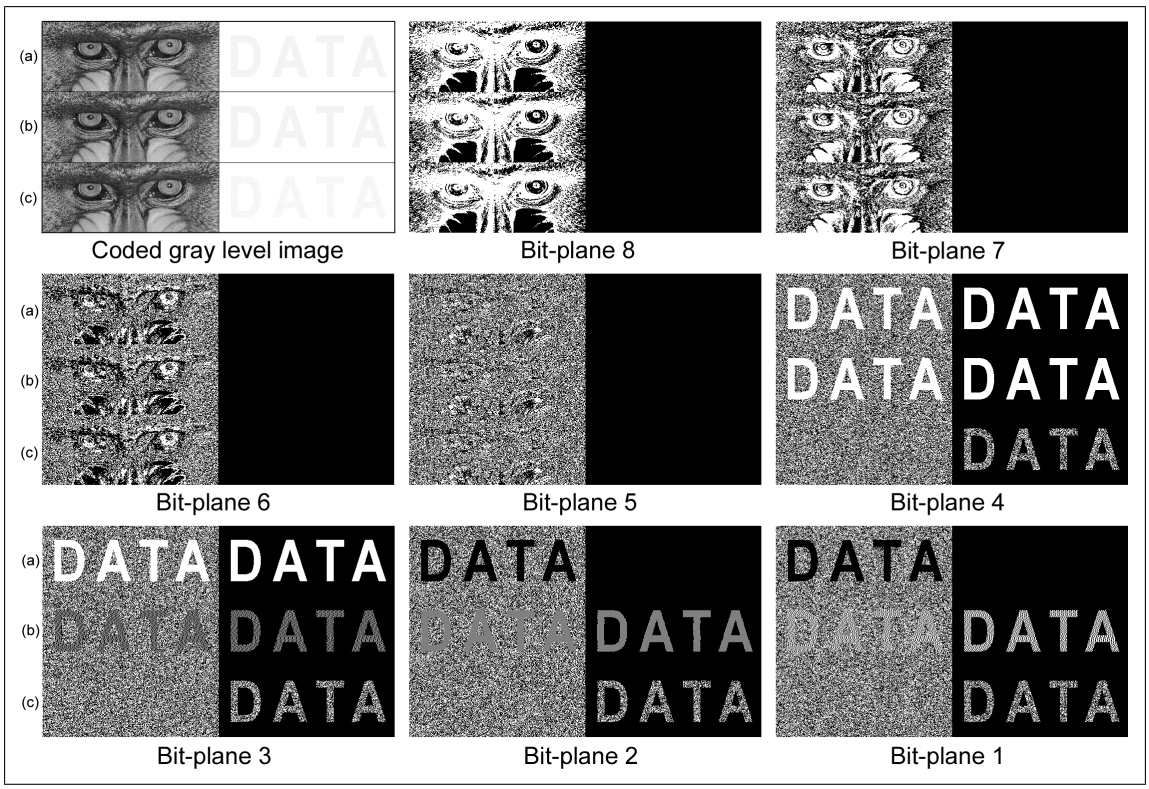


Figure 4.4: *Bit-planes of a watermarked image. The original image is composed of two image types (ape's eye section and white area) each appears three times. Into each of the six subimages, a watermark stream was encoded, varying from line (a) to line (c) and limited to regions marked by the shape of "DATA". The three types of watermark streams are: (a) a sequence of "51" expressed as bytes, (b) a sequence of the string "data", and (c) a sequence of random characters.*

What information can be extracted from Figure 4.4?

The introduced modifications due to data insertion can solely be detected within the white subimages. Thus, signals transmitted through the texture of the ape's eye section seem to inhibit the perception of watermark signals, which indicates that image texture considerably affects watermark perception.

But also the structure of the watermark stream affects its perception, as can be seen in the white subimages. The regions into which the sequence of "51" was embedded (Fig. 4.4(a)) are more eye-catching than the regions with the random watermark stream encoded (Fig. 4.4(c)). This is due to the strong concentration of the watermark signal in bit-planes 3–4, which is in contrast to the scattered watermark signal consisting of random characters. It thus can be stated that an increase in contrast between the watermarked region and the rest of the image within a bit-plane image yields watermark detection to happen more spontaneous.

As to the ape's eye section, when analyzing bit-planes 1–4, the embedded watermark

stream appears differently strong. While the random watermark stream (line (c)) perfectly merges with the bit-plane images, the two remaining watermark streams (lines (a) and (b)) are silhouetted against them. This aspect, however, seems to have no effect on watermark perception in the watermarked gray level image.

Figure 4.4 already reveals that the texture of an image has a significant effect on whether changes introduced to an image in the course of watermark encoding can be detected. But it does not provide any indication of how influential this effect with respect to differently textured images is.

Also, it is obvious that modifying bits within the lower bit-planes influences the appearance of an image less strong than altering bits of higher order. However, the question of up to which bit-plane data can be imperceivably encoded is still unanswered.

4.2.3 A Basic Illustration Watermarking Approach

As it was described in Section 4.2.1, content-based *Illustration Watermarking* requires at least three components: the cover image, an ID-buffer, and the data to be embedded. The regions into which the data is to be encoded are identified with the ID-buffer. The question of how those regions can be identified during watermark recovery must be answered as well as questions regarding watermark stream preparation and the watermarking technique itself.

The *Illustration Watermarking* techniques which are described in this chapter are applied exclusively to images whose color information is specified according to the RGB color model. For each pixel, up to 24 bits (8 bits per color channel) can thus be exploited for watermarking, which yields each pixel to serve as metadata container. Needless to say that modifying each of the 24 bits would result in perceivable image artifacts, as the previous section revealed. Consequently, only a selection of these bits—typically the bits in the lower bit-planes—is exploited.

What is the information that needs to be coded beside the actual watermark stream?

Three types of information are essential:

1. Clear marks that enable identifying a region into which a watermark stream was inserted.
2. The shape and extent of a watermarked region.
3. The parameters of the watermarking technique specified, which can vary depending on the space provided by a region or the size of the watermark stream.

Outlines and seed points

In a watermarked cover image, a coded region is separated from uncoded or other coded regions by a closed outline. This outline whose coordinates are computed according to the marked region provided by the ID-buffer defines shape and extent of the watermarked region.

A pixel becomes part of the outline when it features two attributes: (1) at least one of its four direct neighboring pixels is part of the region to be watermarked and (2) at least one of its four direct neighboring pixels is not part of that region. For outline identification in the watermarked image, each of its pixels is marked by setting the LSB of its blue color channel to 1 whereas the LSBs of all other pixels are unset.

An alternative to directly marking a region's outline by altering the corresponding pixel colors would be to encode the geometry that describes an outline (e. g., circle or rectangle attributes, individual points of a polygon, etc.). This technique would allow for overlapping regions, which is not permitted by the technique which was actually chosen. However, a drawback of this technique would be the considerable amount of additional data to be coded, particularly when the region is defined by a polygon consisting of many points.

In addition to its marked outline, each region has one prominent pixel: the *Seed Point*. The seed point and its neighboring pixels have specific characteristics with which a watermarked region can be identified during decoding. One of these characteristics is the position of the seed point, which is chosen with respect to the outline. When $P_S(x, y)$ is a potential position of a seed point and C_S is the color of its corresponding region in the ID-buffer, the following criteria must be fulfilled:

- P_S's corresponding position in the ID-buffer must have color C_S which is different from the ID-buffer's background color.
- P_S is not part of an outline.
- The pixels at positions $(x - 1, y)$ and $(x, y - 1)$ (northwest) are part of an outline.

When a position for a seed point is searched, the image is scanned top down and from left to right. The first position that fulfills the criteria is chosen. Both the position of the seed point and an ID which is assigned to the seed point and its neighboring points enable a clear identification of the seed point.

In addition, there are two other purposes a seed point serves: (1) it stores information about the watermarking technique which was employed by encoding a specific ID in its color channels and (2) its position within a region's outline defines the inside of the outline so that the remaining pixels of the watermarked region can easily be found.

Embedding the watermark stream

The data to be embedded, which can consist of various data items (e. g., text, image or auditory data), is preprocessed so that it fits into a single binary data stream. This data stream is then compressed, which yields the final watermark stream.

The watermarking procedure itself comprises the identification of the pixels to be modified using the ID-buffer as well as the selection and use of an appropriate watermarking technique with which the whole watermark stream can be encoded.

Data encoding starts with identifying the pixels in the cover image which are located inside a marked region within the ID-buffer. For these pixels, up to the four least significant

bits in each color channel are then replaced by bits of the watermark stream. The number of bits eventually used is adapted to the data space needed. Thus, at most 11 bits per pixel can be modified. The 12th bit, which is the LSB of the blue color channel, is reserved for outline identification.

During data encoding, the blue color channel has the highest priority while the green channel has the lowest. These priorities are based on findings about the human visual system which is least sensitive to blue and most sensitive to green [Gol96]. To insert, for example, six bits per pixel, three bits are replaced in the blue channel (including the bit for outline identification), and two bits are replaced in the red and green channels, respectively.

Recovering the encoded data

The described technique is a blind watermarking technique which means that the original image is not required for data recovery.

The watermark decoding starts with locating seed points which can be clearly identified due to their specific characteristics. Once a seed point has been detected, the remaining pixels of the associated region—enclosed by its outline—can be retrieved. For recovering the data embedded in those pixels, the watermarking technique which was applied must be known. The according information is provided by the ID encoded in the seed point. Finally, the data bits are extracted from the pixels' colors and added to the watermark stream which contains the descriptive metadata of the region.

4.3 Techniques that Include Image Texture

The watermarking technique which was described in the previous section does not consider the texture of an image. The technique rather encodes the same amount of data into each pixel, provided that the pixel correlates with a marked region in the ID-buffer. However, Figure 4.4 indicated that the texture of an image has significant influence on watermark perception. This section hence introduces techniques that analyze image features previous to data embedding so that the encodable data amount can be adapted for each pixel.

4.3.1 Human Color Vision and RGB Color Space

Within the human eye, an image results from light which is focused onto the retina by the eye's lens. Humans can see light with wavelengths between approximately 400 nm (violet) and 700 nm (red) [MF97]. The greater the amplitudes of light waves, the brighter they are perceived. A human eye has about 5 million cones which are responsible for color vision. There are three types of cones each of which has its maximum response to light at a certain wavelength: short wavelength cones (about 419 nm), medium wavelength cones (about 531 nm), and long wavelength cones (about 558 nm) [Gol96]. Our peak sensitivity is to light with a wavelength of about 560 nm, which corresponds to yellow-green light.

RGB color space and its limitations

Mathematically, colors can be specified by positively weighted sums of the primary colors red, green, and blue (RGB color space). *CRT* (*Cathode Ray Tube*) displays exploit phosphor dots of these colors. Since those displays are the primary target, the RGB representation is most common these days [ST97].

However, the RGB color space is not the perfect color representation with respect to human color vision. On different devices, for example, the color gamut can vary which means the RGB color space is device-dependent. That is, instead of the color model, the output device defines what exactly is meant by "red", "green", or "blue" [FvDFH96]. Also, only a small range of all the colors humans can perceive can be expressed by means of RGB components. Furthermore, the manner in which colors are specified—namely in terms of its red, green, and blue components—is not in accord with human nature.

Another important drawback is that the RGB color space is not perceptually uniform. This means in one particular part of the color space, the colors at two points a certain distance apart can be distinguished. After moving the points to a different position within the color space, the colors at those new points can be indistinguishable, even though the distance between the points was kept.

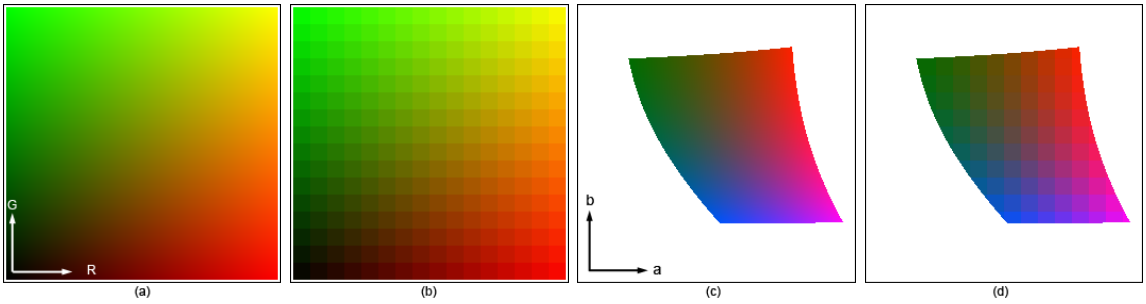


Figure 4.5: *Images (a) and (b) show representations of the RGB color space (const. blue=0) whereas images (c) and (d) are visualizations of the CIELAB color space (const. L=62). The white area in the CIELAB representation indicates that this region, in fact, has CIELAB color values but they cannot be expressed as RGB values demanded by the display. In both types of color representation, images (b) and (d) were subdivided into blocks of equal sizes. The color in each block represents the averaged colors covered by the block.*

CIE XYZ color model

Several color models have been developed that take into consideration human vision (e. g., *HSV* or *HLS color model*). In this context, the *International Commission on Illumination (CIE)* established three standard primaries referred to as X, Y, and Z, which are based on human perception. Their *CIE XYZ* color model is device-independent and its primaries can be combined to match any color a human is capable to see. According to the CIE model, color is described by a luminance primary Y (defined to match the luminous-efficiency function of the human eye [FvDFH96]) and the curves of the two primaries X and Z,

which were obtained from experiments with humans involved.

*CIE L*a*b* (CIELAB)* is a non-linear transformation of the CIE primaries, which almost results in a perceptually uniform color space. Color differences can thus be specified as Euclidian distances [Sto03]. The components of the three-dimensional CIE L*a*b* model represent the color's luminance (***L***, $L = 0$ yields black, $L = 100$ yields white), its green/red part (***a***, negative value indicates green, positive value indicates red), and its blue/yellow part (***b***, negative value indicates blue, positive value indicates yellow).

Figure 4.5 shows two images of a particular face of the RGB color cube and two images of a slice of the CIE L*a*b* color shape. For each of these two types of color representation, the left hand side image ((a) or (c)) shows a standard visualization of the respective color model. The right hand side image ((b) or (d)) shows a visualization in which the image was segmented into blocks of equal sizes whose colors represent the averaged colors covered by a block.

At first view, it can be observed that the RGB block-representation (b) contains several blocks which seem to have equal colors. These blocks accumulate in certain regions (e. g., green region in the upper part) whereas other regions consist of almost only well-confined blocks. This illustrates that even though the color differences are mathematically equal, the perceived differences are far from being equal.

A second observation which can be made is that the space covered by the CIE L*a*b* representation is considerably smaller than the space taken by the RGB representation, even though CIE L*a*b* includes all the colors a human can see. This is due to the fact that the RGB representation comprises many colors which are perceived the same. These visually redundant colors can hence be mapped onto a single color defined by the CIE L*a*b* color model. Figure 4.6 illustrates this aspect. It consists of three images which seem to be equally colored. However, the colors they contain differ in terms of their RGB components.

The watermarking technique which is described in the following section exploits those color redundancies by including the CIE L*a*b* color model. This technique was developed by Michael Specht [Spe06], a student I supervised in the course of my research.

4.3.2 Watermarking Approach based on the CIELAB Color Model

When a watermarking technique directly modifies the least significant bits of a color channel, it firstly resets those bits to 0, which reduces the color space by an amount that depends on the number of bits involved. Modifying two bits of each color channel, for example, subdivides the RGB color space into small cubes of size 4×4×4. Consequently, instead of 16,7 mill. colors, only 262,144 classes of colors can be differentiated. Each of these 262,144 classes covers 64 colors (equals a capacity of 6 bits per pixel). Which of these colors eventually replaces the original color in the cover image is determined by the watermark stream. The assumption behind this approach is that each of the 64 colors is perceived the same.

A similar approach can be applied to the CIE L*a*b* color space. But its advantage over the RGB color space is that already the non-subdivided CIE L*a*b* representation

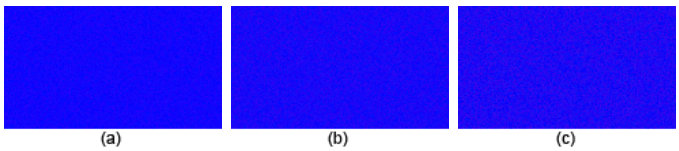


Figure 4.6: *Various RGB color values (perceived as the same blue color) can be assigned to classes of CIELAB colors (the more classes, the less elements): (a) 855 different RGB colors (class unit sizes: 1×1×1, 2^{24} CIELAB color classes), (b) 1630 different colors (class unit sizes: 1×2×2, 2^{22} CIELAB color classes), and (c) 5943 different colors (class unit sizes: 2×4×4, 2^{19} CIELAB color classes).*

allows to encode data, as Figure 4.6(a) demonstrates. When 855 different RGB color values can be mapped onto a single CIE L*a*b* value, about 9.74 bits of the watermark stream ($log_2 855$) can be embedded. However, since for each CIE L*a*b* value the number of corresponding RGB values can vary, the amount of data which can be encoded per pixel varies, as well.

Similar to the RGB color space, the amount of encodable data can be increased by subdividing the CIE L*a*b* color space. Since humans are less sensitive to variations in color compared to changes in luminance [Gol96], the CIE L*a*b* space can be subdivided into class units of size 1×2×2 ($\boldsymbol{L}\times\boldsymbol{a}\times\boldsymbol{b}$), which corresponds to 256 levels of luminance and 128 color levels for a and b, respectively. Note in this context that due to their varying ranges—from -100 to 100 and from 0 to 100 (see Sect 4.3.1)—a subdivision requires to normalize each component of the color space to an 8 bit range.

Nevertheless, studies revealed that humans are by far not able to distinguish 256 levels of luminance, which allows for further subdivision (e. g., 2×4×4, which yields 2^{19} color classes, see Fig. 4.6(c)). Hence, expanding a CIE L*a*b* color class so that it contains more elements results also in an increasing number of RGB color values which can be mapped onto a particular class element.

Description of the technique

The technique aims at encoding a watermark stream into a RGB color image by taking advantage of the CIE L*a*b* color space which is well adapted to human perception. To this end, the RGB color space is subdivided into classes each of which corresponds with one or more CIE L*a*b* colors. The number of elements in each class can vary (see Fig. 4.7(b)). During this preprocessing step, a look-up table is generated which is employed during data encoding.

Before data encoding can start, the watermark stream is divided into blocks of equal sizes. In practice, a block size of 1024 bytes emerged as acceptable compromise between avoiding capacity loss and decreasing efficiency. For illustration, the block size in Figure 4.7 was chosen to be 8 bits.

For watermarking, the image is scanned top-down and pixel by pixel. Once information regarding the watermarking method (e. g., ID, block size, etc.) has been encoded into the first pixels, embedding the watermark stream proceeds according to Algorithm 1. The task is to find the new color $\widetilde{C_p}$ which will replace the color of the current pixel P_{cur}. This

Algorithm 1 *Determining the new color $\widetilde{C_p}$ for a pixel which will represent a portion of the watermark stream.*

1: $B_{cur} \Leftarrow$ First block of watermark stream
2: $P_{cur} \Leftarrow$ First pixel
3: **while** $B_{cur} \neq$ End of watermark stream **do**
4: $N_{B_{cur}} \Leftarrow$ Decimal number represented by B_{cur}
5: **while** $N_{B_{cur}} > 0$ **do**
6: $C_p \Leftarrow$ Color of P_{cur}
7: $Class_C \Leftarrow$ CIE L*a*b* class of RGB colors which contains C_p
8: $S_{Class} \Leftarrow$ Class size of $Class_C$
9: $I_{\widetilde{C_p}} \Leftarrow N_{B_{cur}}$ modulo S_{Class} // (index of new color $\widetilde{C_p}$ in $Class_C$)
10: $N_{B_{cur}} \Leftarrow N_{B_{cur}}$ / S_{Class} // (integer division)
11: $P_{cur} \Leftarrow$ Next pixel
12: **end while**
13: $B_{cur} \Leftarrow$ Next block of watermark stream
14: **end while**

new color is identified with its index $I_{\widetilde{C_p}}$ when it is assumed that all class elements are consecutively numbered in the range of 0 to $S_{Class} - 1$.

To enable identifying the end of the current data block during watermark recovery, it can be required to include pixels in addition to those already representing the block. This is necessary when multiplying all the sizes of those classes which are associated with previously coded pixels yields a number less than 2^{8192} in case of 1024 bytes block size (handling numbers of this size is enabled by specific libraries such as the *GNU Multiple Precision Library*). The color to be chosen for such a pixel is the first element of its corresponding class.

Figure 4.7 illustrates the technique with an example. After the watermark stream was subdivided into 1 byte blocks, the decimal numbers represented by the first two bytes are 127 and 96. The first pixels to be watermarked are magnified and labeled with $c_0, d_2, c_5, f_0, d_5, d_3$, and g_4. Their colors can be assigned to certain classes (b) whose elements were selected by exploiting the CIE L*a*b* color space.

The particular example takes six pixels to encode the first 16 bits of the watermark stream. As a result, the first byte can be represented by c_1, d_2, and c_2:

$$\begin{aligned} 127 \; mod \; 9 &= 1 \quad (c_1) \\ (127/9) \; mod \; 6 &= 2 \quad (d_2) \\ (14/6) \; mod \; 9 &= 2 \quad (c_2) \\ (2/9) &= 0 \quad \text{(completed byte encoding)} \end{aligned}$$

For watermark recovery, the encoding procedure must be reversed. The required parameters are the class sizes and color indices, which can be retrieved from the watermarked

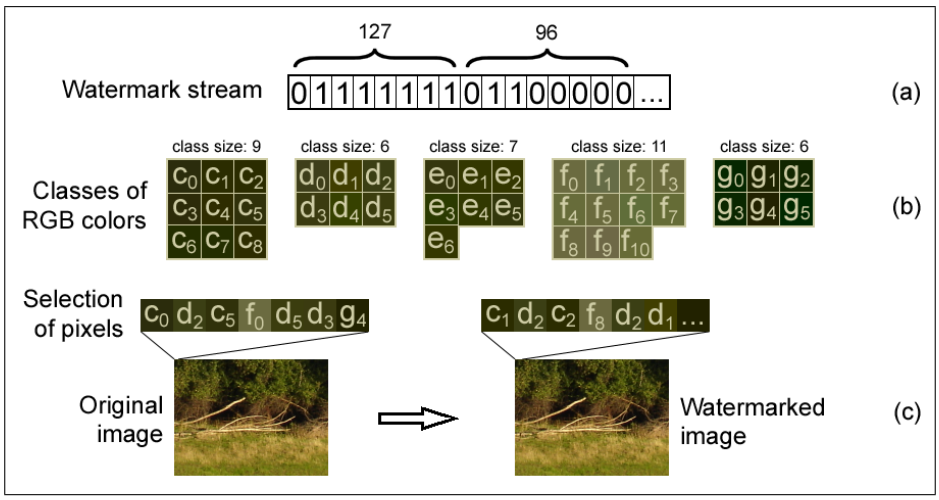


Figure 4.7: *The watermark stream to be encoded is split into byte blocks (a). Portions of byte blocks are then represented by certain colors (c) which are elements of classes consisting of similar RGB colors (b). These colors are determined according to the decimal numbers represented by each byte (see Alg. 1).*

image and the color look-up table. Since the look-up table can be created independently from the original image, the technique can be characterized as a blind watermarking technique. Decoding the first three pixels in reverse order yields the first byte encoded:

$$\begin{aligned} 0 * 9 + 2 &= 2 \quad (c_2) \quad \text{(size: 9, index: 2)} \\ 2 * 6 + 2 &= 14 \quad (d_2) \quad \text{(size: 6, index: 2)} \\ 14 * 9 + 1 &= 127 \quad (c_1) \quad \text{(size: 9, index: 1)} \end{aligned}$$

The end of a data block can be determined by multiplying the class sizes of those classes that include the colors of the pixels to be decoded, as it was described earlier.

Results and discussion

Figure 4.8 shows three images (from left to right): the original image, an image in which the least significant bits were modified, and an image to which the technique described in this section was applied. In contrast to the second image in which clear color shades in the sky region can be perceived, the right hand image can hardly be distinguished from the original image. The disturbing color shades in case of the image watermarked using the LSB technique result from modifying bits without considering aspects of human perception.

The proposed technique fulfills several criteria in regard to *Illustration Watermarking*. First of all, a considerable amount of data can be embedded invisibly. Compared to the LSB approach described in Section 4.2.3, the watermarked images are of higher qualities, i. e., it is hard to distinguish between original and watermarked image by viewing, even when large amounts of data are encoded (9 bits/pixel and more).

Figure 4.8: *Comparison of differently watermarked images. The left image is the original image with no data embedded. The second image was watermarked using the technique described in Sect. 4.2.3 (9 bits/pixel encoded, 3 bits in each color channel). The last described technique was used to embed on average 9 bits/pixel into the right image.*

Due to its lack of adaptivity to image features, the LSB approach allows to predetermine the exact amount of data which can be embedded into a certain region, which the dynamic approach presented in this section does not. Here, the amount of encodable data strongly depends on which colors are present in the cover image and on their respective class sizes. However, once determined, the capacity can be adjusted by altering the class sizes.

Besides its higher computing time in comparison with the LSB approach, another important drawback with respect to *Illustration Watermarking* is that the proposed technique embeds the watermark stream into the whole image instead of certain regions. To enable region-based watermarking, a potential solution is to reserve the first color in each color class for outline identification. In doing so, the color of a pixel which is part of the region's outline can be replaced by the first element of its corresponding color class. Consequently, this color cannot be employed for watermark stream encoding. The indices of class elements must hence be shifted, which yields index 0 to be associated with the second element. The decrease in encodable data capacity, which is linked with region-based watermarking, can be tolerated.

4.3.3 Approach based on Entropy and Color Analysis

The approach proposed in this section was developed in collaboration with Andrea Unger. The basic idea behind this approach is to generate a *Capacity Map* that indicates—for each color channel of a pixel—the number of bits (at most 4) which can be replaced by bits of the watermark stream.

The technique involves analyzing an image's information content and its color intensities. Both subtasks are not new in the context of digital watermarking (see Sect. 2.3.3). Van Droogenbroeck and Delvaux [vDD02], for example, suggested to compute the entropies of pixel blocks to determine the numbers of bits to be modified. Lie and Chang [LC99], as a second example, derive a *threshold function* from intensity values which is then used for watermark encoding. However, both techniques were developed to be applied to gray

level images. Also, both techniques were used in separate approaches, combining them is new.

The basic ideas behind the two techniques can be exploited to develop an *Illustration Watermarking* technique. But for this purpose, they need to be adapted so that they can be combined and so that they are applicable to color images.

Entropy inclusion

The entropy of an image, which is the average number of bits required to represent a sequence of color values, gives hints for the image's information content. High entropy, for example, indicates that an image provides a lot of information and has high variance in color tones. It also means that there is a considerable chance that the weaker signals of a watermark stream to be embedded get masked by other signals within the image (for more details regarding *masking*, see Sect. 2.3.3).

Illustration Watermark insertion is typically limited to certain image regions. Thus, the entropy of solely those regions is of interest. But even within those regions, the entropy can vary to a considerable amount. Hence, image regions to be analyzed are scaled down to pixel blocks of size 8×8.

$$E = -\sum_{i=0}^{K-1} H_p(i)\ log_2\ H_p(i) \tag{4.1}$$

The entropy E of a selection of pixels (each color channel is examined separately) can be computed using Equation 4.1. i is the potential range of color values ($K = 16$ in case of 4 bits to be considered) and $H_p(i)$ are the probabilities for the presence of those color values.

Since in this approach a maximum of 4 bits can be replaced in a color channel during watermarking, only the 4 most significant bits can be considered for entropy computation. This is to ensure that equal entropies are computed before and after watermark insertion. Algorithm 2 determines which bits can be replaced depending on computed entropies. $Cap = 7$, for example, means that at most three bits can be replaced. The entropy thresholds were chosen according to the work presented by VAN DROOGENBROECK and DELVAUX [vDD02] and according to informal test results.

Effects of color saturation

The number of bits of a color channel which can be replaced by watermark bits significantly decreases when the cover image has a homogeneous texture and when entropy is the parameter which is deciding. But even homogeneous textures, which have the same low levels of entropies, can conceal encoded watermarks differently strong, as Figure 4.9 illustrates.

The figure shows six classes of colors which have at least one color channel highly saturated whereas the remaining channels have a value of 0 assigned. Each of the depicted color squares contains a circle whose color was slightly altered with respect to the color of its enclosing square. The amount to which a particular color channel was altered is 15.

Algorithm 2 *Determining watermark bit capacities depending on computed entropies. Note that only the four most significant bits of a color channel serve for entropy computation.*

```
1: for all pixel blocks of size 8 × 8 do
2:   if entropy > 2 then
3:     Cap ⇐ 15
4:   else if entropy > 1 then
5:     Cap ⇐ 7
6:   else
7:     Cap ⇐ 3
8:   end if
9: end for
```

The figure reveals several interesting aspects the most important of which are:

- Regarding colors red, green, and blue, changing the green color channel in case of green ($Green - \Delta Green$ pair) is more perceivable compared to pairs $Red - \Delta Red$ and $Blue - \Delta Blue$. This can, most likely, be attributed to the special sensitivity of the human visual system to green tones.
- For each color class, altering the highly saturated color channel(s) is most perceivable. Thus, red, green, and blue are more suited for watermark insertion than yellow, cyan, and magenta.
- Altering color channels which are unsaturated (value of 0) has no effect on color perception.

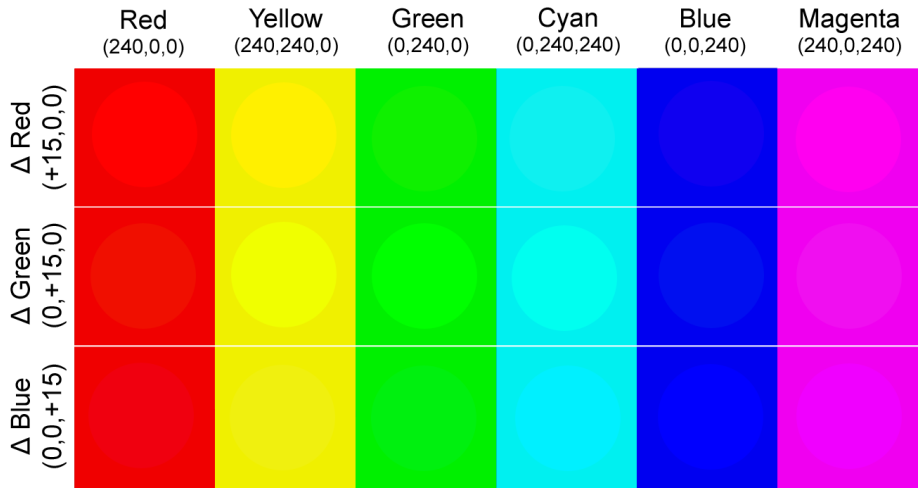


Figure 4.9: *Perception of slight color variations with respect to differently saturated colors. Each square has a circular region whose color was altered by a certain value.*

The itemized observations result from perception. However, it is well-known that the RGB color space is not qualified for analytically modeling aspects which are in relation to human perception. Primarily, this is due to the fact that the RGB color space is non-uniform (see Sect. 4.3.1). Hence, the observations figured out can only be implemented by exploiting a different color space model, such as CIE L*a*b*.

Combining RGB and CIE L*a*b* color model

Watermark data is embedded in colors represented as RGB components. But to implement aspects which are based on human perception, a different color model—in this approach the CIE L*a*b* color model—must be employed. Hence, both RGB and CIE L*a*b* color model must be combined in some way.

In this regard, the basic goal can be formulated as follows: Given a color C from the original image, a new color $\widehat{C}$ is searched which is perceived similar to C and which is capable of storing a considerable amount of data. For this purpose, two demands—with respect to color space—should be met:

1. In RGB color space, for each color channel, the distance between C and $\widehat{C}$ should be maximum. This allows for embedding the largest possible amount of data because the distance defines the range within which the value of a color channel may vary during watermark insertion.

2. In CIE L*a*b* color space, the distance between C and $\widehat{C}$ should not exceed a certain predefined value that defines the extent to which differences in perception are acceptable.

The new color $\widehat{C}$ can be found by firstly determining all colors which have certain distances Δ_{RGB} to C in RGB color space. These colors are then converted into the CIE L*a*b* space to verify that they are still perceived similar to the original color.

During the decoding process, the original color C is unknown since some of its bits may have been replaced by watermark bits. Nevertheless, there are unchanged bits, at least the first four (most significant bits). Thus, solely those four bits can be included when $\widehat{C}$ is searched. Consequently, 4096 color classes exist, each with 4096 elements sharing the four most significant bits of their color channels.

Searching $\widehat{C}$ proceeds as follows: A color C is assigned to its class that contains all colors differing in their channels' four least significant bits (see Fig. 4.10(a)). Next, to ascertain similarity in perception, each of these colors is compared with each other in the CIE L*a*b* color space.

Figure 4.10(b) illustrates color selection with the red color channel. Its four most significant bits, referred to as K, are the same as those of all other class colors' red channels. The remaining bits, referred to as b_i and v_j, are those of interest. Two colors, with R_l and R_m being their red channels, are compared by varying b_i to emulate potential red channel values of C, and by varying v_j to emulate potential bit replacements because of watermarking. Considering that C is unknown, the purpose of comparing all feasible color variations

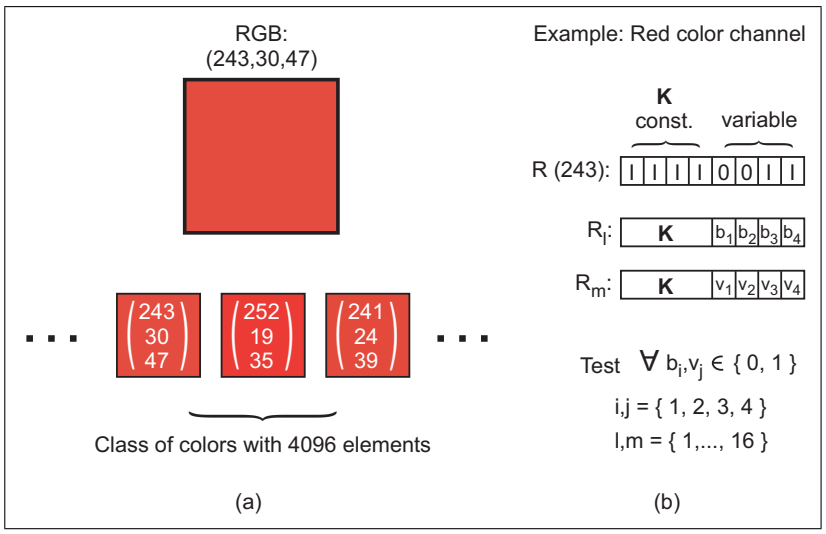


Figure 4.10: *(a) A particular pixel's color is classified according to the four most significant bits of its color channels. (b) The red color channel serves as example for illustrating which bits will remain unchanged and which bits will be varied to determine perceivable similarities between two colors.*

is to determine the worst perception scenario (within the predefined limits), which yields each channel's capacity in the end.

As to comparison in CIE L*a*b* color space, testing positive for all bit elements of one of the following four groups results in the according capacity, synonymous with $\widehat{C}$. For example, a capacity of 4 bits ($Cap = 15$) requires that b_1 is set to 0 and v_1 is set to 1. Then, all colors which can be expressed by varying b_2, b_3, b_4 are compared (in CIE L*a*b* color space) with all color which can be expressed by varying v_2, v_3, v_4. If all these comparisons yield values within the predefined limits, the capacity is 4 bits. Note that testing must be done for all color channels instead of solely the red channel, which is done here for illustration.

1. ***Capacity of 4 bits.*** $b_1 = 0$, $v_1 = 1$. The remaining values $b_2, b_3, b_4, v_2, v_3, v_4$ are combined in all variations.

2. ***Capacity of 3 bits.*** If previous test fails. $b_2 = 0$, $v_2 = 1$, $b_1 = v_1$. The values $b_1, b_3, b_4, v_1, v_3, v_4$ are combined in all permitted variations.

3. ***Capacity of 2 bits.*** If previous tests fail. $b_3 = 0$, $v_3 = 1$, $b_1 = v_1$, $b_2 = v_2$. The values $b_1, b_2, b_4, v_1, v_2, v_4$ are combined in all permitted variations.

4. ***Capacity of 1 bit.*** If previous tests fail

In the worst case scenario, about 16.7 million conversions into CIE L*a*b* space and comparisons are necessary. However, the color classes will never change and hence, their capacities are constant values. Thus, capacities need to be determined only once and can then be adequately stored (e. g., as look-up table).

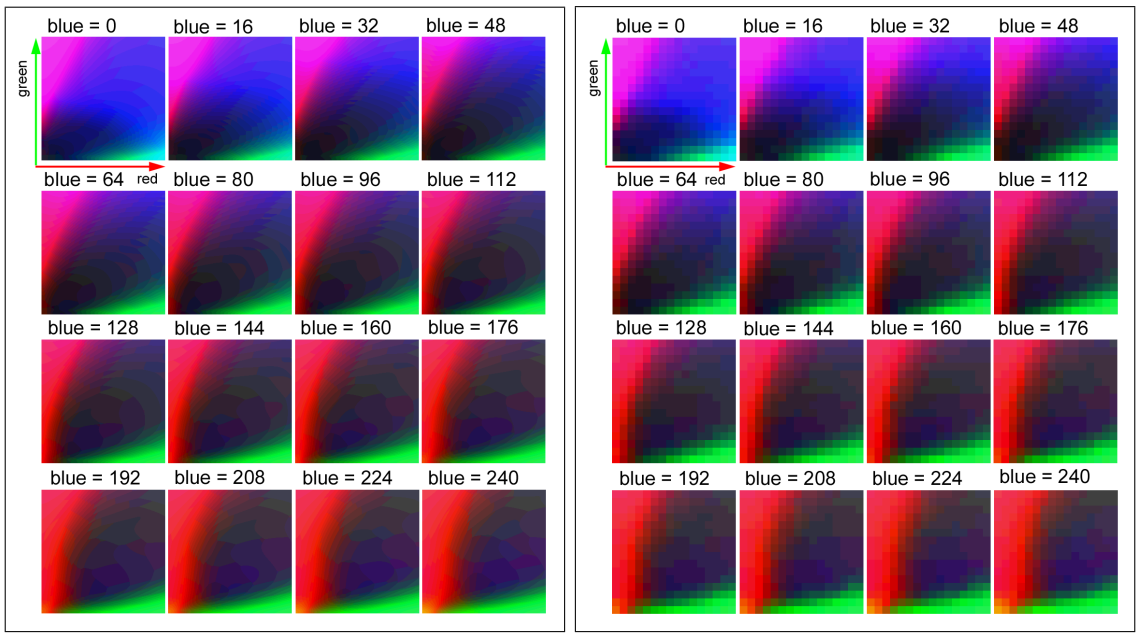


Figure 4.11: *Left: Colors indicate data storage capacities depending on color channel saturations (red and green according to arrow colors, blue according to captions). Green color, for example, signalizes that most bits can be included in the green color channel. Right: Colors were classified according to their channels' four most significant bits.*

Figure 4.11(a) shows capacities depending on color channel intensities in RGB color space. The blue color channel was incremented by a value of 16, whereas for both red and green the whole range of 256 values was included in capacity determination. Capacities are expressed in terms of colors. Saturated red color, for example, indicates that most bits of the original pixel's red color channel can be replaced for watermarking, whereas the bits of the blue and green color channels should remain almost unchanged.

What can be learned from the individual graphics is that with an increasing color channel intensity this particular channel's capacity decreases. A channel intensity is denoted by the green and red arrow, and the caption in case of blue. The graphics further show that there are smooth transitions between adjacent color (capacity) values. Figure 4.11(b) shows the same scenario, but instead of including every green and red color tone, capacities for color classes are depicted. This scenario corresponds with the 4096 color classes addressed in the previous descriptions.

Capacity Map generation

The previous paragraphs described two techniques with which capacity values for a selection of pixels in case of entropy analysis and for individual pixels in case of color intensity analysis can be determined. Those capacity values, which serve as guiding values, are computed for each color channel. They can range from 0 to 15, which corresponds to a maximum codable data amount of 4 bits per channel. Capacity values are stored in a

Capacity Map which is a 2D raster graphic of the same size as the cover image whose pixel values represent the capacity values.

To generate the *Capacity Map*, the differing capacity values which result from applying the two techniques need to be adequately combined. The goal in this regard is to determine the highest possible amount of data which can be encoded. Algorithm 3 shows how the two techniques were integrated. At first, the entropy of a pixel block is computed, which yields a first capacity suggestion. Thereafter, for each pixel of the block, a second capacity value is determined which results from color intensity analysis. The capacity values recommended by both techniques are then compared to determine the pixel's final suggested capacity.

Algorithm 3 *Determining the capacity Cap_p of a pixel p by combining those capacities suggested by entropy and color intensity analysis.*

for all pixel blocks of size 8×8 **do**
  for each color channel, compute block's entropy and store results in $cap_{entropy}[k]$ with $k = \{red, green, blue\}$
  for all pixels p of the block **do**
    for each color channel, determine pixel's capacity according to perceptual aspects and store results in $cap_{perception}[k]$
    for all color channels k of pixel p **do**
      if $cap_{entropy}[k] > 2 \parallel cap_{perception}[k] = 15$ **then**
        $Cap_{p,k} \Leftarrow 15$
      else if $cap_{perception}[k] \geq 7$ **then**
        $Cap_{p,k} \Leftarrow cap_{perception}[k]$
      else if $cap_{entropy}[k] > 1$ **then**
        $Cap_{p,k} \Leftarrow 7$
      else
        $Cap_{p,k} \Leftarrow cap_{perception}[k]$
      end if
    end for
  end for
end for

Figure 4.12 includes three images: the original, the watermarked, and an image that represents the *Capacity Map* with normalized color values as capacities. In the capacity image, dark regions indicate low storage capacities whereas bright regions allow for embedding high bit rates.

Watermark retrieval

The described technique is a blind method. To retrieve the encoded data, the *Capacity Map* must be computed from the watermarked image. Since only the four most significant bits of a color channel, which were not modified during the watermarking procedure, were included in generating the *Capacity Map*, it is ensured that the *Capacity Map* computed

Figure 4.12: *The original image (left) has been watermarked (center) according to the determined* Capacity Map *(right). On average, 7.8 bits were encoded per pixel; in terms of each individual color channel, 2.7 bits (red), 2.66 bits (green), and 2.5 bits (blue) were embedded.*

from the watermarked image is identical to the *Capacity Map* computed from the original cover image.

However, the *Capacity Map* served only as guideline to suggest potential data encoding rates. The number of bits eventually inserted into the color channels of a pixel was determined depending on the size of the watermark stream to be embedded. Hence, besides those pixels used as region identifiers (compare with Sect. 4.2.3), four other pixels located at the beginning of a region were exploited to store the size of the watermark stream using a predefined watermarking scheme. Analyzing those reserved pixels enables to detect a watermarked region as well as to decode the amount of encoded data, which together with the *Capacity Map* suffices to decode the watermark stream contained in that particular region.

4.4 Wavelet Transform Based Approach

In this section, a transformed domain approach is discussed which is based on the *Discrete Wavelet Transform (DWT)*, a technique widely-used in recent watermarking approaches. Many authors state that watermarks embedded using the DWT are more robust compared to those embedded using techniques such as the *Discrete Cosine Transform (DCT)* (e. g., [GE04, MU01]).

When the wavelet transform is applied to a signal (e. g., a 2D image), the signal is split up into different time-scale representations (signal's frequency content local in time). These representations can then be subject to further studies. Xia et al. [XBA98], for example, state that edges in an image are typically well-confined to the high frequency subbands which result from DWT appliance. Large coefficients in those subbands, which indicate edges, can thus be exploited for watermark insertion.

The wavelet transform $T^{wav}_{b,a}$ of a signal f results from the inner products $\langle f, \psi_{b,a} \rangle$, with $\psi_{b,a}$ being translated and dilated versions of the *mother wavelet* ψ. The parameters b and a are

termed as *translation* and *dilation* parameters, respectively. In case of the discrete wavelet transform, the values of a and b are restricted to discrete values with $a \neq 0$. The wavelet transform is specified in Equation (4.2). Further readings concerning formula derivation, wavelet properties, etc. can be found, for example, in [Dau95, GC99, Kai94].

$$T_{b,a}^{wav}(f) \quad = \quad |a|^{-1/2} \int f(t)\, \psi \left(\frac{t-b}{a} \right) dt \tag{4.2}$$

A widely-used wavelet function is the *Haar* function, which was also employed for the watermarking approach proposed in this section. In the following, the decomposition of an image and its reconstruction is illustrated using the Haar wavelet.

4.4.1 Image Decomposition using Haar Wavelets

The basic idea behind the wavelet transformation is to approximate an arbitrary function by linear combinations of wavelets, in this section Haar wavelets. The Haar wavelet is defined by the following function:

$$\psi_H(x) \quad = \quad \begin{cases} 1 & : \quad 0 \leq x < 1/2 \\ -1 & : \quad 1/2 \leq x < 1 \\ 0 & : \quad Otherwise \end{cases} \tag{4.3}$$

Parameters for a and b (see Eq. 4.2) are chosen such that $a = 2^m$ and $b = 2^m n$ (see [Dau95]). Hence, a scaled and translated Haar wavelet can be described as:

$$\psi_{m,n}(x) \quad = \quad \psi_H \left(2^{-m} x - n \right) \tag{4.4}$$

When it is assumed that a given function f is piecewise constant on intervals between $l\, 2^{-J_0}$ and $(l+1)\, 2^{-J_0}$, the constant values f_l^0 represent f on these intervals (see Fig. 4.13). Each f_l^0 can be expressed as a sum of f_k^1 and δ_k^1 or as a difference between them, alternately. f_k^1 is an approximation to f_{2k}^0 and f_{2k+1}^0, and δ_k^1 is the difference between f and the function's approximation:

$$f_k^1 \quad = \quad \frac{1}{2} \left(f_{2k}^0 + f_{2k+1}^0 \right) \tag{4.5}$$

$$\delta_k^1 \quad = \quad f_{2k}^0 - f_k^1 = \frac{1}{2} \left(f_{2k}^0 - f_{2k+1}^0 \right) \tag{4.6}$$

To compensate the difference between f and f_k^1 using Haar wavelets, the width of the mother wavelet must be adapted so that its new width is twice as large as 2^{-J_0} ($m = -J_0 + 1$, see Eq. 4.4). Furthermore, the wavelet has to be moved to the position that corresponds to the approximated segment of the function. Thereafter, the Haar wavelets can be scaled by linear combinations with δ_k^1 (see Fig. 4.13). The function f can thus be represented as the sum of f_k^1 and the scaled and translated Haar wavelets.

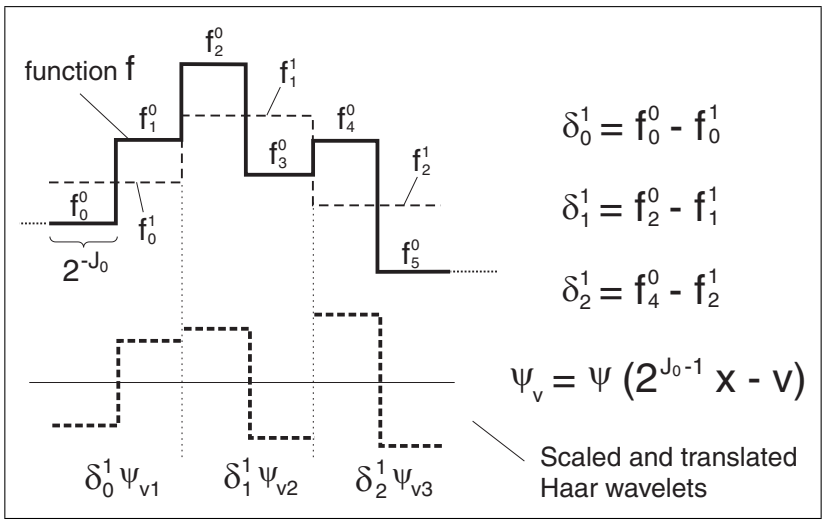


Figure 4.13: *A part of a given function f is represented by its piecewise approximations f_k^1 and adapted Haar wavelets ($\delta_k^1 \psi_v$) (according to [Dau95]).*

Image decomposition

The 2D wavelet transform can be expressed as a number of 1D transformations afore addressed [Par97]. Hence, when the input signal is an image represented as a 2D matrix of pixel values, the 1D wavelet transform can first be applied to each row, which yields a transformed matrix. Thereafter, the wavelet transform can be applied to each column of the transformed matrix. Similar to the wavelet transform in 1D, the result is the input image split up into two equal parts, typically low frequencies (the approximations) and high frequencies (the details). Since the low frequency signal can still contain details in which a user is interested, it can be subject to further decompositions. The number of decompositions applied to a signal is referred to as *decomposition levels*.

In Figure 4.14, a 1-level wavelet decomposition is applied to an image, more precisely, to four pixels (p_i^0) representing RGB color channel values. The second scheme in the figure illustrates applying the wavelet transform to the image's rows, which results in low-frequency (p_k^1) and high-frequency (d_k^1) subbands. Finally, the third scheme illustrates the computation of the resulting 1-level decomposed image. The reverse transformation of the decomposed image is illustrated in Figure 4.15.

Figure 4.16 shows the result of applying a 2-level wavelet decomposition to an image. Here, the low-frequency subband was decomposed a second time. Since edges and regions with varying details can easily be detected in the middle- and high-frequency subbands (HL, LH, HH), these subbands are typically considered when watermarks are to be embedded using a DWT-based approach. To this end, a selection of the values in each subband (called *wavelet coefficients*) are modified.

In the following, a watermarking technique is discussed that modifies a selection of coeffi-

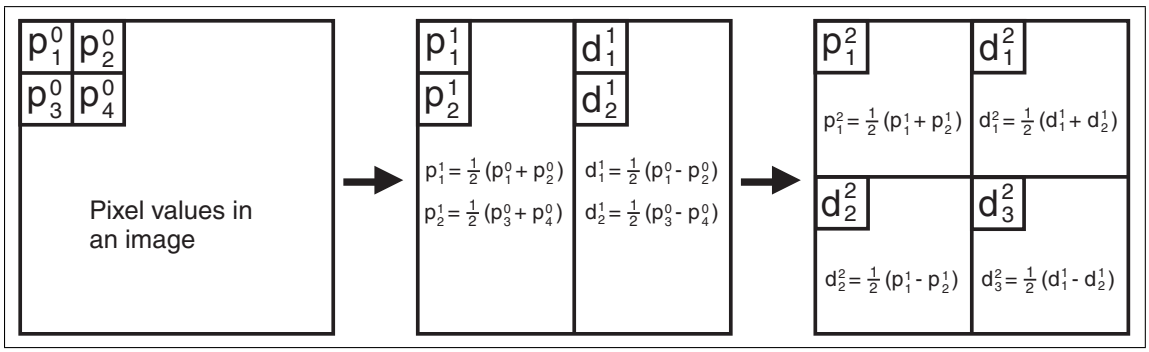


Figure 4.14: *Four pixel values of an image $(p_1^0, p_2^0, p_3^0, p_4^0)$ are transformed line by line (second scheme) and column by column (third scheme) to yield the image part's decomposition.*

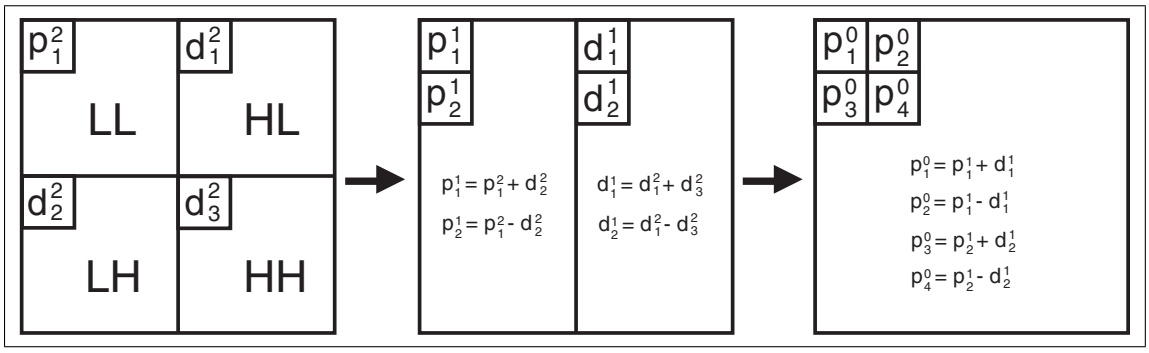


Figure 4.15: *Correlating coefficients of a decomposed image $(p_1^2, d_1^2, d_2^2, d_3^2)$ are inverse transformed to reconstruct the corresponding part of the original image.*

cients in the low- and high-frequency subbands (LH_1, HL_1, and HH_1 in Fig. 4.16) which result from applying a 1-level wavelet decomposition. The coefficients are selected according to previously specified regions which are those regions associated with metadata, and according to coefficient magnitudes. The proposed technique is similar to the technique discussed by Su et al. [SWK99]. They also embed watermarks by modifying wavelet coefficients with respect to afore selected regions. However, since the authors aim at security aspects, they try to increase the robustness of the encoded watermark by altering solely those coefficients which have specific characteristics (e. g., whose magnitudes are higher than certain thresholds), which restricts them in terms of capacity.

4.4.2 Region-Based Watermarking Technique

Applying the wavelet transform to a signal f results in $T(f)$ which is a transformed version of the signal. The original signal can then be reconstructed by applying the inverse wavelet transform (see Fig. 4.15). However, when the transformed signal is modified during the watermarking, which yields $\widehat{T(f)}$, a new signal $\widehat{f}$ can result upon inverse transform appliance. In terms of image watermarking, the goal is hence to minimize the difference between f

Figure 4.16: *The original image (left) is wavelet transformed using a 2-level decomposition (center). The transformation results in low-, middle-, and high-frequency subbands (LL, HL, LH, and HH).*

(cover image) and $\widehat{f}$ (watermarked image), while, at the same time, a considerable amount of data is encoded.

Typically, a subset of the series of wavelet coefficients is considered for watermark insertion. High-frequency coefficients, for example, provide information about prominent features in an image (e. g., edges), which makes them particularly applicable for watermarking. But in case of *Illustration Watermarking*, a fixed amount of data is to be inserted, limited to certain image regions. Selecting solely the most significant coefficients may hence not suffice to embed the whole data amount, which requires to include other coefficients, as well. Also, due to the fact that watermarks are embedded only into certain regions which were selected according to content instead of determinable features, it is necessary to mark those regions. In other words, it must be possible to determine for each pixel whether it stores a portion of the watermark stream.

Another important aspect to be considered is that the valid range of RGB color channel values is between 0 and 255. Transforming the signal f into a wavelet representation, modifying a selection of coefficients, and applying the inverse transform can yield invalid RGB color values, which is to be prevented.

Algorithm description

After the image has been decomposed, watermark insertion proceeds in three steps:

1. For each subband to be coded, a *Capacity Map* (comparable to the *Capacity Map* introduced in Sect. 4.3.3) is generated which indicates how many bits of a coefficient can be replaced by watermark stream bits. In these maps, all positions of coefficients C_{min} which correspond to the region to be watermarked are marked. In addition, among those coefficients, all coefficients C_{max} with a magnitude higher than

Figure 4.17: *Left: The original image has a certain region which is associated with data to be coded. This region is highlighted in each subband image. Right:* Capacity Maps *are overlayed on their corresponding subband images. Dark values indicate lower codable data rates than brighter values within the region.*

a specified threshold are specifically marked. These latter coefficients are those which indicate abrupt changes (edges) in the image. The threshold has to be carefully chosen so that a coefficient does not change its state after watermarking (e. g., from C_{min} to C_{max}). This is done by setting the threshold to the maximum value which can be expressed by those bits to be coded. For example, in case of a maximum of 4 bits to be encoded into a coefficient, the threshold would be 15.

Figure 4.17(right) shows the *Capacity Maps* overlayed on their corresponding subband images. Capacity values are provided for each individual coefficient. The brighter a value in the *Capacity Map* which is here confined to the region to be watermarked, the more bits can be embedded.

2. For data insertion, the absolute values of coefficients that are marked in the *Capacity Map* are modified. Embedding data into those coefficients is then equal to embedding data into RGB color channel values by replacing their bits with watermark bits. During the data insertion, coefficients marked as C_{max} are modified stronger than those marked as C_{min}, which yields variations between the amounts of encoded data.

   In Figure 4.18, the encoding of the character *'a'* is illustrated using a constant capacity level of 2 bits per coefficient. Assuming that the coefficients c_k^r, c_k^g, and c_k^b correspond to the red, green, and blue color channels, respectively, the two LSBs of c_k^r and c_k^g, and the second bit of c_k^b are replaced by character bits. Note that the LSB of a coefficient that corresponds to the blue color channel is reserved for region identification so that it cannot store watermark data.

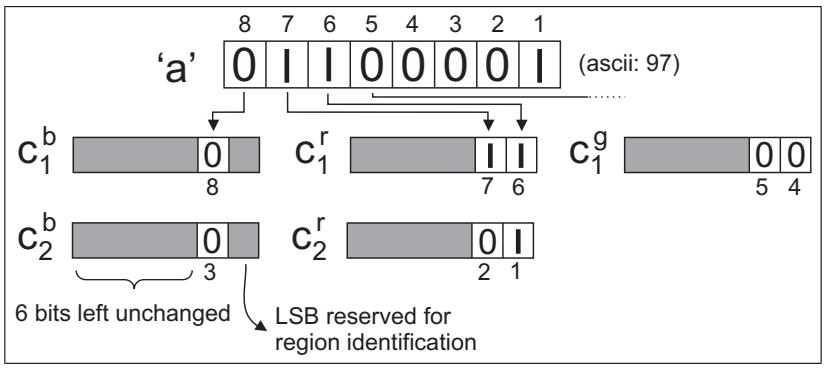


Figure 4.18: *Example of how the character 'a' can be encoded into five coefficients.*

3. After watermarking a particular coefficient at a certain position in each of the three subbands, the inverse wavelet transform is emulated to verify that the modified RGB color channel is still valid (i. e., between 0 and 255). If it exceeds its limits, the according original color channel value has to be adapted, i. e., it is increased or reduced accordingly. Thereafter, the corresponding coefficients and *Capacity Map* values have to be recalculated and watermarking is repeated.

Watermark retrieval

Just as the other techniques described in the previous sections, the wavelet domain technique is a blind technique. After decomposing the watermarked image, the coefficients that represent the four subband images are equal to those coefficients the watermarking procedure yielded. The subsequent step is to generate the *Capacity Map*. Even though the watermarked coefficients differ from the original (unmodified) coefficients which were used to generate the *Capacity Map* during data insertion, both computed *Capacity Maps* are equal. The reason is no coefficient changed its state (C_{min} or C_{max}) during data insertion since the threshold was appropriately chosen. Once the *Capacity Map* has been generated, watermarked regions can be identified and the encoded data can be extracted.

4.4.3 Results and Discussion

Results

Figure 4.19 shows two example images and the associated ID-buffer which is used for both images. The first image (*Trees*, Fig. 4.19(left)) was chosen because it contains two regions that considerably vary in their entropies. The marked region in the ID-buffer covers those regions to similar parts. The second image (*Color-Pattern*, Fig. 4.19(center)), which is computer-generated, consists of four uniformly colored regions.

The results of encoding different data rates into the first example image (*Trees*) are illustrated in Figures 4.20 and 4.21. The first image in each figure shows the decomposed image with the corresponding *Capacity Maps* overlayed. The second image is the watermarked image for which data encoding was confined to a circular region marked in the ID-buffer.

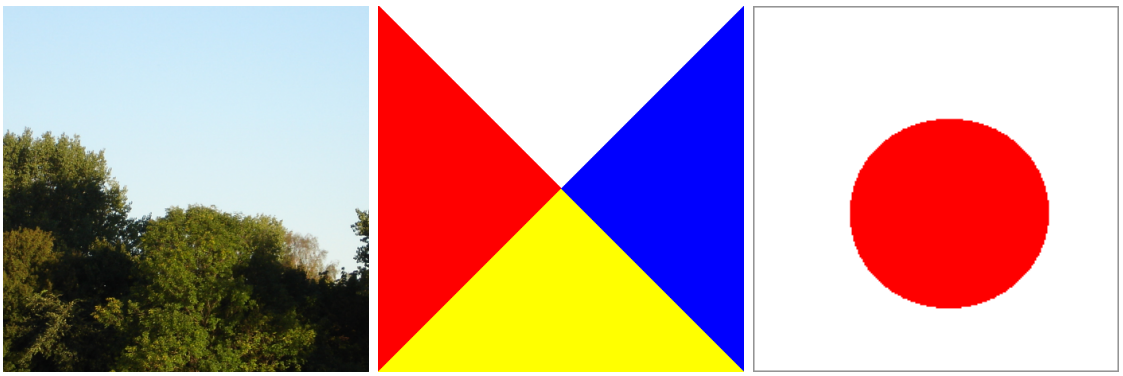


Figure 4.19: *Example images (left and center) and their associated ID-buffer (right).*

Depending on the magnitude of a coefficient, two different bit rates were encoded. The first bit rate $Bits_{min}$ was encoded into all coefficients inside the marked region whose magnitudes were below the specified threshold. The second bit rate $Bits_{max}$ was encoded into the remaining coefficients of a region. The third image in each figure is a scaled image that illustrates the difference between the original and the watermarked image. Due to the high bit rates chosen in the watermarked image in Figure 4.21 ($Bits_{min} = 3$ and $Bits_{max} = 4$ yield an average bit rate of 6.83 bits/pixel), the coded region is clearly perceivable.

Figure 4.20: *Left: Decomposed image with* Capacity Maps *overlayed. Center: Watermarked image.* $Bits_{min} = 1$ *and* $Bits_{max} = 3$ *yield an average encoded bit rate of* 2.77 *bits/pixel (total number of encoded bits: 130209). Right: Scaled difference between original and watermarked image (the darker a pixel value, the higher the difference).*

In Figures 4.22 and 4.23, the watermark stream was embedded into the second example image (*Color-Pattern*). Because of its low entropy, modifications introduced during watermarking can be detected faster than in images with high entropies. Consequently, already low encoded bit rates are perceivable.

Numerical differences between the original and watermarked images are summarized in Table 4.1. It can be seen that the changes introduced to the *Color-Pattern* image influence the image texture far more than the equivalent modifications introduced to the *Trees* image.

Figure 4.21: *Left: Decomposed image and* Capacity Maps. *Center: Watermarked image.* $Bits_{min} = 3$ *and* $Bits_{max} = 4$ *yield an average encoded bit rate of* 6.83 *bits/pixel (total number of encoded bits: 320448). Right: Difference image.*

Figure 4.22: *Left: Decomposed image and* Capacity Maps. *Center: Watermarked image.* $Bits_{min} = 1$ *and* $Bits_{max} = 3$ *yield an average encoded bit rate of* 2.29 *bits/pixel (total number of encoded bits: 107697). Right: Difference image.*

Several $PSNR$ values of less than 40 indicate that the capacity limit was reached, which is confirmed by the image in Figure 4.23. Setting $Bits_{min}$ to 2 in this particular case results in the $PSNR$ value of 37.45 and clearly perceivable artifacts.

Drawback of this technique

Figure 4.24(a) shows a 2x2 pixels subregion of the cover image and its four values f_k^0 which represent an arbitrary color channel. Applying the wavelet transform yields the subregion's coefficients (b) whose inverse transformation results in those values depicted in diagram (c). When the coefficients are left unchanged, the inverse transformation results in the original cover image: $f_k^0 = \widehat{f_k^0}$. Assuming, on the other hand, the coefficients were modified, the inverse transformed image is most likely different from the original image.

In contrast to watermarking techniques that operate in the spatial domain, applying a wavelet domain technique can affect more bits in the cover image than those which were actually modified in the transformed representation. The reason is changes can sum up

Figure 4.23: *Left:* Decomposed image and Capacity Maps. *Center: Watermarked image.* $Bits_{min} = 2$ *and* $Bits_{max} = 3$ *yield an average encoded bit rate of* 4.52 *bits/pixel (total number of encoded bits: 212271). Right: Difference image.*

Image	**#bits/pixel** ($Bits_{min}/Bits_{max}$)	**MaxDiff**	**MSE**	**PSNR**
Color-Pattern	2.29 (1 / 3)	30	2.37	44.39
	2.32 (1 / 4)	56	5.03	41.12
	4.52 (2 / 3)	30	11.69	37.45
	4.54 (2 / 4)	56	14.44	36.54
	6.77 (3 / 4)	54	58.69	30.45
Trees	2.77 (1 / 3)	23	1.25	47.17
	2.49 (1 / 4)	39	1.43	46.57
	4.76 (2 / 3)	15	2.07	44.98
	4.66 (2 / 4)	34	2.14	44.82
	6.83 (3 / 4)	43	8.99	38.59

Table 4.1: *Deviations of the watermarked images from their originals (*MaxDiff*: maximum difference between color channel values of the original and watermarked image,* MSE*: Mean Squared Error,* PSNR*: Peak Signal-to-Noise Ratio).*

during the inverse transformation, as the following example illustrates.
The values $\widehat{f_1^0}$ and $\widehat{f_4^0}$ are determined as follows:

$$\widehat{f_1^0} = (c_1 + c_3) + (c_2 + c_4) \qquad \text{and} \qquad \widehat{f_4^0} = (c_1 - c_3) - (c_2 - c_4) \tag{4.7}$$

When each middle- and high-frequency coefficient is modified by one of the positive values w_2, w_3, w_4, the maximum differences between f_1^0 and $\widehat{f_1^0}$ and between f_4^0 and $\widehat{f_4^0}$ are:

$$\widehat{f_1^0} \quad = \quad (c_1 + (c_3 + w_3)) + ((c_2 + w_2) + (c_4 + w_4))$$

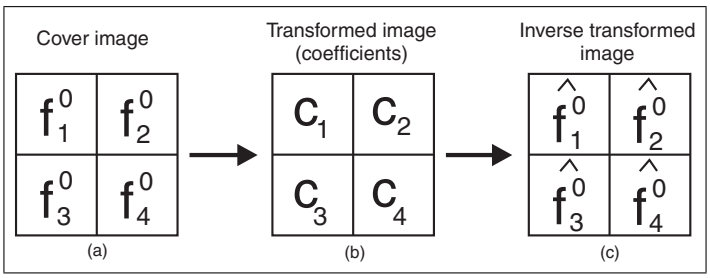


Figure 4.24: *Four color channel values of an image subregion (a) are transformed, which yields the coefficients in (b). These are then inverse transformed (c).*

$$\begin{aligned} &= (c_1 + c_2 + c_3 + c_4) + (w_2 + w_3 + w_4) \\ &= f_1^0 + (w_2 + w_3 + w_4) \end{aligned} \tag{4.8}$$

$$\begin{aligned} \widehat{f_4^0} &= (c_1 - (c_3 + w_3)) - ((c_2 + w_2) - (c_4 - w_4)) \\ &= (c_1 - c_2 - c_3 + c_4) - (w_2 + w_3 + w_4) \\ &= f_4^0 - (w_2 + w_3 + w_4) \end{aligned} \tag{4.9}$$

When a spatial domain technique is used, the bits of a RGB color channel are directly affected. Consequently, the maximum change of a color value correlates with the number of bits modified. In contrast, encoding the same amount of data into a coefficient using a wavelet domain technique can affect the original color value by the data rate encoded into that coefficient plus the data rates encoded into the two remaining coefficients involved, as Equations 4.8 and 4.9 show. Hence, encoded data rate and introduced color change do not correlate, which can have a negative effect on the watermark's transparency.

Discussion

Many proposed techniques embed watermarks into the wavelet transformed representation of the original image. This representation allows to select prominent image features (e. g., edges and regions with noise) which can be modified without introducing perceivable changes. In addition, those techniques are said to be more robust than others.

However, to embed watermarks which are to be robust to image modifications, solely those coefficients can be altered which were selected according to specific characteristics (e. g., high magnitude). Also, less significant bits cannot be modified since these are the bits soonest affected when image operations are applied which can result in watermark data loss. Both the selection of solely specific coefficients and the manner in which these coefficients can be modified limits the amount of data which can be embedded. In case of content-based approaches such as *Illustration Watermarking*, the number of encodable bits decreases even more because coefficient selection is limited to certain regions. These restrictions and the drawback addressed in the previous paragraph are incompatible with the requirements demanded by *Illustration Watermark* insertion, in spite of a potential increase in robustness.

4.5 Summary

This chapter introduced several techniques for embedding *Illustration Watermarks* in 2D raster graphics. It started with a standard technique that simply replaces the least significant bits of a color channel value with bits of the watermark stream. This technique does not consider image texture characteristics, which results in a constant number of bits inserted in each pixel. However, since the inclusion of those characteristics during the watermarking can improve the transparency of the introduced changes, techniques were developed which make use of a color space which conforms better with human perception and which analyze the image texture prior to data insertion. Finally, a watermarking technique based on the Wavelet transform, which is widely used in the traditional watermarking domain and which primarily aims at security aspects, was adapted so that it can be exploited for *Illustration Watermarking*.

5 Evaluation of Illustration Watermarking Techniques for Raster Graphics

Illustration Watermarking aims at imperceivably embedding descriptive metadata into a cover medium so that it can be explored by an end-user. The efficiency of a watermarking technique in terms of whether changes introduced to a cover medium because of data insertion can thus only be assessed by end-users instead of software (*experimental* vs. *analytical method* [AD86]).

To evaluate a selection of the techniques proposed in the previous Chapter 4, a user study was conducted—in collaboration with Andrea Unger and Thomas Vogel (see [SUS+06])—which should primarily give information about:

- the number of bits per pixel which can be embedded without introducing perceivable changes,
- the effects of inserting equal numbers of bits into different types of images, and
- whether there are noticeable differences between the employed watermarking techniques.

Hypotheses

During the study, participants were shown three types of images varying in their information contents (entropy levels). Besides their differing characteristics, the displayed images differed in their amounts of encoded data and the method which was used for data encoding. We assumed that the image with the highest level of entropy would conceal encoded data most. We also assumed that with an increasing amount of encoded data, changes would be detected easiest. The third assumption we made was that the technique described in Section 4.3.3 would conceal encoded data most effectively.

This chapter starts with a description of the study in Section 5.1. Thereafter, Section 5.2 presents the results of the study which are discussed and summarized in Sections 5.3 and 5.4.

5.1 Description of the Study

This section describes aspects concerning the general setup and design of the study. This includes, in particular, introducing the images which were chosen for evaluation and describing the actual task participants were ask to perform.

5.1.1 Participants and Experimental Setup

112 voluntary participants (36 female, 76 male), aged between 19 and 52 years, took part in the study. The study was conducted in a research lab at the local computer science department.

During the study, a standard PC (*MS Windows XP*) with a high-resolution flat panel monitor (*IBM T221*: 22.2 inch viewable diagonal image size, 3840 x 2400 maximum resolution, 204 ppi) was employed (see Fig. 5.1). A standard computer mouse served for interaction. The software used was implemented in *PureBasic*. It basically asked for personal data, displayed an image for a particular time, and registered and logged mouse events triggered by the participant.

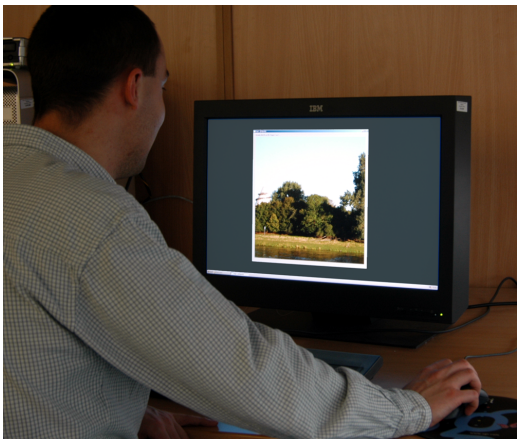


Figure 5.1: *Participant who performs the task.*

Though there are various parameters which may have influence on a participant's viewing conditions, it was tried to minimize parameter disparities between participants. For example, each participant performed the task at the same workstation with the same monitor. Also, the room was darkened to avoid differing lighting conditions and reflections on the display.

5.1.2 Watermarking Techniques

Three watermarking techniques discussed in the previous chapter were chosen for evaluation:

- ***General LSB technique (LSB).*** This technique was introduced in Section 4.2.3. It basically replaces up to the four least significant bits of a color channel value with watermark stream bits. The sequence in which bits are included complies with human sensitivity to color bands (see Sect. 4.3.1). Cover image features such as its texture are not involved during watermarking.
- ***Adapted LSB technique (LSB_Ext).*** The adapted LSB technique was discussed in Section 4.3.3. It first computes a *Capacity Map* by analyzing the entropies of pixel

blocks and by comparing colors in the RGB and CIELAB color space. The values of the *Capacity Map* then indicate how many bits can be imperceivably inserted into a certain pixel's color channels.

- ***Wavelet transform based technique (Wave).*** This technique was introduced in Section 4.4.2. It embeds watermark data into wavelet coefficients which result from applying the wavelet transform to the cover image. The amount of data inserted into a coefficient depends on the coefficient's magnitude.

The *LSB* technique was included because it can be used to embed considerable data amounts, which is an important feature with respect to *Illustration Watermarking*. However, since it does not consider aspects of human perception, improving this technique may yield better results. The adapted technique *LSB_Ext* incorporates human perception characteristics during watermarking. It is therefore expected that the *LSB_Ext* technique will outperform the basic *LSB* technique.

The *Wave* technique serves as example of those techniques that insert watermark data into the transformed representation of a cover image. In doing so, these techniques mainly aim at preventing watermarks from being damaged when image operations are applied. High capacity is, in contrast to what *Illustration Watermarking* aims at, only a secondary aspect. How well the *Wave* technique will perform in comparison with the two other techniques will be clarified by the study.

5.1.3 Selection of Images

The three images in Figure 5.2 are the original images whose watermarked copies were presented during the study. These images will be referred to as *Pattern* image, *Landscape* image, and *Flowers* image.

Figure 5.2: *The original uncoded images which were selected for evaluation:* Pattern *image (left),* Landscape *image (middle), and* Flowers *image (right).*

The *Pattern* image is computer-generated and consists of stripes of alternating colors. In contrast to the *Pattern* image, the *Landscape* image is a photograph taken with a digital SLR camera (*Nikon D100*). Parts of its content such as the sky and green plants can be found in many photographs. The *Flowers* image is also a photograph taken with the

digital camera. It features high diversification in information content and includes almost no regions of same colors.

Reasons for this choice

One of the reasons for selecting those images is that a proper evaluation of how many bits can be imperceivably encoded demands for including both computer-generated and real-world images. Klein [Kle93] states in this regard that simple stimuli (in this scenario caused by the computer-generated image) should be used in addition to complex ones (caused by the real-world images). This is due to the fact that complex images are much harder to analyze than images that provide only little information. The author further specifies two advantages of using special (computer-generated) images: (1) vision models can be easier applied to simple stimuli and (2) it is possible to isolate specific image features, which eases the study of a particular technique.

In this regard, the *Pattern* image was designed such that it contains large regions of same colors, which should facilitate the detection of image modifications. The colors were selected with respect to the observations discussed in Section 4.3.3. Regarding their capacities, the colors—white (RGB: 255,255,255) and green (RGB: 0,255,0)—allow for encoding differing amounts of data. Due to its two unsaturated color channels red and blue, considerably more data can be inserted into the green color (10 bits per pixel are suggested by the *Capacity Map*) than into white (6 bits per pixel suggested), which is illustrated in Figure 5.3(a).

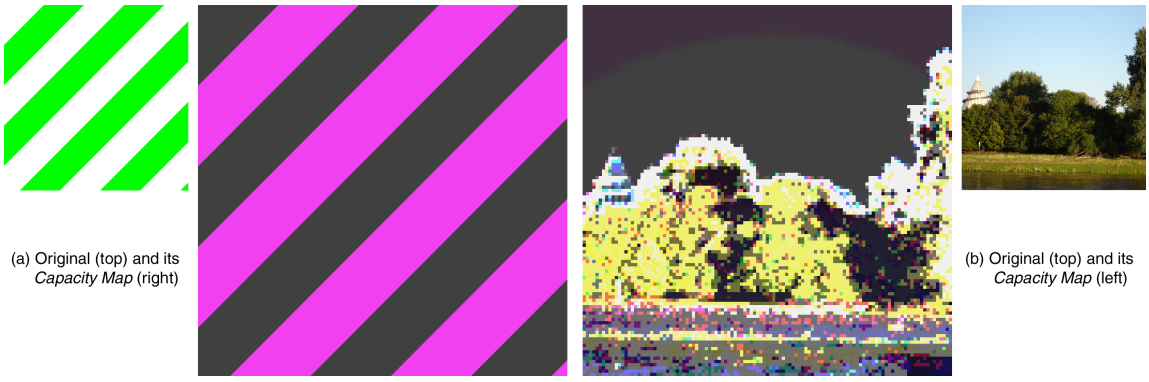
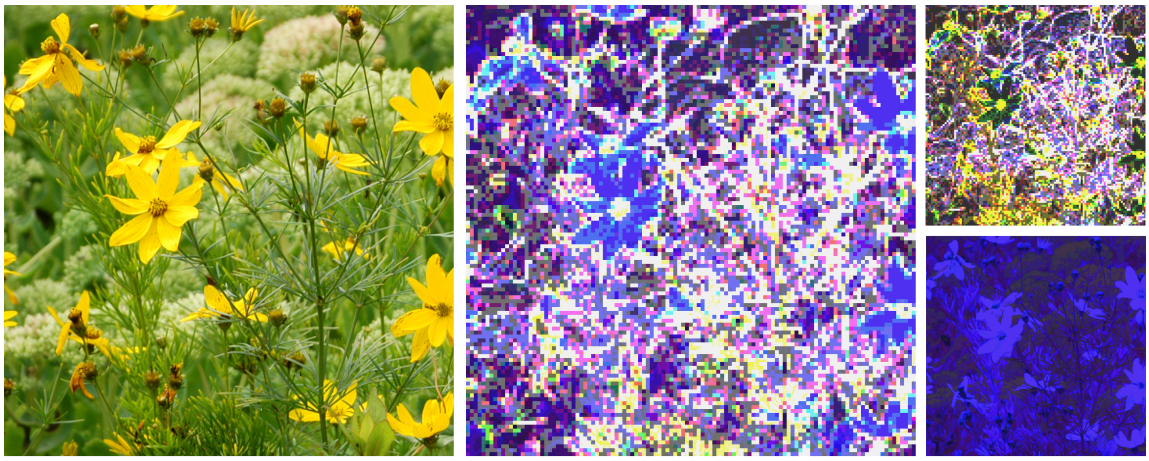


Figure 5.3: *The* Pattern *and the* Landscape *image with their* Capacity Maps.

However, since *Illustration Watermarking* is basically envisioned as a technique that is to be applied to photographs, the *Landscape* image and the *Flowers* image were selected as real-world images. These two photographs differ in their entropies. The *Landscape* image can be divided in two regions: a sky region which has an almost smooth texture and a ground region which is high in contrast. This image is particularly well suited for evaluating the *LSB_Ext* and the *Wave* technique since these techniques consider image characteristics during data encoding provided by their *Capacity Maps*. These maps suggest to encode lower data rates into the sky region than into the ground region (see Fig. 5.3(b)). The

Flowers image will presumably allow to encode the highest amount of data. The reason is its high-contrast texture which can conceal slight modifications introduced during the watermarking. Figure 5.4(b), which shows the *Capacity Map*, confirms this assumption.



(a) Original image

(b) Final *Capacity Map*

(c) *Capacity Map* after entropy (top) and color analysis (bottom)

Figure 5.4: *The* Flowers *image and its* Capacity Map *which results from combining the* Capacity Maps *yielded by entropy and color analysis. Note that color analysis alone would yield lower data capacities, which is indicated by the darker colors in the corresponding* Capacity Map.

5.1.4 Watermark Insertion

A specific amount of data (capacity level) was embedded into each image which was shown to a participant during the evaluation. The data encoded was a random byte stream whose length was adapted according to the data amount to be inserted.

Data encoding was limited to one circular region in each image. These regions were selected randomly in case of the *Flowers* image. For the two other image types, the regions were selected almost randomly, which means the regions were chosen so that they covered a borderline in the image. The borderline divides sky and ground in the *Landscape* image, and green and white in the *Pattern* image. All regions had the same size. Region locations differed between image types as well as between capacity levels. They did not differ between images of the same type which had the same data amount encoded using one of the techniques.

The amount of data which was inserted into a pixel ranged from 3 bits to 12 bits (capacity levels). In case of *Capacity Map* use, varying capacity levels could be suggested for the pixels of a region. The capacity level for a pixel was hence chosen in a way that the averaged capacity level for all pixels approximately corresponded to the level demanded.

The three variables—image type, capacity level, and watermarking technique—yield 90 different images to be evaluated. However, to prevent participants from getting tired of

viewing all these images, the number of images was reduced. Images with clearly visible artifacts caused by watermarking were omitted as well as images with no degradation detectable even though the original image could be employed for comparison and the location of the watermarked region was known. In this way, the number of images to be presented to a participant could be reduced to 54 images. Figures 5.5–5.7 show for each technique those images included in the study which were watermarked with the lowest and highest capacity levels.

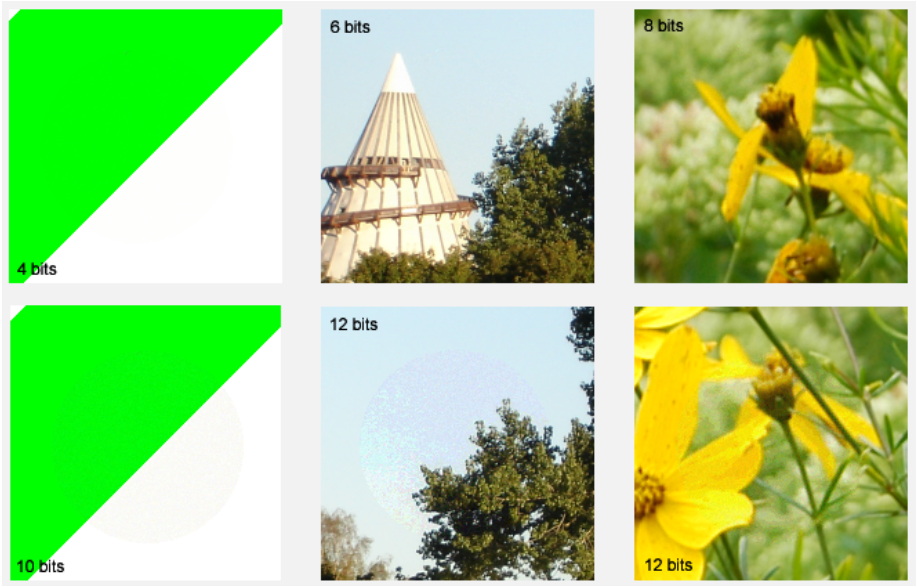


Figure 5.5: *Image parts which were watermarked using the* LSB *technique. The upper row shows evaluated images which have the lowest amount of data encoded, the lower row shows images with the highest data amount encoded. The circular region into which the watermark stream was inserted is located in the center of an image.*

5.1.5 Task Description

At the beginning, participants were asked to answer a pre-study questionnaire including questions about their gender, age, and visual impairments. Thereafter, the task was explained and demonstrated with an example. Each participant was told that the images to be shown would have a circular region which had been modified to a certain degree. Finally, the actual task, the completing of which took about 12 minutes, started.

The basic task was to analyze watermarked images and to find the regions into which a certain amount of data was encoded. To this end, 54 images were successively presented to the participant. Each image was shown for at most 10 seconds. During this time, the participant should find the modified region and select it with the mouse cursor. Once the mouse button was pressed, the next image appeared, regardless of whether or not a region was selected correctly. A region was considered to be hit when the mouse click happened while the cursor position was inside a watermarked region. If no region could be detected

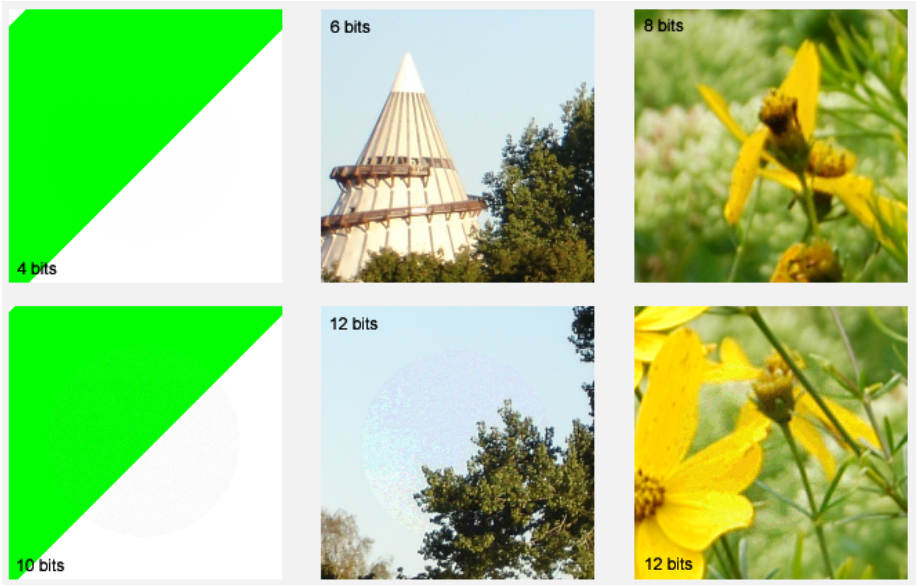


Figure 5.6: *Image parts which were watermarked using the* LSB_Ext *technique.*

Figure 5.7: *Image parts which were watermarked using the* Wave *technique.*

within the 10 seconds, which means no mouse click occurred, the next image appeared automatically.

Reasons for design decisions

During performing the task, the participants were asked to pay their complete attention to analyzing the images. It was hence taken heed that a participant was neither distracted by an incident nor by a person. Also, a participant's concentration was not necessarily constant throughout the task duration. To compensate for this inhomogeneity, the sequence in which images were shown was random and varied between participants.

A former study had shown that setting no time limits caused participants to spend different times on analyzing images presented. Consequently, task completing times differed noticeably between participants, which was no acceptable basis for the study analysis. In this study, the maximum time for which each image was shown was hence limited to 10 seconds. During pre-tests, this time span emerged to be appropriate to detect perceivably modified image regions. However, task completing time was also limited to emulate a more realistic scenario in which a user typically does not spend much time on searching for image artifacts.

Another aspect which the former study revealed to be disadvantageous was that, though the image sequence was random, showing same image types consecutively was not avoided. Differences between those images could thus be detected easily. This aspect was prevented in this study. In addition, a black image was shown for a moment between two images.

5.2 Study Results

During the study, a log file recorded data such as personal data (age, gender, colorblindness), image ID and its capacity level, region detection, and time passed until selection. Analyzing this data revealed information about technique efficiency, image type influence, capacity level limits, etc.

The diagram in Figure 5.8 shows for each image type the total number of correctly selected regions depending on data amounts encoded. The techniques which were employed for watermark insertion are not distinguished in this diagram. Consequently, 336 correct selections (112 participants and 3 techniques) is the maximum which can be achieved for a capacity level. The *Pattern* image is obviously the type of image which concealed embedded data worst, whereas the texture of the *Flowers* image masked introduced changes most effectively. It even caused participants to not detect 118 watermarked regions in case of 12 encoded bits per pixel. These observations correlate with the entropies of the images: the higher an image's entropy, the more data can be embedded ($H_{Pattern} = 1$, $H_{Flowers} = 7.39$). In case of the *Landscape* image, two regions have to be differentiated: the sky region ($H_{Sky} \approx 3.22$) and the ground region ($H_{Ground} \approx 7.23$). Since both regions were involved in each watermark insertion, the *Landscape* image concealed encoded data worse than the *Flowers* image.

The diagrams in Figures 5.9–5.11 allow to draw conclusions from analyzing the effects of watermarking technique, image type, and capacity level on correct region detection. Each diagram shows that the *Wave* technique is associated with most correct selections, which indicates that this technique is most ineffective. The diagrams also show that, in most cases,

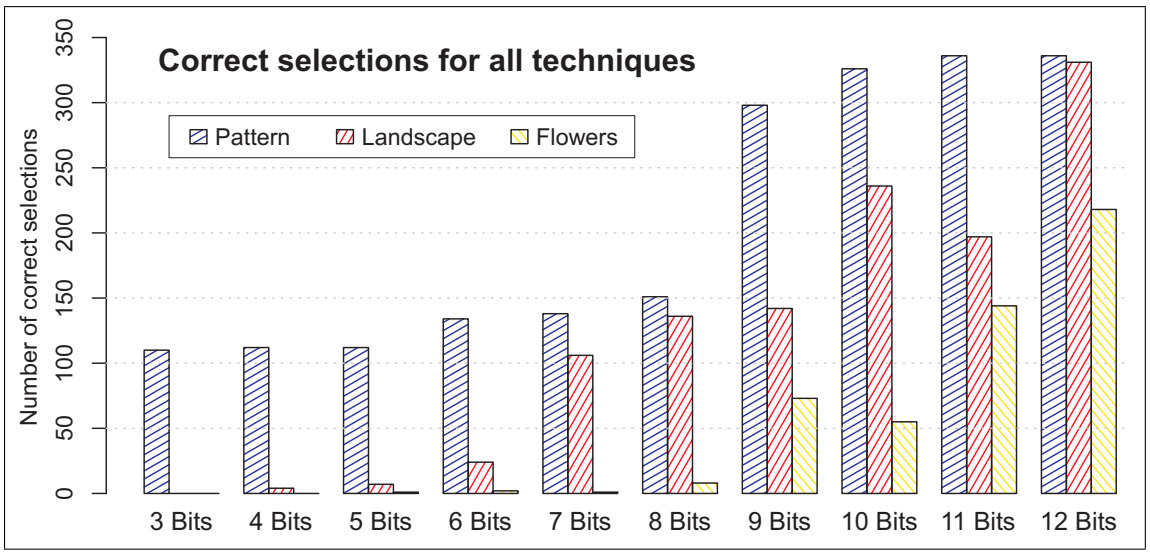


Figure 5.8: *Correctly selected regions for all techniques, separated according to type of image and capacity level.*

the *LSB_Ext* technique performs better than the standard *LSB* technique. Regarding the effects of technique and image type on region detection, *One-Way ANOVA (Analysis of variance)* yields that both variables have a similar influence: $F_{2,6,tech} = 2.453, p = 0.167$ (technique type) and $F_{2,6,img} = 2.655, p = 0.149$ (image type).

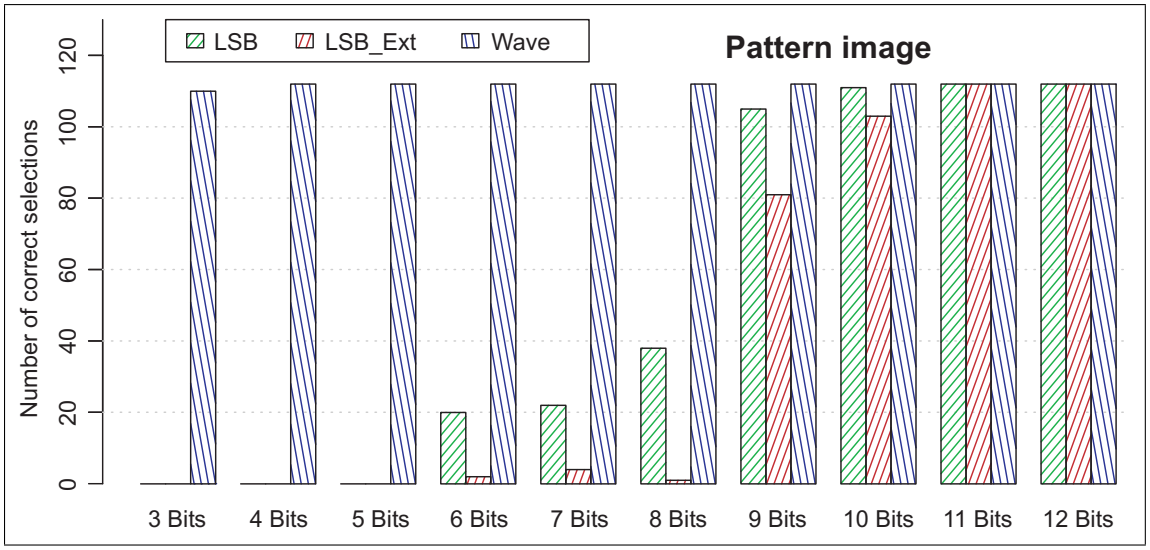


Figure 5.9: *Numbers of correctly selected regions in the* Pattern *image.*

A selection of times which passed until participants selected a region are shown in Figure 5.12. These times can give hints about the degree to which a particular watermarked region was visible: a low time value, for example, can indicate that a watermarked re-

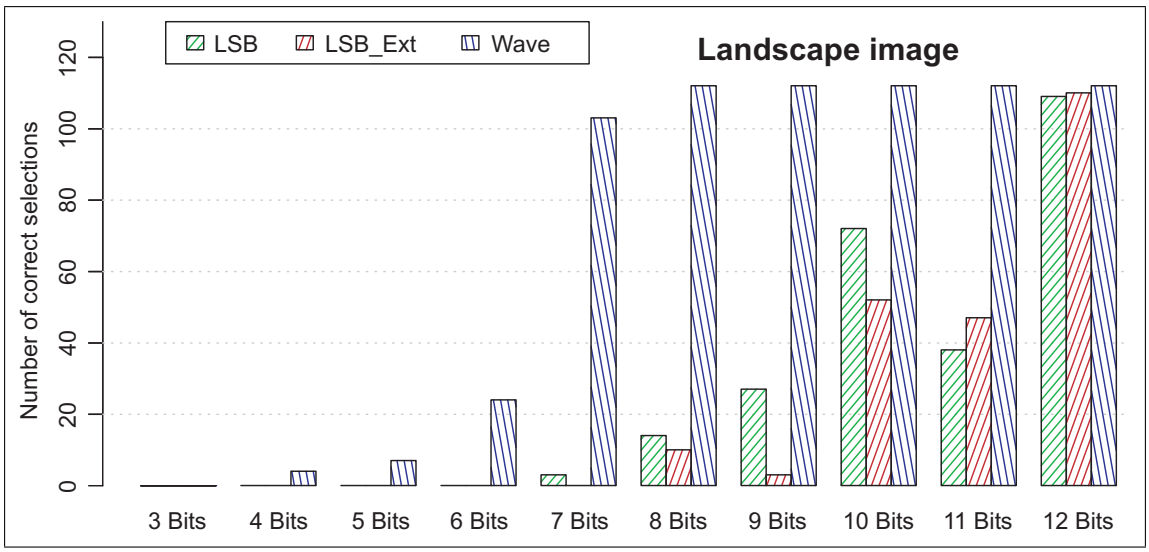


Figure 5.10: *Numbers of correctly selected regions in the* Landscape *image.*

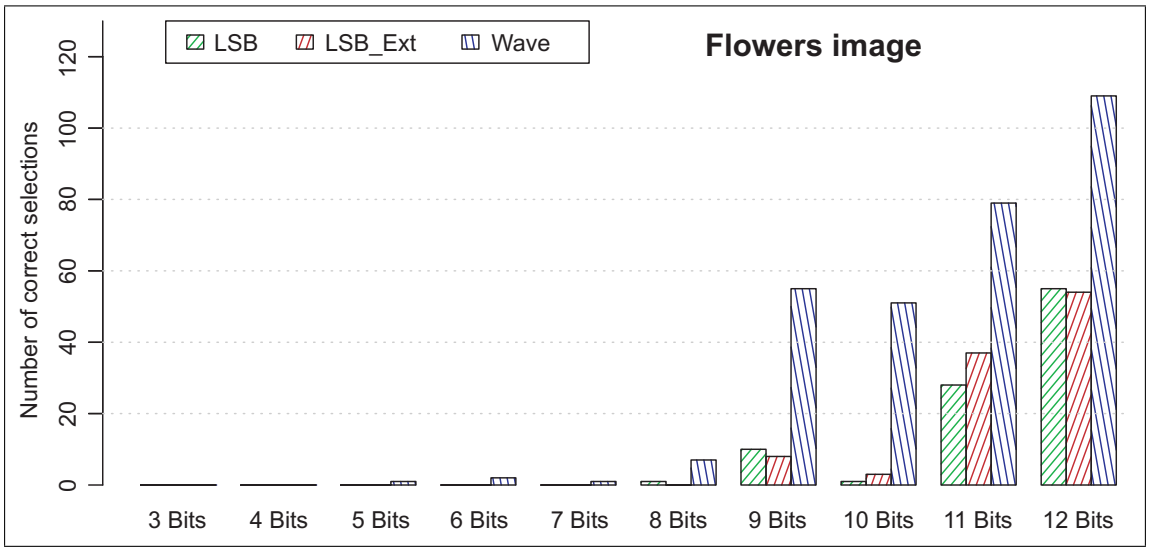


Figure 5.11: *Numbers of correctly selected regions in the* Flowers *image.*

gion was easily detectable. The diagram only considers the *LSB* and *LSB_Ext* technique since these techniques performed better than the *Wave* technique. The *Pattern* image was chosen because it has the lowest entropy compared to the other images so that image texture was less influencing. What can be gathered from the diagram is that elapsed times decreased with increasing capacity levels and that, for each capacity level, participants needed more time to make a selection when the *LSB_Ext* technique was used. The mean times for both techniques and the four selected capacity levels confirm this observation: $Mean_{LSB} = 5.78\ sec$ and $Mean_{LSB\_Ext} = 7.19\ sec$.

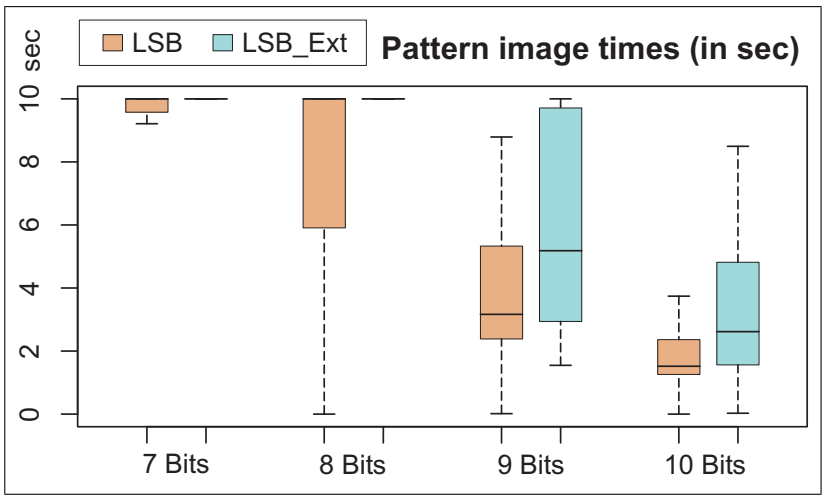


Figure 5.12: *Selective elapsed times for the* LSB *and* LSB_Ext *technique. The boxes provide information about the lower and upper quartiles as well as the medians.*

5.3 Discussion

The study has clearly shown that the *Wave* technique is inapplicable as *Illustration Watermarking* technique. This can basically be attributed to pixel modifications which can cumulate during coefficient watermarking (discussed in Sect. 4.4.3). The study has also shown that even though both the *LSB* and the *LSB_Ext* technique can be employed to embed respectable data rates, the *LSB_Ext* technique outperformed the basic *LSB* method.

One of the primary questions which should be answered by the study was how many bits can be imperceivably embedded into a pixel. Since the *LSB_Ext* technique was the technique that performed best, its results are analyzed to suggest potential capacity limits. The diagram in Figure 5.9 shows the number of correct selections participants made for the *Pattern* image which has the most homogeneous texture and which hence is best qualified to evaluate artifact visibility. The diagram reveals that at least 5 bits can be inserted per pixel. But even up to 8 bits would be possible since only 1 participant detected the region with this capacity level encoded (results for lower capacity levels: 6 bits: 2 participants, 7 bits: 4 participants).

There are two reasons which suggest raising the capacity limit. First, *Illustration Watermarking* is a technique that aims at embedding descriptive metadata into media which typically provide more information than the *Pattern* image does. Real-world images such as the *Landscape* or the *Flowers* image are hence rather addressed. They allow for embedding data rates of 8 or 9 bits per pixel. The second aspect is that the participants knew about modified regions they should search for. They were thus highly motivated to find those regions. A viewer, on the other hand, who simply looks at an image and who does not search for artifacts, would presumably be less likely to detect modified regions.

5.4 Summary

In this chapter, a user study was described which was conducted to research a selection of techniques which can be employed to embed *Illustration Watermarks* in raster graphics. To this end, each of the 112 voluntary participants was asked to look at images which had a certain region with watermarks embedded. The participants should find and select those regions which varied in position and degree of change. Besides technique evaluation, the influence of image type should be explored and an answer should be found to the question of how many bits can be imperceivably inserted into a pixel's color channel values.

The most important results this study revealed are:

- The *Wave* technique can only be used to insert very low data rates. The basic *LSB* technique can be employed to insert considerable data rates. The *LSB_Ext* technique performed best.
- Image texture had a significant effect on watermarked region detection. The higher the entropy of an image, the more the watermark signal was superimposed by the signal of the original texture.
- At least 5 data bits can be inserted per pixel by employing the *LSB_Ext* technique. Taking into consideration image texture, a capacity of 8 bits and more is possible.

As to the hypotheses which were stated at the beginning of this chapter, each of the three hypotheses was confirmed:

Hypothesis 1: The image with the highest entropy conceals encoded data most.

The *Flowers* image has the highest entropy ($H = 7.39$) compared to the other images involved. The diagram in Figure 5.8 clearly shows that participants detected less watermarked regions when this type of image was presented. But also the *Landscape* image ($H = 7.24$) caused participants to fail detecting watermarked regions considerably more frequently than the *Pattern* image ($H = 1$). This indicates that images with high entropies conceal encoded data more than images with lower entropies.

Hypothesis 2: Detection of watermarked regions is easier when the amount of encoded data increases.

The diagram in Figure 5.9 reveals that only a capacity level higher than 5 bits caused participants to detect watermarked regions in case of the *LSB* and *LSB_Ext* technique. When 10 bits or more were encoded, almost each participant was able to detect every watermarked region. The increasing detection rates with increasing capacity levels can be observed in the other diagrams in Figures 5.8, 5.10, and 5.11, as well.

Hypothesis 3: The *LSB_Ext* technique conceals encoded data most effectively.

The diagrams in Figures 5.9–5.11 show that images which were coded using the *LSB_Ext* technique concealed watermark data best in most cases. Only in 22.0 % of all selections that participants could make the correct region was detected (*LSB*: 26.1 %, *Wave*: 63.1 %).

6 Interaction with Embedded Data

Embedding descriptive metadata in digital media serves the purpose of providing information in addition to the information that is shown by nature. The previous chapters described various techniques which can be employed to store the metadata associated with a medium. However, the device that eventually displays the content of a medium is limited in space. Its resolution defines the number of pixels which can be exploited to present the information conveyed by the medium. But in contrast to, for example, printed media, digital media have the advantage that their contents can be dynamically adapted according to the demands an end-user has. For this reason, the information included in the *Illustration Layer* can be displayed temporarily at will. But many challenges exist in this regard. These comprise, for instance, the manner in which *Metadata-Related Information* can be provided. Such information indicates which parts of the medium are associated with metadata and what about is its content. Another challenge is to adequately display the metadata.

In this chapter, aspects that are associated with *Illustration Layer* data display are analyzed and discussed. Section 6.1 provides in this regard a brief overview of basic aspects which are important with respect to systematic information display. Thereafter, Section 6.2 introduces techniques which can serve to provide *Metadata-Related Information*. Aspects concerning the actual metadata display including the presentation of an experimental study which was conducted are addressed in Section 6.3. Finally, Section 6.4 summarizes this chapter.

6.1 Introductory Considerations

The descriptive metadata included in the *Illustration Layer* is invisible by nature. Even when an end-user knows that there is other information available, questions about the metadata and how it correlates with the medium remain. Without information about the metadata, a user has to spend a great deal of time searching the information that is of interest, while not knowing if at all it exists.

6.1.1 Metadata-Related Information

Two basic aspects facilitate gathering the expected information. The first is to guide the viewer to those regions of the medium for which metadata is available. This can be done by either highlighting all regions which are associated with metadata or by directing a viewer selectively to those regions the metadata behind which corresponds with the viewer's interests, provided that these had been specified before. The second aspect concerns information

about the type and content of the metadata. Some users, for example, may solely be interested in textual metadata whereas others want to hear sounds which are in relation to the medium. Information about its content can reveal details such as whether a text describes what can be seen or whether it describes how the medium was generated and by whom. In this research, the term ***Metadata-Related Information*** refers to location- and content-related information which concerns the metadata of a medium. For its adequate display, aspects regarding a viewer's *Visual Attention* play a prominent role.

6.1.2 Visual Attention

What happens, in short, when viewing an image? According to Yantis [Yan98], viewing starts with *early vision* (other authors call it *low-level vision*) by which the retina and early cortical areas respond to patterns of light and dark that vary over space and time. On this stage, visual input might be perceived as a pattern similar to what is shown in Figure 6.1(b). *High-level vision* (see Fig. 6.1(d)), on the other hand, comprises complex operations such as interpreting and understanding visual information (e. g., object and face recognition, mental imagery, etc.). It is situated at the end of the pipeline when viewing is regarded as continuous process. In between these levels of vision, perceptual organization takes place and attention is focused on certain image features (*intermediate-level vision*). While perceptual organization creates concrete object representations from what is delivered by early vision, visual attention acts as filter before high-level vision. It covers perceptual operations responsible for selecting those features which are provided for further detailed visual processing [Yan98].

Hatfield [Hat98] expresses visual attention in terms of stimuli being in competition to be perceived. Following his thoughts, attention can be shifted by objects that seem to be unusual and extraordinary. However, according to the author, perceiving a whole tree, then switching the focus to the leaves, branch and trunk, successively, is also a form of attention shifting, even though the structure of a tree is typically not unusual or extraordinary. This shows that the eye, indeed, permanently scans its visual environment to seek for surprising features.

When viewing the image which is shown in Figure 6.1(a), there are most likely several stimuli which are perceived in a certain sequence depending on how well they did during the competition. Region A in Figure 6.1(d) (marked by the blue rectangle) could be one of those prominent stimuli. If so, the viewer's attention was shifted by the content of the image. But prominent stimuli do not necessarily lead to descriptive metadata. On the one hand, not every eye-catching image region is associated with metadata. On the other hand, there can be image regions with metadata associated which are permanently ignored by visual attention. Region B in Figure 6.1(d) could be such a region. But what if exactly this small region shows a particular person's residence for which metadata is available? In this case, the viewer's attention must be guided to the region by means of visual cues. After interpreting such a cue correctly during high-level vision, the viewer may focus the region shown in the rightmost bottom image—pointed at by the lower arrow which serves as a visual cue.

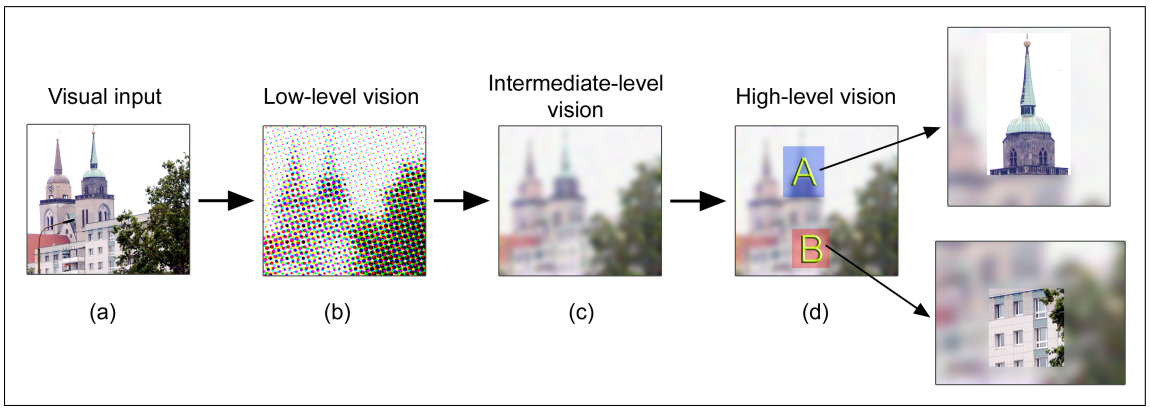


Figure 6.1: *Levels of vision. Intermediate-level vision (c) analyzes what is delivered by low-level vision (b) and sends a filtered version to high-level vision (d). Without visual cues, a viewer's attention may initially be shifted to region A. Region B, on the other hand, may only be noticed when visual cues are employed.*

6.2 Guiding a Viewer's Visual Attention

As stated in the previous section, to support a viewer in gathering the descriptive metadata which is available, it is necessary to strengthen those visual stimuli which are associated with the metadata. The viewer must be assisted during filtering the visual input so that the detailed information sought can be provided.

The process: "information overview followed by filtering, then receiving the information sought" correlates with SHNEIDERMAN's *Visual Information Seeking Mantra*: "overview first, zoom and filter, then details-on-demand" [Shn96]. However, even though SHNEIDERMAN's mantra and the visualization pipeline (illustrated in Fig. 6.1) follow a similar principle, there are significant differences. SHNEIDERMAN assumes that there is a large information space which cannot be visualized in its entirety at once. The viewer must hence select from a specific view of the information space which was scaled down and reduced to classes which are represented by certain members. Selecting such a member yields then, in case of success, the detailed information searched by the viewer. In the scenario of *Illustration Watermarking*, a large information space is also subdivided, but in a different way. Instead of information being represented by various abstract classes, the information space is here subdivided by means of the two layers: information represented by the *Background Layer* and information represented by the *Illustration Layer*. The term "overview" thus refers to differing representations: abstract classes of the whole visual input (SHNEIDERMAN's mantra) and arbitrary visual input provided by the *Background Layer* (*Illustration Watermarking* scenario).

However, both scenarios aim at providing the end-user with detailed information that correlates with the information already visualized. Also, in both scenarios, the detailed information cannot be displayed at once due to limitations in space and potential information overloads since humans are only capable to handle limited information at a definite time. And finally, in both scenarios, the viewer needs assistance in reaching the detailed

information sought. This can be done, for example, by filtering a viewer's interests in search masks or by augmenting the visual information already present with visual cues integrated with hindsight.

6.2.1 Visual Cues and What Makes Them Effective

Image regions of same colors are unattractive to the human eye which is rather interested in contrast. High-contrast regions hence attract the human eye more and demand longer time spans of eye fixation than low-contrast regions. Consequently, visual cues, which claim to be effective in attracting visual attention, must be more interesting to the human eye than those regions otherwise considered first.

Matlin and Foley state that specifically edges and change are important for human vision [MF97]. According to the authors, *edges* are locations of sudden change in color, brightness, or lightness. They contribute to increasing the contrast in image regions and hence, are eye-attracting. What is meant by "change" can be illustrated with the example shown in Figure 6.2(e). All circles are filled with patterns that feature high contrast, except for one circle which is single-colored. Because of its significant difference to the remaining circles, this circle typically attracts a viewer's first attention.

Preattentive processing

According to Treisman [Tre85], visual information processing comprises *preattentive processing* and *focused attention*. Preattentive processing corresponds with humans' low-level vision. It happens automatically and aims at rapidly detecting basic object features by analyzing the whole visual input. Within several studies, a limited set of visual properties—called *preattentive features*—were discovered which have a great effect on low-level vision (overviewed in [Hea99]). Examples are shown in Figure 6.2. In each image, one particular *target object* draws the viewer's attention. In Figure 6.2(a), for example, the red circle is immediately focused by the viewer because it is in significant contrast to the white background. As to the other images, all objects can be classified according to certain properties (e. g., color, size, shape, etc.), except for the target object that differs in one of these class properties.

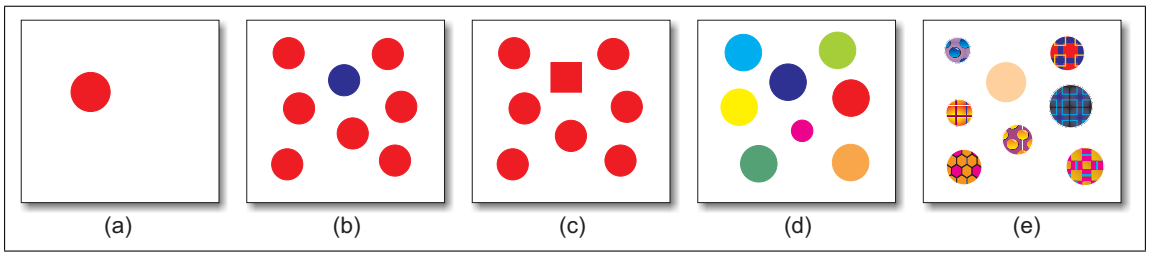


Figure 6.2: *Preattentive features. In each image, one specific object—the target object—attracts visual attention.*

In contrast to preattentive processing, which is done in parallel, focused attention is a

serial process and hence, more time-consuming. It is assumed that preattentive processing provides the information that guides focused attention.

Differences between visual cues

A viewer's focus can be guided by integrating preattentive features. Such a feature could be one of those arrows shown in the second image in Figure 6.3 because: (1) its contrast to the background is adequate and (2) the correct interpretation of the symbol "arrow" provides topographic information.

Figure 6.3: *The left hand side image has several regions with metadata associated. Techniques for communicating those regions to the viewer are arrows integrated into the scene (center) or highlighted contours combined with blurred context (right).*

But including arrows the way it was done in Figure 6.3 has also several drawbacks:

- ***Occlusion.*** The arrows occlude context information. The more regions are associated with metadata, the more arrows are required, which increases occlusion.
- ***Impreciseness.*** The arrows point at regions with metadata associated. They do not provide information about their shapes and extents.
- ***Incompleteness.*** The arrows provide no information regarding the type (short label, text, image, etc.) and amount of the metadata. They also do not indicate to what extent the metadata does correlate with the user's preferences.

A classic *focus+context* technique, which Kosara et al. [KMH02] termed as *Semantic Depth of Field*, is illustrated in the rightmost image in Figure 6.3. Here, the contours of augmented regions are emphasized and regions with no metadata associated are blurred. Employing visual cues in this way facilitates region detection. But it also hinders to perceive all details in the image. Besides, the characteristic "incompleteness" applies to this technique, as well.

Requirements for effective visual cues

Besides attracting a viewer's attention, visual cues should give hints concerning the information a viewer cannot see but which could be of interest. To be effective in this regard, visual cues should fulfill the following requirements:

- ***Rapid detectability.*** Visual cues should be immediately perceived as such. This can be achieved by including appropriate objects or by altering parts of the visual input.
- ***Prevention of context loss.*** Context occlusion should be avoided. In case of modifying context information, the introduced changes should be as unobtrusive as possible.
- ***Clearness.*** The correlation between visual cue and region being addressed should be clear. Furthermore, the scope of a visual cue should be obvious.
- ***Richness in information content.*** Visual cues should provide information about what is referenced. This can include the type of information, its amount, and how much it does correlate with the viewer's interests.
- ***Adaptability.*** It should be possible to adjust the strength of visual cues, which includes providing the opportunity to hide them.

Ideally, visual cues should conform with what Tufte [Tuf01] calls *Graphical Excellence*:

> "Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space."

6.2.2 Visual Cues in Related Work

Internet pages have embedded hyperlinks that allow users to travel from one unit of information to another. Analyzing those pages, two types of visual cues can be noticed: *active* and *passive* cues. Active cues permanently indicate that certain actions can be evoked by a user. They include underscored texts as well as words or phrases which have different colors than the remaining text. Shneiderman [Shn96] itemizes various other highlighting techniques such as boldface text, brightening, and boxing. But also blinking texts or images can be exploited as active visual cues. Passive cues, on the other hand, require user interaction to show up. Examples are mouse cursors that change their style to signalize that an object (e. g., an image integrated into a page) serves as a link to another portion of information.

Visual cues in 3D visualizations

Preim and Ritter [PR02] classified several techniques for emphasizing objects in 3D visualizations. They differentiate techniques according to the amount to which the visualization is being modified due to object highlighting. For example, changing colors,

displaying object contours or bounding boxes, or integrating arrows and other symbols are *local emphasis techniques*. In contrast, rendering context objects transparently or moving them aside by applying 3D-fisheye lenses are techniques that modify scene objects beyond those being focused (*regional* or *global emphasis techniques*). Some of those techniques can be attributed to the field of non-photorealistic rendering because they consciously abstract scene components, render silhouettes in specific styles, or only roughly indicate the presence of objects in the context [PTD05].

Another technique which can be employed to guide a viewer's attention is to integrate auxiliary planes into the visualization onto which certain information units can be projected. In the approach presented by König et al. [KDG99], three such planes are arranged in a cubic form around the actual rendition. By varying the render attributes, the individual planes can show different visualizations of the model (e. g., rendition of solely those model components being focused). Ritter et al. [RSHS03] followed a similar approach. They integrate a shadow plane into the visualization to provide information which cannot be extracted from the actual rendition of the model.

Due to the increasing amounts of data and the limitations that exist in perceiving the data, Bartram [Bar97] reasoned that additional communication and representation dimensions are required to improve information bandwidth. From her point of view, "motion", which is known to be perceptually salient, is a promising candidate. Advantages of using "motion" to convey information are, for example, its perceptual efficiency and its rich interpretative scope. In this regard, Jesse [JS01] focused on data glyphs which are commonly exploited to map data onto graphical representations. He expanded the set of their display variables (e. g., color and shape) with "motion".

Visual cues with meaning

The "Post-it" metaphor which is attributed to the well-known "sticky notes" can roughly be described as visual cue with meaning. Due to the mostly flashy appearance of those "sticky notes", they immediately attract attention so that the information provided quickly reaches the person being addressed. The "Post-it" metaphor has been implemented in several approaches. An example was presented by Jung et al. [JGD02] who employed it for annotating 3D models. Their approach, which was designed to facilitate collaborative work, allows users to leave notes, symbolized by geometric objects, within a 3D visualization. The textual content represented by those geometric objects can be retrieved in a separate window. A similar approach was followed by Loughlin and Hughes [LH94]. They also employ small geometric objects as graphic annotation markers. Those markers can be assigned graphic attributes (e. g., differing shapes) that give visual feedback on what is represented. The annotation markers can also serve as landmarks to indicate interesting points.

Toyama et al. [TLRA03] add symbols to maps that indicate the presence of pictures which are associated with the marked map locations. The size of those symbols correlates with the number of pictures being represented. However, besides the attribute "size", many other visual variables can be exploited to communicate information. In [AA99], for example, the authors make use of size, shape, color, and value as well as combinations of these

attributes to facilitate visual data exploration. Assigning various attributes to brush strokes (e. g., different colors, sizes, densities, or orientations) was exploited by Healey [Hea01] to express environmental and weather conditions. In [KLS00], the attributes "location", "orientation", and "length" are assigned to geometric primitives (connected cylinders) to convey information.

Exploitation of Magic Lenses

The term *Magic Lens* refers to operators which are placed in front of a visualization to display alternative information within the region that is confined by the lens. *Magic Lenses* can be exploited for various purposes. Loughlin and Hughes [LH94], for example, assigned different functionalities to the lenses they used: (1) emphasizing annotation markers by setting the colors of all other objects to gray, (2) displaying only those markers that satisfy the criteria specified by the user, or (3) hiding all markers behind the lens.

Classic applications of *Magic Lenses* were proposed by various authors (e. g., [BSP+93, SFB94, VCWP96]). They display, for example, more detailed information within the scope of a lens, change the rendering style, or hide information that distracts from the focused data. But *Magic Lenses* could also be used to show a preview of the descriptive metadata which is linked with a particular visual hint.

6.2.3 Approaches for Metadata-Related Information Display

Based on the considerations made, this paragraph proposes two techniques which were developed in the course of this work that combine both drawing a viewer's attention to certain regions and providing information about the associated metadata.

The *Informative Cursor*

Informative Cursor is an approach which has not been implemented so far. Simple prototypes of the *Informative Cursor* are mouse cursors that change their styles to indicate that a certain action can be performed (e. g., hand cursor over Web links) or that the computer is busy (e. g., sandglass cursor). *Informative Cursors* can be regarded as extensions of those traditional cursors. Besides signalizing the current position of the input device, which is a cursor's original function, *Informative Cursors* provide four other kinds of information:

1. ***Regions with metadata.*** The shape of the cursor consists of two arrows (see Fig. 6.4). Each arrow points at a region which is associated with metadata. The length of an arrow corresponds, inversely, with the relative distance to the nearest region. That is, if one of the two arrows is longer than the other, the region pointed at by the longer arrow is closer. While moving the cursor, the arrow lengths change smoothly but are kept within a certain margin (see Fig. 6.4(a)).

2. ***Amount of metadata.*** The amount of metadata associated with a region is expressed by the thickness of an arrow (see Fig. 6.4(b)). This amount increases with the number of bytes that represent the metadata. Hence, more words, a larger image, or a collection of differing metadata types yields a thicker arrow.

3. ***Type of metadata.*** Types of metadata which can be expressed by the cursor are text, image, sound, or a combination of them. A symbol, which is the initial letter or an $'A'$ in case of any combination (see Fig. 6.4(c)), is mapped as hint onto the arrowhead.

4. ***Accordance with interests.*** The accordance of the metadata's content with interests a user had specified before can be gathered from the gray shade of an arrow (gray shades can express quantity better than color [Tuf01]). The darker an arrow, the more contributing is the metadata (see Fig. 6.4(d)). However, deciding whether one portion of knowledge (viewer's interests) correlates with another portion of knowledge (metadata's content) is a very difficult task and no issue in this work. Hence, it is assumed that another system provides those levels of accordance.

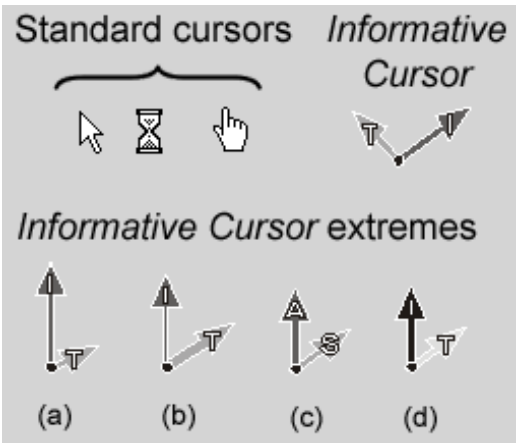


Figure 6.4: *Illustrations of* Informative Cursors *whose sizes can be compared with those of traditional cursors. The lower row shows their extremes: (a) longest and shortest, (b) thickest and thinnest, (c) character types, and (d) darkest and brightest.*

The two images in Figure 6.5 illustrate the use of the cursor. In those images, there are two regions with metadata associated which are situated in the cursor's neighborhood. Those regions show one of the daisy blooms (*Region 1*) and a part of a dandelion leaf (*Region 2*). Two conditions are assumed: (1) *Region 1* is associated with an image as metadata and *Region 2* has corresponding text. (2) There is a higher correlation between the viewer's interests and the metadata of *Region 1*.

In the first image, the cursor is closer to *Region 1*, which is signalized by the differing lengths of the arrows that point at the regions. In the second image, it is vice versa. The widths of the cursor's arrows give hints regarding the amounts of metadata. Obviously, *Region 2* has, in terms of data bytes, less metadata than *Region 1*. What can also be observed is that the region being closest to the cursor is emphasized by highlighting its contour. The viewer can thus quickly reach the region and demand the metadata. Once the mouse cursor is over a region, its shape is changed to that of the standard cursor.

Nevertheless, there are special cases that need special handling. Examples are two nearest regions being situated in the same direction or two regions that have exactly the

Figure 6.5: *Images in which* Informative Cursors *are used.*

same distance from the cursor. Solutions can be to hide one of the two arrows (1st example case) and to choose the region whose metadata corresponds more with the viewer's interests (2nd example case).

The proposed technique is not intended for exploring all the regions in an image which have metadata. For such a purpose, overview information would be necessary that facilitates navigation. The proposed technique rather aims at viewers who spot interesting features in an image, head for those regions, and, finally, request further information if available. Compared to the arrow metaphor exploited in Figure 6.3, *Informative Cursors* have various advantages the most important of which are: they occlude less context information, they provide information about the metadata, and the extent of regions being focused is indicated.

Even though more stylistic features could be employed for communicating information with *Informative Cursors* (e. g., background color, texture, etc.), it would not necessarily contribute to augmenting comprehension. Tufte states in this regard:

> "The danger of multifunctioning elements is that they tend to generate graphical puzzles, with encodings that can only be broken by their inventor."

Meta-Previewer: Exploiting the *Magic Lens* metaphor

Meta-Previewer is a tool that displays both locations of all regions which have metadata associated and information about the metadata associated with a focused region. It follows the idea behind *Magic Lenses* since the data which is displayed by the *Meta-Previewer* differs from the original visual data underneath.

The *Meta-Previewer* has a rectangular shape whose extent correlates with the size of the visualization to which it is applied. The data displayed by the previewer is split in two components (see Fig. 6.6):

1. *Background area with region markers.* The background area consists of a down-scaled copy of the visualization into which markers are integrated that indicate the locations of augmented regions. A certain region of this background area—the focused region—is magnified by distortion (compare with *Fisheye views*, e. g., [LA94, CM01, GS03] and *non-linear magnification* [KR97]).

2. *Frame with information about the metadata.* The background area has a frame that displays information about the metadata. Similar to the *Informative Cursor*, it includes information about the metadata type and amount as well as information about the accordance with the viewer's interests. Values are expressed by variable bars.

The *Meta-Previewer* can be moved across the visualization. Figure 6.6 illustrates a potential scenario in which the previewer's initial position is at the bottom of the visualization. Which region is focused by the previewer depends on its position relative to the visualization. In the figure, the initial focal region is the region around the dandelion leaf, which yields this region to be magnified by the *Meta-Previewer*. When the *Meta-Previewer* is moved towards the dandelion bloom, it changes its state, which is illustrated in Figure 6.6(a)–(d):

(a) The visualization is copied and region markers are inserted. Region markers are small spheres which were designed with the objective of quickly attracting a viewer's attention. The markers' locations are the centroids of the corresponding regions. Besides serving as identifiers, those markers can also be selected to request the actual metadata.

(b) The augmented region which is closest to the center of the *Meta-Previewer* (the focused region) is emphasized by highlighting its contour.

(c) A non-linear distortion lens is applied to the image so that the focused region is magnified. The effect is that, besides the region being focused, the region markers around the focal region are magnified whereas other markers situated in the periphery are scaled down.

(d) Finally, information about the metadata associated with the focal region is attached to the *Meta-Previewer*. This includes information about which types of metadata

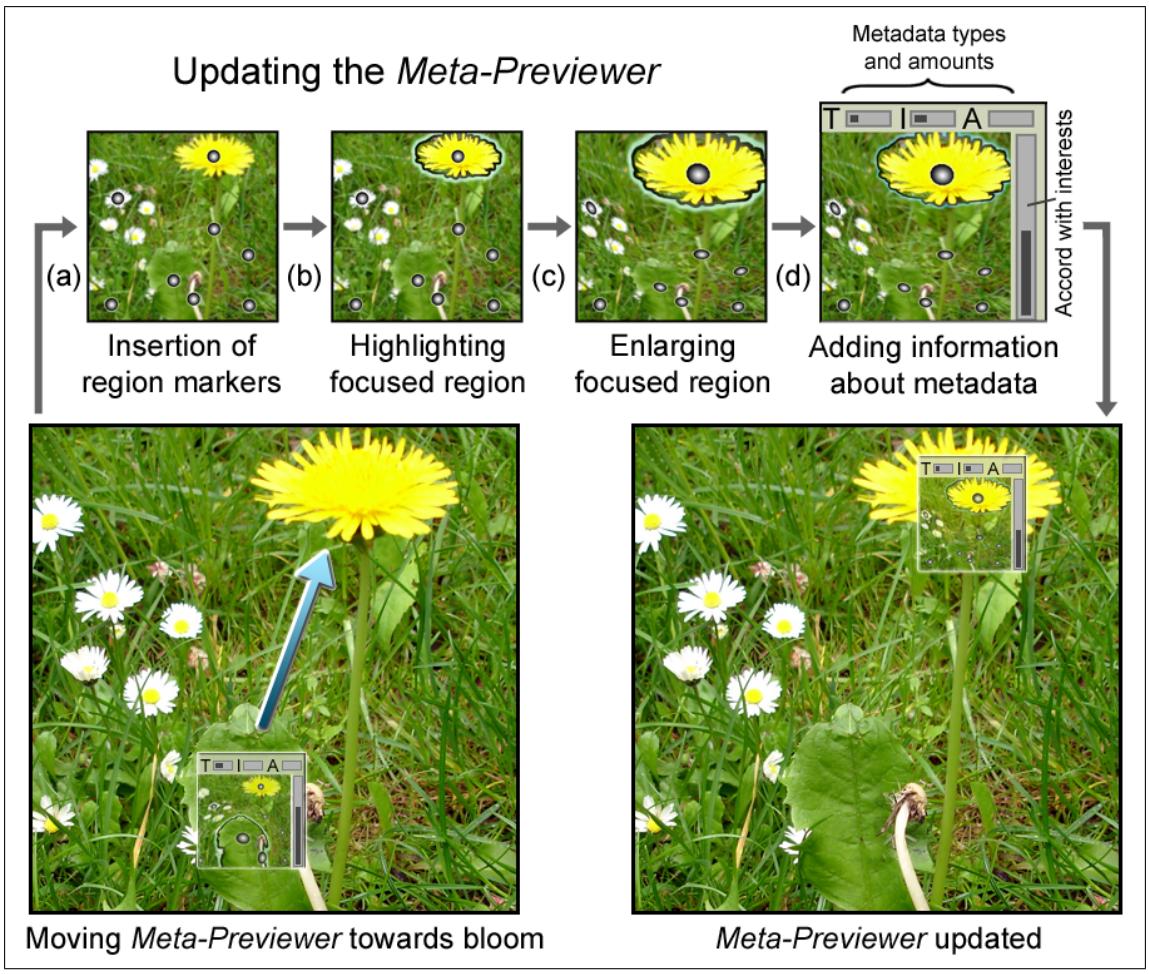


Figure 6.6: *When the* Meta-Previewer *is moved across the visualization, its content is permanently updated. This includes magnifying the region being focused as well as displaying information about the focused region's metadata.*

are available and what are their relative amounts (in terms of bytes). In the figure, for example, there is some textual and a bit more visual information available. In addition, the accordance with the viewer's interests (compare with the previous paragraph) is expressed by the vertical bar.

The necessity for applying a magnification lens technique follows from the down-scaled view of the *Meta-Previewer*. Visualization details as well as region markers would be hard to recognize without the focal regions being magnified.

The advantage of the tiny view is that only a small portion of the original visualization is occluded. To further compensate the occlusion problem, the *Meta-Previewer* could be displayed semitransparently. Regions of the original visualization, superimposed by the previewer, could then still be perceived at a fraction. In this regard, Baudisch and Gutwin [BG04] proposed a technique, which they call *multiblending*, that improves per-

ceiving details semi-occluded by overlayed data.

Discussion

The two proposed approaches basically aim at guiding a viewer's attention in 2D raster graphics. The question that arises is:

Can those techniques be employed in 3D space, as well?

The answer is "yes", but they must be adapted. The *Meta-Previewer*, for example, takes as input a 2D raster graphic. Rendering a 3D representation results in such a graphic so that combining 3D and *Meta-Previewer* is imaginable. But navigation in 3D space and occlusion of 3D components are two new aspects which must be considered.

Navigation in 3D is in general a difficult task when adequate equipment (e. g., 3D input/output devices) is lacking. Assuming that navigation is facilitated by such equipment, the *Informative Cursor* could be coupled with camera navigation. That is, the arrows would suggest directions for camera movement. The *Meta-Previewer*, on the other hand, can remain in the 2D viewing plane and display what is delivered by the camera. The camera is then controlled independently from the previewer.

The second aspect, occlusion, can easily be handled in case of the *Informative Cursor* because direction and length of an arrow provide sufficient information to localize a region. But in case of the *Meta-Previewer*, the region markers should indicate whether a referred region is currently visible or not. Perhaps the region markers could also provide information about a region's distance from the viewing plane, for example, in terms of adaptive gray tones.

6.3 Presentation of Descriptive Metadata

The previous section addressed techniques that facilitate the search for regions which are associated with metadata an end-user could be interested in. This section discusses techniques which can be employed to present metadata adequately. Besides broaching the issue of what should be considered when metadata is to be displayed, the specific challenge of integrating textual information in explorable 3D visualizations is addressed.

6.3.1 Displaying Illustration Layer Data: Basic Guidelines

When descriptive metadata is demanded by the viewer, the data of the *Background Layer* is temporarily augmented with data of the *Illustration Layer*. The *Illustration Layer* in itself can consist of differing types of metadata (e. g., textual, visual, and auditory data) that need specific handling.

The metadata which is addressed

Besides auditory metadata, textual and visual data are likely those types most in use. The term *textual metadata*, as it is used within the scope of this section, refers to a limited

number of alphanumeric characters (around 500 characters). Longer texts (including Web pages) which can also be associated with a region should be displayed in a separate window to prevent occlusion, which is trivial. The term *visual metadata* will refer to images whose extent does not exceed a certain fraction of the visualization being explored (e. g., 50% of the visualization's extent). Larger images should be displayed in separate windows, as well.

Basic guidelines

Two important aspects which should be considered when *Illustration Layer* data is to be displayed are: (1) the displayed data shall be in semantic relation with the information provided by the *Background Layer*, and (2) the manner in which the metadata is presented shall not be confusing. Further requirements are:

1. ***Readability.*** Convenient reading of text requires, amongst others, a sufficient contrast to the background, an adequate font and font size, and an appropriate technique for scrolling through the text if it does not entirely fit into its assigned region (further aspects are discussed in [MW87]). As regards images, image content and image size should harmonize; too much detail put in too small space makes no sense.

2. ***Clarity of attribution.*** The correlation between a *Background Layer* region and the associated component of the *Illustration Layer* must be clear. This is particularly important when metadata of more than one region is displayed.

3. ***Prevention of occlusion.*** Occlusion means that parts of the *Background Layer* are invisible to the viewer, which can result in context loss. Occlusion should thus be avoided whenever it is realizable.

4. ***Coherency and Consistency.*** The design and usability of provided metadata are important. Textual metadata, for example, should be presented using consistent design attributes such as font, font size, accentuation, or color. Also, attribute-meaning pairs (e. g., bold for importance) should be maintained. When a viewer can interact with the displayed metadata (e. g., scrolling through text), it should be clearly signalized. Interaction in itself should be simple and same tasks should be completed by employing same techniques.

5. ***Adaptability.*** Modifying display attributes at will should be permitted. This includes, for example, hiding metadata, changing its size and position, and customizing text attributes.

Shneiderman and Plaisant [SP04a] itemized further aspects, for example, data should only be presented if they assist the user, information should be presented graphically, where appropriate, and users should be involved in the development of new displays and procedures.

6.3.2 Techniques Proposed in Related Work

Many techniques exist to provide a portion of information which is part of the *Illustration Layer*. These techniques are adapted according to the type of information to be presented (e. g., text labels, longer texts, images, etc.), the dimension of the data presentation space (e. g., 2D or 3D), or the manner in which a user can interact with the provided information (e. g., selection of information items). But these techniques also differ in whether or not they modify parts of the *Background Layer* for data presentation (see Fig. 6.7). In this regard, they can be classified in:

- ***Integration techniques.*** Techniques that aim at presenting the information close-by their counterparts without modifying the *Background Layer*. To this end, they try to exploit empty or insubstantial regions. The occlusion of meaningful content is avoided.
- ***Spatial separation techniques.*** Techniques that solely use space beyond that occupied by the *Background Layer's* visual content. A clear separation between *Background* and *Illustration Layer* can be observed.
- ***Deformation techniques.*** Techniques that intentionally modify parts of the *Background Layer* to enable the visualization of other information. The presented information can correlate with what was replaced.

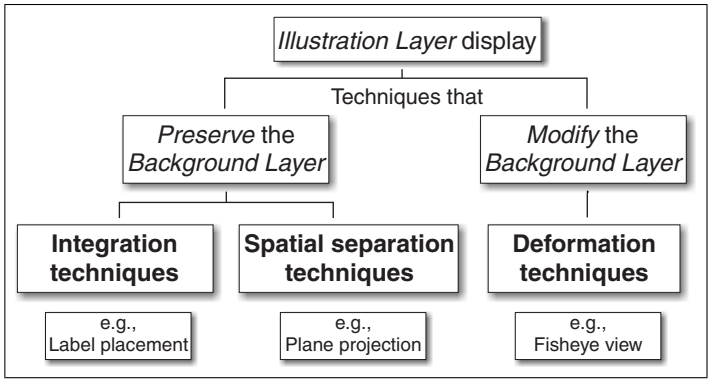


Figure 6.7: *Classification of techniques with which information can be presented.*

Integration techniques

According to Kakoulis and Tollis [KT98], the integration of text labels into the visualization involves the compliance with three basic rules: (1) overlapping of a label with other labels or other graphical features of the drawing should be avoided, (2) it should be easy to identify for each label the graphical feature of the drawing that is addressed, and (3) among all acceptable positions, each label should be placed in the best possible position.

The occlusion of essential parts of the *Background Layer* can be avoided by identifying its empty or unimportant regions. Ishak and Feiner [IF04], for example, proposed a technique that employs unimportant regions to visualize other data which is in their particular use case the content of windows underneath the focused. *Dual-Use of Image Space* [CS02] is another technique that makes use of untextured regions for text display. It even goes one step further since the employed space itself is part of the region to be labeled.

The latter technique is also an example for keeping minimal the distance between label and referenced region. The importance of placing labels directly on and around their regions was pointed out by Li et al. [LPS98]. The authors also discussed the aspect that the space which is available around a region decreases with labels placed close-by. Another dynamic technique that focuses labeling of a neighborhood of objects was proposed by Fekete and Plaisant [FP99].

Metadata presentation in 3D space is associated with further challenges since changing the view in three dimensions is feasible. A prominent example is *Voxel-Man* ([HPP+95, PHP+01]) which is a system that allows to explore the human body interactively. The system comprises various spatial exploration techniques. But textual metadata is basically provided in context menus and by using a simple "object-line-label" technique. There exist many other systems that use variations of the "object-line-label" technique (e. g., [BWHR99, GKH+99]). Preim et al. [PRS97] proposed a system called *Zoom Illustrator* which places annotation boxes at fixed positions beside the 3D model. For certain text items within those boxes, more detailed information can be requested, which is then displayed by employing a zoom technique. Götzelmann et al. [GHS06] improved the technique of placing text labels. To this end, they use "agents" that permanently search for adequate positions nearby the model.

Augmented Reality applications demand for intelligent placement of metadata that correlates with what a user sees on the display, as well. In this regard, Bell et al. [BFH01, BFH02] proposed techniques for integrating labels and pop-up windows. Their techniques adjust positioning and size of the metadata so that overlapping with important content is avoided.

Spatial separation techniques

Spatial separation of visualization and metadata basically serves two purposes: interferences with the *Background Layer* are avoided and, at the same time, a considerable amount of data can be presented. A general approach is to attach a new window to the window that displays *Background Layer* data. The new window, which is visually separated by its frame, can then be employed for showing *Illustration Layer* data (e. g., [SS00, JGD02]). However, due to long spatial distances between visualization and metadata, it is often difficult to point out correlations. Thus, only metadata for the focused visualization component is shown in most cases.

Including primitives such as illustration planes arranged in 3D space and close-by the visualization of the *Background Layer* is another technique. Besides adding self-contained display space (e. g., rondell surfaces in [PRS97]), they can also serve for displaying illustra-

tive views of the visualization (e. g., [KDG99]). In this regard, shadow projections are an example for simplifying the original visualization without context loss so that individual components can easily be highlighted (e. g., [RMC91, HZR+92, RSHS03]).

Deformation techniques

The two problems Leung and Apperley [LA94] see when data is to be presented in confined space are a spatial and an information density problem. A technique that solves those presentation problems only in parts was presented by Bouvin et al. [BZGM02]. With the focus on the layout of 2D Web pages, they create space for annotations by moving aside those items that would reach into the space covered by the annotation. However, moving aside items can also yield information loss, particularly when space is confined.

Distortion-oriented techniques, on the other hand, aim at keeping the context while expanding the space for displaying details. Such techniques have been overviewed, classified, and evaluated in various publications (e. g., [LA94, BGBS02, GF04]). Several approaches that employ distortion-oriented techniques were proposed. Examples are: the *Document Lens*—a tool for navigating in documents with unknown structure [RM93], exploration tools for large information spaces such as graphs, maps, or 3D structures (e. g., [SB92, CCF95, CCF97]), or an approach that allows for displaying Web pages in their entireties [BLH04].

Magic Lenses can also be regarded as a technique that modifies the *Background Layer* for information presentation (e. g., [BSP+93, LH94, VCWP96]). But in contrast to distortion-oriented techniques, they do not necessarily magnify the content underneath. Also, there is typically no smooth transition between revealed information and displaced context, even though there is generally some correlation between them.

6.3.3 Approach for the Exploration of 3D Visualizations

Displaying *Illustration Layer* data in 3D space is challenging, especially when the information to be presented is text. Besides the challenge of maintaining correlations and readability, the 3D interface in itself is linked with a number of problems, for example: disorienting navigation, complex user actions, and annoying occlusions [Shn03].

Besides requesting *Illustration Layer* data, the exploration of a 3D visualization includes interacting with the camera for view control. But changing the view requires layout updating when *Illustration Layer* data is displayed. According to the afore itemized demands, this comprises searching for empty or unimportant regions as well as verifying that there is a clear correlation between *Background* and *Illustration Layer* components. Also, abrupt changes in the layout should be avoided because they can be annoying and confusing.

Figure 6.8 shows an annotation technique that directly attaches text to its corresponding scene component (published in [SCS04]). Besides object closeness, a semitransparent polygon provides further visual feedback regarding the object-text correlation. Images (a) and (b) illustrate design variations: (a) shows an annotation of rectangular shape that facilitates reading whereas (b) shows an annotation whose shape is adapted to the curvature of the corresponding scene object, which links object and annotation even closer.

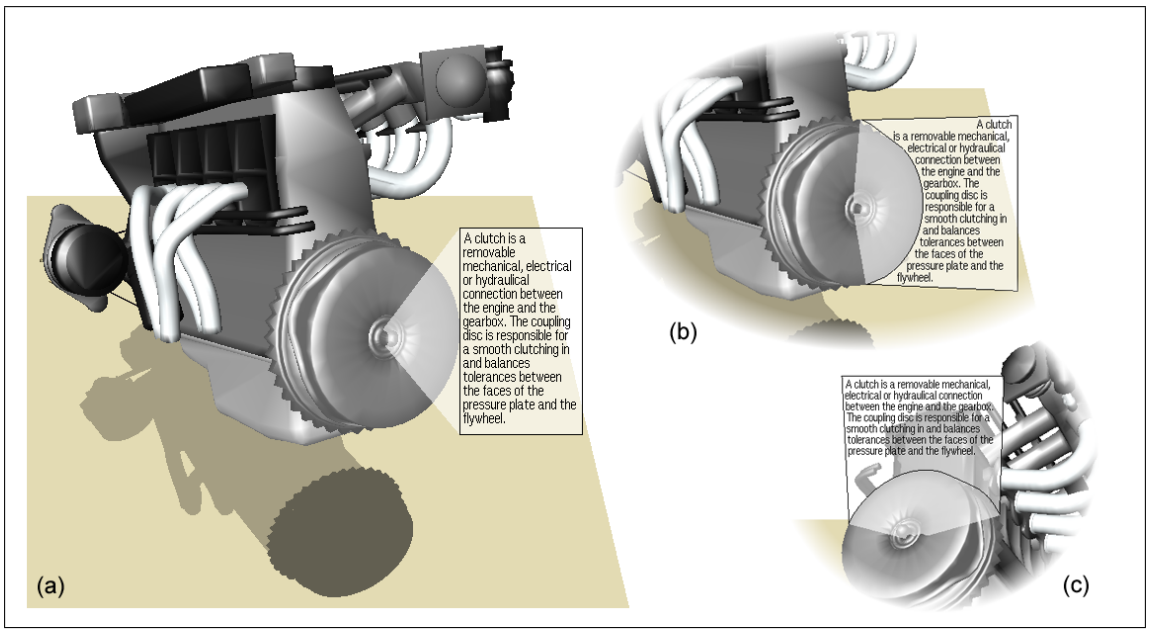


Figure 6.8: *Object-attached annotations. The annotation remains attached to its scene object when the view changes. Images (a) and (b) show design variations; (c) illustrates occlusion, which is reduced due to semitransparent rendering.*

The most important features of this technique are:

- Annotation reading is facilitated by exact screen-aligned text rendering.
- Correlation between scene component and annotation is clear due to object closeness.
- Annotations remain attached and move smoothly with their objects when the view changes during scene exploration.
- Annotation shape and size can be adapted so that scene occlusion can be reduced or the amount of displayed text can be adjusted.

The last point indicates that scene occlusion can be a drawback of this technique. Figure 6.8(c) illustrates such a scenario. But it also shows that the semitransparent rendering style still lets the viewer guess the structures being occluded. An alternative approach would be to place annotations with a certain distance outside of the area covered by the rendition. In static layouts, this method is appropriate. But when the layout permanently changes, longer distances not only complicate identifying correlations, they also increase the distance an annotation has to move when its counterpart changes the position.

Employing distortion tools for scene exploration

Distortion lenses can be considered as parameterizable objects which are typically employed to magnify a focused region while the context is scaled down. Their basic components are:

- the focal point, or region, at which the distortion is maximal,
- the area, or scope of influence, in which the lens creates an effect, and
- the manner in which this effect attenuates (drop-off function).

In the following, two applications of lens distortion are proposed which can assist in exploring a 3D visualization [SCS04].

The first application serves to adjust certain parameters, which can be compared with the functionality of traditional sliders. But there are two fundamental differences: (1) sliders are in general one-dimensional controllers whereas distortion lenses can be two- or three-dimensional, and (2) sliders adjust attributes linearly whereas the effects caused by distortion lenses depend on the drop-off function applied.

In this approach, a distortion lens is attached to a particular object of the visualization so that the lens moves with the object or remains at a place when the object pauses. In addition, the lens which is two-dimensional covers the region the object takes when it is projected onto the screen. When the user now moves the mouse cursor within the lens' scope of influence, certain parameters associated with the particular scene object can be adjusted. That is, moving the cursor towards the focal point yields a maximum value, moving it towards the peripheral region decreases the value.

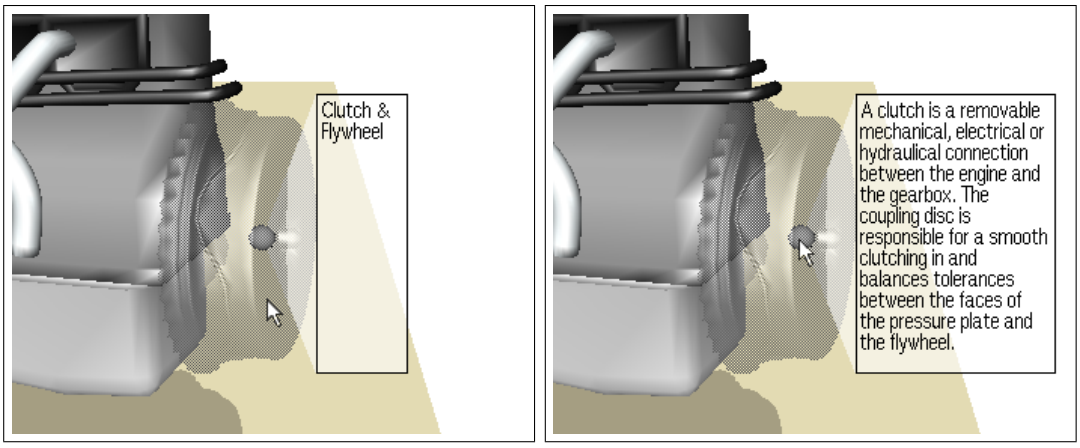


Figure 6.9: *Employing distortion for text display. The level of distortion affects the size of the annotation box and text scrolling.*

Figure 6.9, for example, shows how a user can control the extent to which the annotation box grows towards the image boundary when lens distortion is employed. To this end, a scene object becomes transparent enough to reveal its centroid (focal point rendered as a sphere) once it is reached by the mouse cursor. Selecting the object with a mouse click starts to reveal the annotation. As the cursor is moved towards the centroid, the annotation expands. A second click freezes the annotation's current state. Now, moving the cursor will scroll the text if it does not entirely fit into the annotation box. When the cursor leaves the object, it is no longer rendered as transparent. This technique is an example for

indirectly controlling two parameters—annotation extent and text scrolling—by moving the mouse cursor within the same small region.

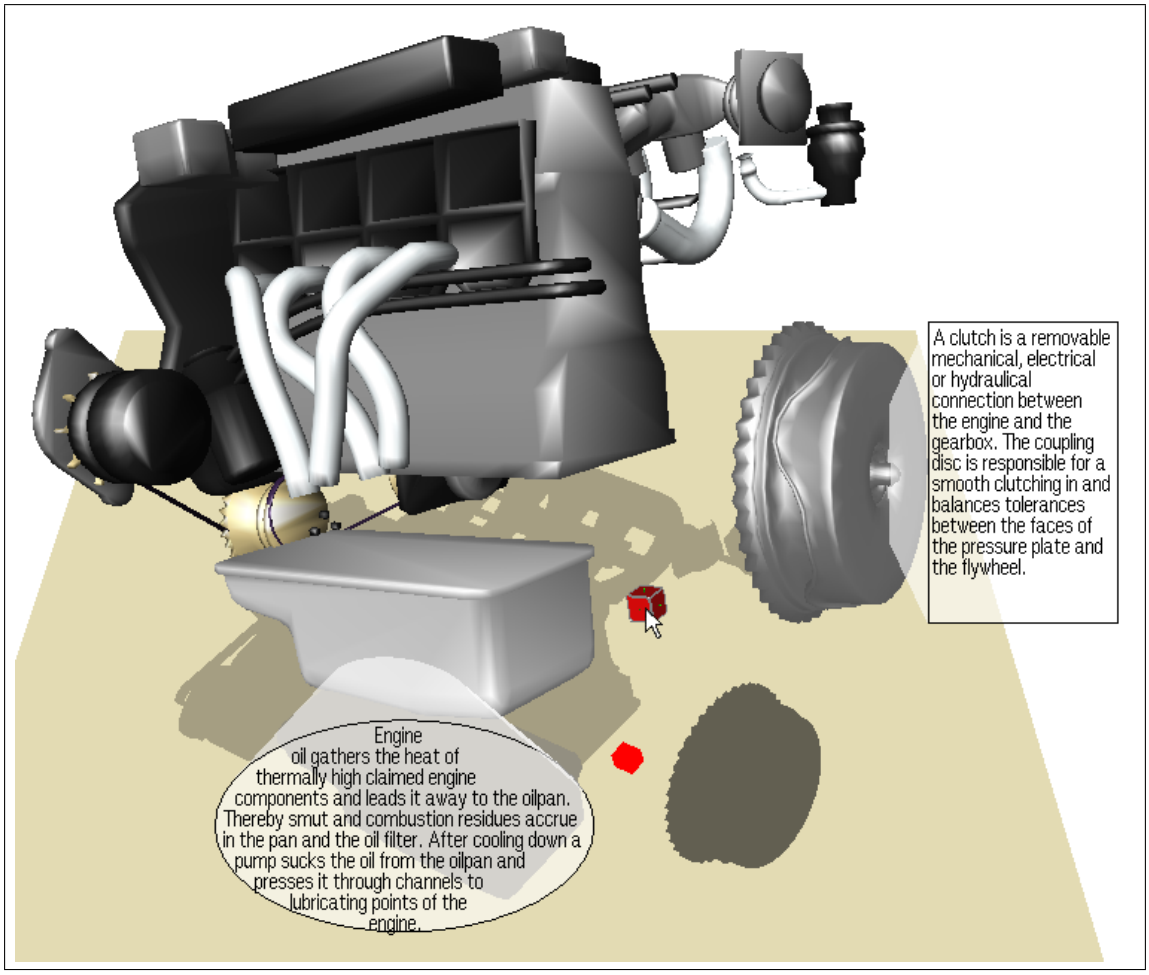


Figure 6.10: *Employing distortion for exploration tasks. A 3D lens (probe) is moved through the scene and creates an explosion diagram.*

The second application temporarily affects the composition of the visualization. Here, the lens serves as a probe which can be moved through the scene to reveal occluded scene objects and to shed light on the scene structure (see Fig. 6.10). In this way, the user needs to handle only one object—the probe, which spares to select and reposition every single scene object involved.

In contrast to the first application, the lens (probe) is independent from particular scene objects, and it operates in three dimensions. It still has a focal point and a certain scope of influence. A user can control the probe with a cube which moves with mouse repositioning in the plane associated with the face selected. As the probe is moved into the scene, the line of sight between the probe and the user is kept clear. To this end, the probe affects

scene objects by pushing them apart, which is illustrated in Figure 6.10. The image also shows that annotations attached to scene objects remain with their objects when they are moved. As soon as objects leave the scope of the probe, they are moved back to their initial positions.

6.3.4 Evaluation of Techniques that Link 3D Data and Text

Several techniques exist that combine 3D scene and text visualization with the aim of enabling unhindered scene exploration despite text display. The effectiveness of a selection of those techniques has been studied in the course of this work ([SCS05] provides a detailed description). The study, for which 36 voluntary participants were recruited from the local computer science department, was divided in two parts: Part 1 focused the clarity of correlations between scene components and associated texts, and Part 2 should shed light on whether text would remain comprehensible during a scene exploration.

Hypotheses

During the study, 3D model components were linked with text using varying techniques. For some scenarios, additional hints were provided which were assumed to facilitate finding the correlation between model and text. It was further assumed that the distance between model and text would be of importance: the shorter the distance, the faster the detection of correlations. Another assumption was that the composition of the model would be of influence: models of compact shapes and with intersecting components would hamper correlation detection. Regarding readability, regular text alignment, no interferences between model and text, and sufficient space for text display were assumed to enhance reading and understanding the text presented.

Description of evaluated settings and models

The software which was used during the study enabled a user to explore a 3D scene interactively. This means arbitrary viewpoints could be chosen and information from displayed texts could be gathered. The two basic components of a 3D scene are a model (arrangement of 3D geometric objects) and a setting (manner by which textual information is correlated to the objects). During the study, models and settings were varied and each model-setting combination was shown equally often.

Altogether, three tasks were designed and six different settings were selected for evaluation. Settings I-III were used during Task 1 and Task 3.

- ***Setting I—Object-attached labels.*** Annotations are directly connected to their respective scene objects (Fig. 6.11(a)). As described in Section 6.3.3, a translucent polygon, intersecting with the scene object, provides a visual link between annotation and object. During the scene exploration, annotations remain attached to their associated scene objects.

- ***Setting II—Shadow annotations***. Annotations are located within a shadow plane [CSRS03]. The region within which the text can be positioned results from projecting a scene object onto the plane (Fig. 6.11(b)). Due to potential occlusions by scene objects, annotations can be temporarily displayed in the foreground at will. The space available for text can only be enlarged by applying a zoom operation which yields scene objects and their projections to increase.

- ***Setting III—Separated annotations***. In contrast to the previous settings, the text is shown within a separate window (Fig. 6.11(c)). To enable identifying individual objects, clear textual descriptions of all scene objects are provided.

To find out whether what is perhaps the most common method of labeling—attaching the text to its associated object by means of a simple line—can be improved with the addition of visual hints, Settings A-C were evaluated during Task 2.

- ***Setting A***. Object and text are linked with a line (Fig. 6.12(a)). There are no further hints which can help to find a correlation between scene object and annotation. The text remains at a static position during scene exploration. Only the end point of the line, which is located inside the corresponding scene object, moves with the object.

- ***Setting B***. In addition to the line that connects scene object and annotation, the respective projection of the annotated scene object (object's shadow) is highlighted using a gray tone that differs from the general shadow color (Fig. 6.12(b)).

- ***Setting C***. The only difference to *Setting B* is that the shadow of the annotated object is highlighted using the diffuse color of the corresponding scene object. The shadow color as well as the position of the shadow relative to the respective scene object can thus support locating the annotated scene object (Fig. 6.12(c)).

The models created for the study were composed of arbitrary geometric objects. Basically, two reasons were deciding: (1) previous knowledge should not aid participants when objects were to be found on the basis of descriptions, which could be the case when real-world models would have been used, and (2) each model was constructed to consciously stress specific aspects to be analyzed during the study.

Description of tasks

After an introduction was given, each participant completed questionnaires and performed three tasks. For each task, three different settings were used. The tasks were designed to explore problems that can occur when textual information is integrated into a scene. These include, for example, unclear correlation between scene object and text, scene occlusions, or other interferences during a scene exploration.

- ***Task 1—Object-text correlations***. Using Settings I-III, participants were asked to find the correlations between scene objects and annotations. Specifically, a participant was asked to identify the scene object to which the annotation currently

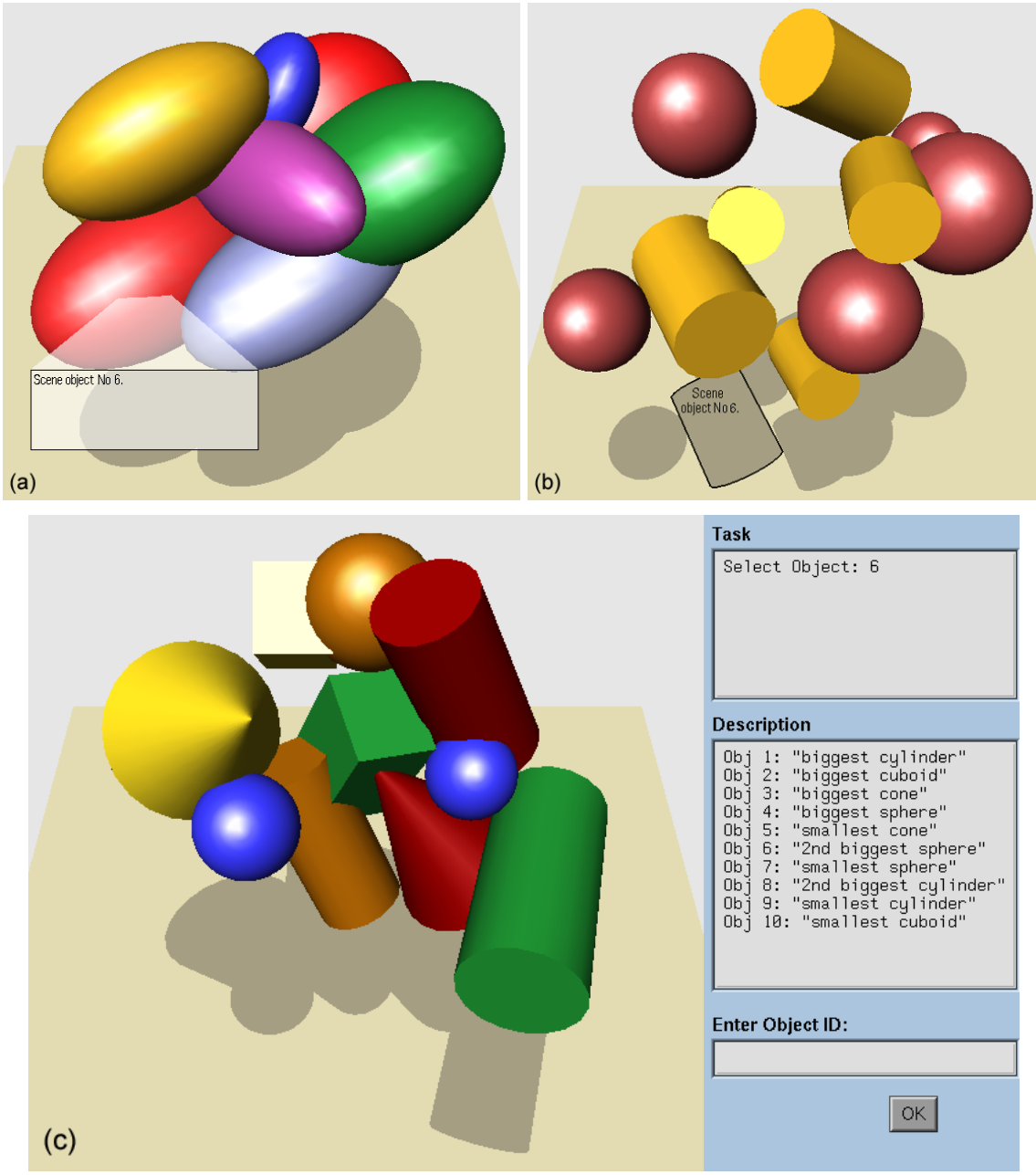


Figure 6.11: *Settings with varying text display methods in use. (a) Setting I: annotations are directly attached to scene objects; (b) Setting II: annotations are positioned within the objects' shadows; (c) Setting III: 3D render area and text area are separated.*

displayed referred, by selecting that object. For that purpose, eight annotations were shown for each setting. A new annotation appeared every time the participant had completed the subtask of selecting a scene object. In Settings I and II, the annota-

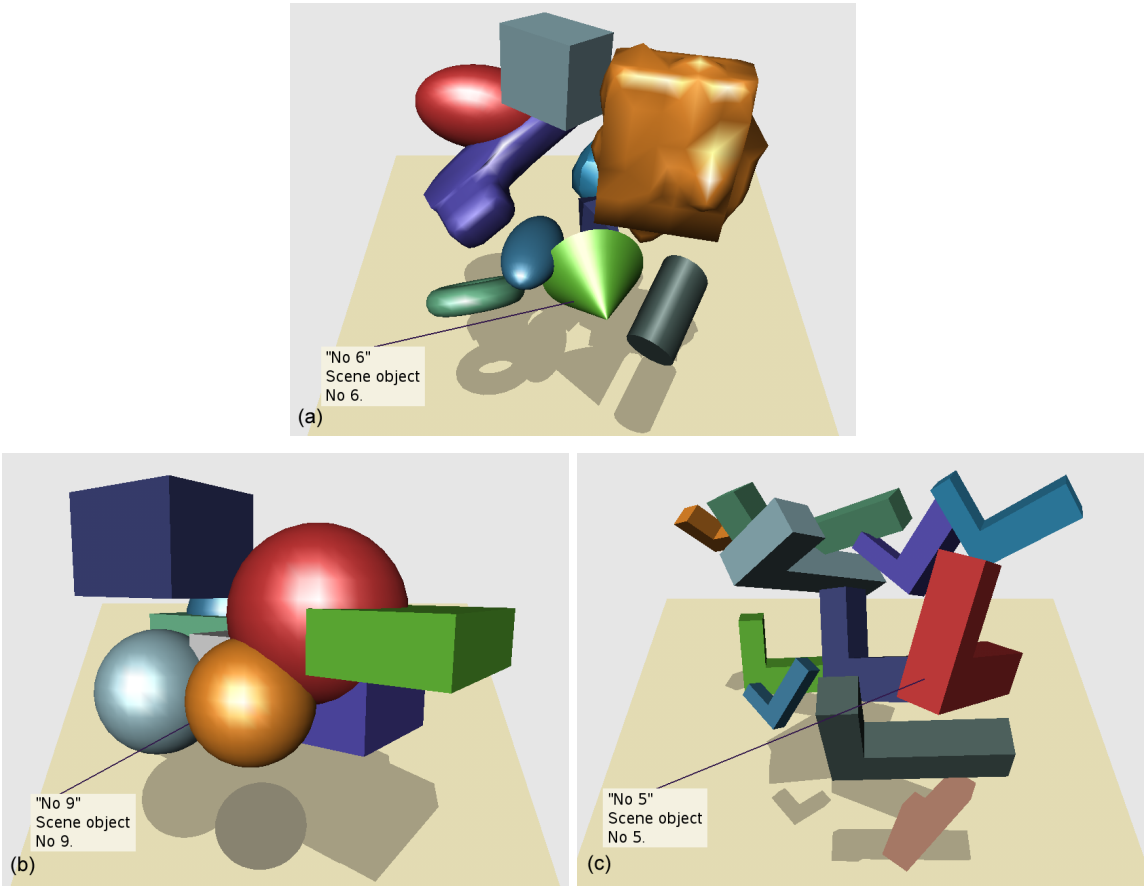


Figure 6.12: *All settings use the same simple "object-line-label" technique, but they differ in the manner of providing additional visual hints. (a) Setting A: no additional hints are provided; (b) Setting B: the shadow of the object is highlighted using a gray tone; (c) Setting C: the diffuse color of the respective scene object is used for highlighting.*

tions were displayed together with the rendered model within the same window (see Fig. 6.11(a) and (b)). Setting III differed from the other settings in that scene object descriptions were shown in a separate area on the right hand side. Instead of integrating eight annotations into the scene render area, successively the descriptions of eight objects were shown in the annotation area. The participant found the objects being referred to by their descriptions (see Fig. 6.11(c)).

- ***Task 2—Object-text correlations with additional hints.*** This task was designed to discover whether there are benefits to be gained when additional hints are included with simple labels attached by straight lines. The task used Settings A-C (see Fig. 6.12). The basic task was the same as Task 1. Participants were asked to find the correlations between the scene objects and the annotations. Again, successively eight annotations appeared which should be assigned to the corresponding scene objects.

- ***Task 3—Task-based scene exploration.*** This task was designed to explore reading and understanding the displayed text information. The task itself was displayed to the participant in the text annotation of the setting that was in use (Setting I, II, or III). It proceeded as follows: After an annotation—attached to an arbitrary scene object—was displayed, the participant read the complete text. When the participant found and selected the described scene object, a second annotation appeared that displayed the ID of the selected object. At this point, both annotations were integrated into the scene and remained visible during interaction. The last step was to enter the found object's ID into the first annotation which changed its task description text to "Enter Object ID here:" once it was selected. This procedure was repeated 5 times.

Summary of results and discussion

The parameters which were evaluated are: the time a participant needed to complete a task, the mistakes that occurred during performing a task, the comments which were given by a participant, and the observations which were made.

For ***Task 1***, Setting III was most time-consuming (followed by Setting II and Setting I), in particular, when the model had evenly shaped objects that differed only in size. This was probably due to descriptions such as "2nd biggest sphere" or "smallest cylinder" with which the objects should be found. The type of model was also influencing in Settings I and II. Here, compact models with intersecting objects caused more problems than models with more wide-spaced objects. But *Analysis of variance (ANOVA)* revealed that the type of model had less effect on the time than the type of setting ($F_{model} = 3.47$, $F_{setting} = 10.39$). Regarding the mistakes participants made, the number of incorrect object selections correlated with the time durations. The more time a participant needed to perform a particular task, the higher was the number of false selections. Table 6.1(left) shows how participants rated the settings after performing this task. 70 % favored Setting I, whereas 50 % rated Setting III to be the least qualified for this task.

For ***Task 2***, Setting C was the setting with which participants performed the task fastest (followed by Setting B and Setting A). 75 % of the participants stated explicitly that highlighted shadows facilitated the search. The number of false object selections, which was significantly lower for Settings B and C than for Setting A, confirms this estimation.

The strategies participants followed during this task differed. In Setting C, some of the participants were quite fast as they simply compared scene object colors with the highlighted shadow color before they made their choice. However, other participants did not adapt their strategy to the type of setting. Some of them (14 %) even did not consider the highlighted shadows or stated that the shadows were not very helpful. It still has to be studied whether a longer practice period would yield other results.

Task 3, which was the most complex and time-consuming task, yielded considerable variations in the times measured. Independently from the type of model, Setting II was the setting with which participants needed the most time to perform the task (nearly twice as much time as with the other settings). There were basically two reasons: (1) the text did

Setting	1st place	2nd	3rd
I	70 %	22 %	8 %
II	11 %	47 %	42 %
III	19 %	31 %	50 %

Setting	1st place	2nd	3rd
I	36 %	64 %	0 %
II	0 %	3 %	97 %
III	64 %	33 %	3 %

Table 6.1: *Ratings for Settings I-III: after Task 1 (left) and Task 3 (right).*

not always fit completely into its assigned space, which caused participants to zoom into the scene to enlarge the text area and (2) annotations often overlapped. Since scene render area and text area were separated in Setting III, participants could not get disturbed by annotations during the scene exploration. However, the measured times in Setting I and Setting III did only differ marginally. This indicates that annotations integrated into the scene have not such a high impact on a task-based scene exploration as it was expected. The post-questionnaire in which participants were asked whether they felt disturbed by annotations confirmed these results. As regards mistakes, both the number of false object selections as well as the number of incorrectly entered object IDs were marginal.

Finally, participants were again asked to rank the settings (see Table 6.1(right)), which yielded interesting results. Now, a clear majority of 97 % rated Setting II to be the worst setting. Furthermore, a majority of 64 % favored Setting III, the setting which got the worst rating after Task I.

As a result, it can be stated that none of the techniques was perfect in all situations. However, each technique has specific features that make it particularly applicable for certain tasks. Techniques, for example, that aim at placing short text labels nearby their corresponding scene objects enable a user to find those objects being referenced quickly. Setting I as well as an improved version of the "object-line-label" technique (e. g., the agent based approach by GÖTZELMANN et al. [GHS06]) are promising in this regard. When the goal, on the other hand, is to provide more detailed information, which typically results in more extensive text, the information should be located in a separate area that does not interfere with the rendition of the scene. Providing further visual hints, for example by emphasizing scene objects or by employing the shadow, can serve as helpful link between text and scene.

6.4 Summary and Discussion

Providing the descriptive metadata which is linked with a medium involves two challenging tasks: (1) the viewer's attention has to be drawn to those parts of the medium which are associated with the metadata sought and (2) the metadata has to be displayed adequately in case the user demands it.

Assisting a user in metadata search

Descriptive metadata is invisible by nature. Provided that a user knows there is metadata available, one strategy to gather parts of it can be to let the system show all regions which

have metadata associated. The user can then go through all metadata items, and if lucky, the appropriate information is among. But this procedure can be time-consuming and disappointing. Another strategy could be to go directly to those parts of the medium which are of interest and to ask the system if metadata is available. Unfortunately, this proceeding is unsuccessful if the metadata sought is associated with a different part.

But what is the right strategy and how can the system support the user?

A first step can be to highlight all regions with metadata associated, which gives an elementary overview. Providing *Metadata-Related Information* in this regard can assist the end-user in metadata search. Such information can include the type of metadata (e. g., text, image, or auditory data), its amount, and how much it correlates with what is searched. In addition, a rating given by other users, which is a quite common methodology in the World Wide Web, can also help to find the appropriate metadata. However, how effective such a proceeding is eventually has to be researched in the course of an experimental study.

Metadata display

When metadata is to be displayed, several issues can be influencing, the most important of which are:

- Is the visualization static or does it change over time? If it changes, the layout has to be updated permanently, which requires efficient layout algorithms. A static visualization, on the other hand, allows for iterative adaptations of the layout so that the final layout can conform better with demands which were specified.
- How much space is available? This concerns the size of the window in which the medium is displayed as well as the space within that window which is available because it is empty or because it contains information unimportant in the current context.
- Of what type is the metadata? In case it is text, its layout can be adapted with respect to the visualization or according to user demands. Certain patterns of symbols or images in general, in contrast, do not allow for adaptations to the extent text does.
- How long will the metadata be displayed? If it is only for a moment, which typically means the metadata has the viewer's focus, occlusion of the visualization by metadata and other layout aspects are secondary. If the metadata is to be displayed for a longer time period, an adequate layout is essential.

The user study which was conducted with regard to metadata display addressed one of the most challenging tasks: providing textual metadata in dynamic visualizations. Several requirements for appropriately integrating text in 3D scenes emerged during discussions with participants, from observations noted and from analyzing the results. The graphic object (line or polygon), for example, that links scene object and annotation should clearly indicate which scene object is referenced. In case of line use, some participants suggested

in this regard that the intersection between line and scene object should be marked. As regards text display, showing extensive text in a separate area was accepted by most participants. Also, a rectangular text box was favored over arbitrary polygonal shapes. When annotations changed their positions abruptly, some participants were confused, which indicates that annotations should change their positions smoothly.

Most of the hypotheses which were stated at the beginning of Section 6.3.4 were confirmed by the study.

Hypothesis 1: The integration of additional hints facilitates finding the correlation between model and text.

During the study, highlighted shadows in Settings B and C served as such hints. Setting C, which used colored shadows instead of gray tones, caused participants not only to select less scene objects incorrectly but it also caused participants to save considerable time (mean times: Setting A: 52.24 sec, Setting B: 49.34 sec, Setting C: 41.19 sec).

Hypothesis 2: The shorter the distance between model and text, the faster the detection of correlations.

This hypothesis was confirmed in case of Setting I in which text labels were directly attached to their corresponding scene objects. In case of Setting II, the composition of the model had significant influence. Even though the distance between model and text was shorter than in Setting III, participants spent longer time periods in correlation search when a compact model was shown. On the other hand, Setting II can hardly be compared with Setting III in this context because the settings use strongly differing techniques.

Hypothesis 3: Models of compact shapes and with intersecting components hamper correlation detection.

This strongly depends on whether visual hints (Settings I and II) or textual descriptions (Setting III) are provided to indicate model-text correlations. While correlation search was almost not affected by the composition of models when textual descriptions were given, the manner in which visual hints were provided could have an effect. For example, participants had difficulties when intersection markers between model and line were missing in case that simple lines were employed as visual hints while compact models were shown.

Hypothesis 4: Regular text alignment, no interferences between model and text, and sufficient space for text display enhance reading and understanding the text presented.

Sufficient space for text display and regular text alignment (e. g., in a rectangular box) enhanced reading and understanding of text, as Task 3 revealed. Setting II, for example, in which the area for text display was defined by the shape of the corresponding scene object required interaction for reading the whole text, which was annoying and time-consuming. Interferences in terms of text which overlapped the model, on the other hand, had less influence than expected. However, for reading extensive text, regular alignment and separate areas for text display were preferred.

7 Application Scenarios

The research at hand addresses illustrations that have dynamic contents. Such illustrations have their right to exist in a wide variety of applications. This is for example whenever there is demand for the display of information beyond that what a printout can offer which is limited due to its content being static. Aspects which were focused in this regard concerned techniques with which descriptive metadata can be appropriately stored and which can be employed to provide the metadata eventually to a viewer.

To demonstrate the validity of the proposed concepts, this chapter introduces and discusses a selection of application scenarios. In Section 7.1, a program is presented with which arbitrary digital images can be enriched with descriptive metadata or converted into pictorial dictionaries. Section 7.2 then introduces scenarios for images that have animated and controllable elements. Examples for augmenting 3D visualizations with descriptive metadata are addressed in Section 7.3.

7.1 Smage—Smart Images

In the age of prospering digital photography, annotating digital photographs becomes more and more important in every day life. Amongst others, annotations facilitate sharing stories about experiences, travels, friends, and family, which, according to Balabanović et al. [BCW00], ranks among the most common and enjoyable uses for photographs. In this regard, the authors address the particular application scenario of sending someone a set of photos which are accompanied by some commentary. Other authors also discuss the issue of adding annotations to make a funny comment or to add some knowledge (e. g., [KPC+99, SK00, RV04]). In all of these systems, the metadata is stored separately from the medium with which it is associated.

Specifying various types of metadata, assigning the metadata to certain image regions, storing the metadata as *Illustration Watermarks*, and retrieving the metadata included in an image are the basic features offered by a system called *Smage* which has been implemented in collaboration with Lothar Schlesier.

7.1.1 Program Features

Figure 7.1 shows snapshots of the program *Smage* that illustrate some of the features the program offers. In Figure 7.1(a), the program's main window is shown that contains an image which is currently being annotated. At the moment, three regions in the image are assigned varying types of metadata. The metadata itself can be managed by employing the overview window in Figure 7.1(b). It includes a dictionary entry (symbolized by the

character D) which was specified with the dialog in Figure 7.1(c), and two short texts the styles of which were selected using the dialog in Figure 7.1(d). Simple unstructured text is represented by the symbol T whereas other types of text are displayed the way they will appear during the exploration of the watermarked image. The basic steps involved to create an image with metadata encoded are illustrated in Figure 7.2.

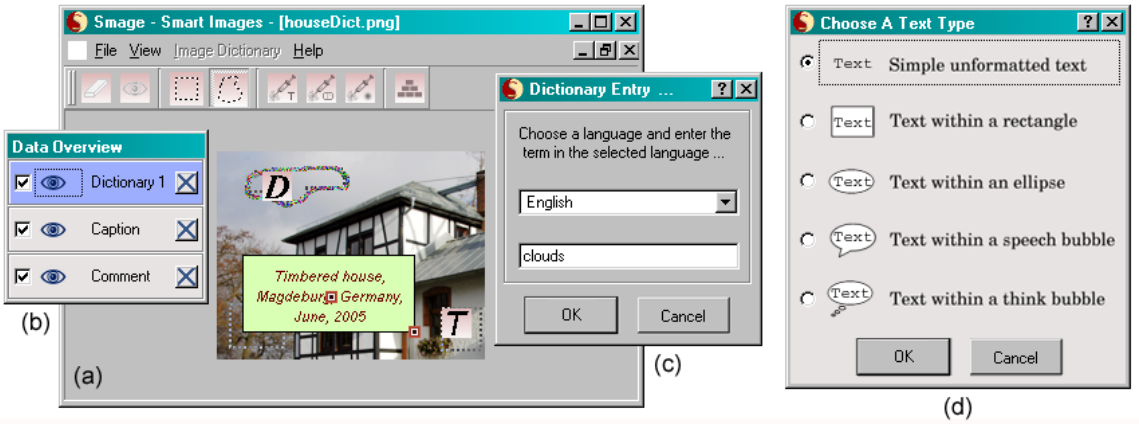


Figure 7.1: *Basic features which are provided by the software tool* Smage.

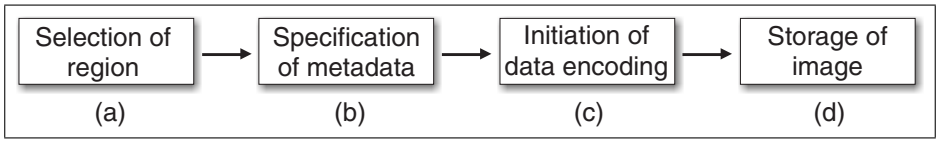


Figure 7.2: *The basic steps which are necessary, on user side, to create a watermarked image.*

The types of metadata which can be encoded are:

- ***Simple texts*** which will be shown within a scrollable box after decoding. This box then moves with the mouse cursor as long as the cursor is located inside of the watermarked region. The box can also be fixed at its current position, namely when the small button at the left bottom corner is clicked (see Fig. 7.4).
- ***Stylized texts*** which will appear in the form they were created. The text itself can be adjusted in terms of its font, color, size, etc. The shape that surrounds the text can vary in its color and contour. It can be circular, rectangular, or it can have the form of an adjustable speech balloon (see Fig. 7.1(d)).
- ***Dictionary entries*** that facilitate to learn how what is seen in the image is termed in foreign languages. For this purpose, the author of the image can first choose the preferred language and then enter the term that characterizes what is seen in the selected region. A dictionary file, which is part of the program, already includes

the translations of basic terms in several languages (e. g., English, French, German, Spanish, and Italian).

- ***Images*** can be encoded into the cover image. Prior to their insertion, those images are converted into indexed images, which reduces the required storage space while an acceptable image quality is maintained.

Once all metadata items have been specified and assigned to the corresponding image regions, the user can initiate the watermarking process. After watermarking, the coded image can be stored, which is the final step.

Besides the creation of watermarked images, the program allows to remove the metadata which is associated with a certain region and to explore images that already have *Illustration Watermarks* embedded.

```
<script src="SmagePlugin.js" type="text/javascript">
</script>
<script type="text/javascript">
<!-- document.writeln( smage( "house.png", 311, 269 ) ); //-->
</script>
<noscript>
  <object ID="AXSmage"
     CLASSID="CLSID:DF16845C-92CD-4AAB-A982-EB9840E74665" width="311" height="269">
    <param name="smage" value="house.png">
    <embed type="application/smage" smage="house.png" width="311" height="269">
  </object>
</noscript>
```

Figure 7.3: *Example source code which can be included in a* HTML *document to enable the exploration of watermarked images in Web browsers.*

Images encoded with *Illustration Watermarks* are primarily intended to be published on Web pages. For this reason, plugins for common Web browsers such as *Netscape*, *Mozilla*, or *Microsoft Internet Explorer* were implemented that enable end-users to explore watermarked images. The program *Smage* provides a function that generates the source code to be included in a Web page's source code. Figure 7.3 shows an example of a source code that would include the image file "house.png". This code could be even more compact. But ensuring compatibility with common Web browsers requires to add specific lines of code.

7.1.2 Implementation Aspects

The software is implemented in *C++* and includes the library *libSmage* which is responsible for encoding the metadata into selected image regions and for recovering the data. The user interface, image processing functions, and browser plugins were implemented using the *Qt* software library. Other essential third-party software libraries included are *libxml* and *zlib*.

With respect to the image creation steps depicted in Figure 7.2, the first two steps, which basically concern the user interface, were realized by employing *Qt* functionality.

Step (c) is enabled by the software implemented in *libSmage*. Provided that image regions were selected and the according metadata has been specified, *libSmage* does the following (compare with Sect. 4.2):

1. The specified metadata is converted into a byte stream. While simple text is already a stream of characters, the properties of stylized text (geometry, colors, font, position, etc.) must be converted into a representation which can be included in the byte stream. Images to be embedded are transformed into indexed images (color quantized), which yields a color table (array of RGB color values) and byte pointers into that table. Each metadata item is then appended to the byte stream. A number at the beginning of each item indicates how many of the following bytes are associated with the item.
2. After this, the byte stream is compressed using the *zlib*.
3. Finally, the compressed byte stream is encoded into the cover image. The ID-buffer, which was generated with respect to the regions a user had selected, defines the image regions into which the data is to be embedded using one of the watermarking techniques described in Chapter 4.

Step (d) completes the creation of a watermarked image. To this end, the user is asked to select a location at which the watermarked image is to be stored. During this step, the software verifies whether the stored image is still identical with the image before it had been compressed according to the specifications associated with the file format (e. g., *PNG* lossless compression).

7.1.3 Examples

A first example was introduced in Section 4.2.1. The cover image in the illustration in Figure 4.2 contains a rendered 3D model of a liver and a small icon which indicates that further information is available. One type of metadata encoded is a second image that contains a transparent rendition of the liver and hence, provides an insight into the model revealing excrescences. The second type of encoded data is text that describes what is depicted.

Figure 7.4 illustrates another application. It shows an example for annotating an arbitrary photograph which was taken with a digital camera. Besides information about place, time, and what is depicted, comments can add specific atmosphere to a picture. Figure 7.5 illustrates how exploring the picture can be supported by the *Meta-Previewer* (see Sect. 6.2.3) which has been implemented by Nguyen Dinh Quyen. This previewer provides *Metadata-Related Information* such as markers for watermarked regions, and information about the metadata's content, type, and size which is shown in a status line that continously changes the information presented.

Toyama et al. [TLRA03] had the idea of linking specific regions in a map with images that show how those places look like in real life. In Figure 7.6, a similar use case was emulated. All the photographs shown in Figure 7.6(right) were encoded into the cover image (the map) so that map and photographs merged to a single *PNG* image data file.

Figure 7.4: *Photograph which has its different types of information included as* Illustration Watermarks.

Figure 7.5: *Exploring the watermarked image of Fig. 7.4 using the* Meta-Previewer. *The rightmost person is focused and hence, magnified by the viewer. The remaining regions with metadata encoded are symbolized by small red dots.*

7.1.4 Future Directions

User interface design

Since *Smage* is a program which was developed to be employed by a wide variety of end-users, an easy-to-use and appealing user interface is of high importance. Its usability should

Figure 7.6: *Map with photographs encoded as* Illustration Watermarks.

hence be explored in a user study with all types of potential end-users involved.

Exploitation of existing software functionality

Many software tools exist that feature image processing which is rich in functionality. Functions which are of relevance with respect to *Smage* are, for example, assisted image region selection, image layer handling, and qualitative image downgrading. Such functionality could be employed for metadata processing and its assignment to image regions. In addition, including *Illustration Watermarking* in existing software tools would allow for both image processing and image annotation using one and the same program.

Integration of what digital cameras offer

Several digital cameras allow to add metadata such as location, date, or technical settings which were in use during photographing. Some of those cameras can also serve as voice recorders so that auditory metadata can be associated with a photograph. These are types of metadata which could be automatically inserted as *Illustration Watermarks* into the photographs currently taken.

7.2 Images with Dynamic Contents

In the foregoing application scenarios, metadata content was static. It could either be displayed or not. This section introduces scenarios in which metadata content not only appears at will but in which it also changes over time. All the information required for such application scenarios is stored as *Illustration Watermarks*.

Figure 7.7 shows an illustration of the system components. The cover image serves two purposes: it is the *Background Layer* of the visualization and it has the metadata encoded. In addition to the components to be displayed (display data), the metadata also contains information and rules about how the content is to be animated and how it can be affected

by the user (animation data). The animation data comprises information such as paths along which display data items will move, time durations for certain movements, or which display data items can be affected by what kinds of user input. Once the metadata was extracted, the content manager determines the display of the metadata content with regard to user input (e. g., mouse, keyboard, etc.) and animation data.

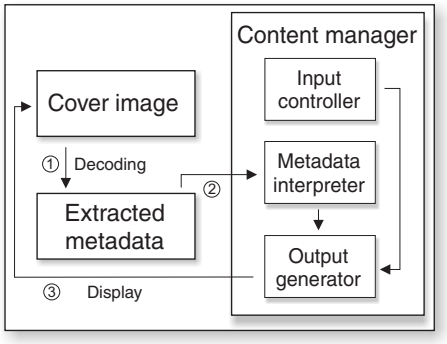


Figure 7.7: *Illustration of system components that enable the display and control of dynamic contents in images.*

In the following, two potential scenarios are described: an illustration which is enhanced by dynamic content and a simple computer game.

7.2.1 Dynamic Illustrations

The first example is depicted in Figure 7.8(left). It shows an illustration of the human heart and its basic blood vessels. The dynamic components are small arrows that move along the paths blood takes through the heart. There are basically two advantages of animated arrows over arrows that remain at assigned positions: (1) movement is one of the most efficient ways for attracting attention, which allows for faster arrow detection, and (2) even though static arrows would point into the directions of movement, they would not characterize the exact paths; dynamic arrows are more exact in this regard.

The data items which are embedded as *Illustration Watermarks* are an image of the arrow as well as information concerning display and animation. The latter includes the number of image (arrow) copies, and for each copy a start position, a series of points that define the path of movement, and the time duration the image will take to travel the whole path. During movement, an arrow's orientation is permanently aligned so that it always points into the direction of movement.

7.2.2 Gameable Images

The second example is a simple computer game which was developed in collaboration with Sadiq Sowdagar. This game has both animated components and components which can be controlled by the user. Figure 7.8(right) shows a snapshot of the game. The cover image, which is the game background, is divided into grassland, sky, and a tree. These

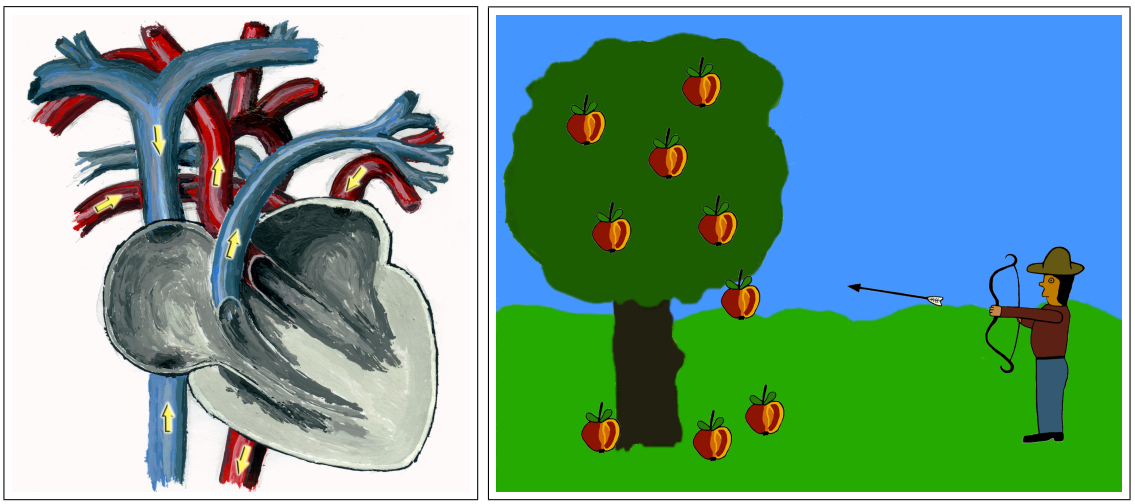


Figure 7.8: *Examples for images with dynamic contents. Left: Illustration of the heart with arrows that change their positions (painted with [Lip00] as master). Right: Image that provides the data for a simple computer game.*

components are static. The animated components are the apples and the arrow. The archer can be controlled by the user. The basic goal of the game is to hit each apple with the arrow before the apple comes off. The user can adjust the direction along which arrows are shot and trigger those shots.

In this particular example, three images are encoded into the cover image: one apple, the arrow, and the archer. Similar to the illustrative arrows in the previous example, information about number of copies, placement, path, and time duration is necessary with regard to the apple. The movement of the arrow follows physical law once it has been shot. As regards the archer, it must be defined that certain keys cause the character to rotate and the arrow to get shot. These are information items included in the metadata and handled by the metadata interpreter. Aspects such as computing the arrow's path, displaying continuous arrow movement, or determining whether an apple was hit by the arrow are processed by the output generator (see Fig. 7.7).

7.2.3 Future Directions

The introduced examples are specific use cases of two different domains: education and entertainment. However, both examples utilize same components (dynamic images, instructions, and rules), which indicates that any type of image with dynamic content could be created using a same framework. Such a framework needs standards for instructions and rules as well as a compact form for their storage.

In addition, an appropriate user interface is required that enables the author to create such images intuitively and with little effort. In this regard, it has to be explored what are the most important features an author would like to integrate and what are those features an author avoids and which would hence unnecessarily increase the complexity of the system.

7.3 Application Scenarios in 3D

3D model data is often enriched with data that provides further information. This information can be detailed knowledge that complements the visualization, but it can also define certain behaviors and constraints of particular 3D model components. The inclusion of such information as *Illustration Watermarks* provides an opportunity for a compact storage of model and information and their close interlinkage.

7.3.1 Constraints and Behavior in 3D Environments

Ellen Do's researching group works in the field of 3D environments which are composed of what they call *Smart Objects* [EGD01]. For those objects, certain constraints and behaviors are defined that influence the way a user can interact with them. A potential application for *Illustration Watermarks* could be to encode those object properties into the objects themselves.

Another domain in which *Illustration Watermarking* could be employed are 3D computer games which enable a user to personalize the characters involved. Such game environments empower a user to assign specific abilities and characteristics to individual characters. Stored as *Illustration Watermarks*, characters and their properties would form entities, which could facilitate the interchange of personalized characters between users.

7.3.2 Descriptive Metadata for 3D Model Data

Another application scenario concerns the storage of descriptive metadata. Götzelmann et al. [GHS06] introduced a promising technique with which different types of text can be integrated in 3D scenes (see Fig. 7.9). Provided that the model is segmented into individual objects, an ID-buffer and a distance field, which are represented as 2D images, are generated. Employing those images and some additional information, internal and external labels as well as more extensive text boxes can be adequately arranged (compare with Sect. 6.3). The additional information includes, for example, IDs that correspond with the ID-buffer colors, text (labels and their shortcuts or longer descriptions that will appear in text boxes), and information about text grouping and which text items are more important than others within a certain context. If this additional information would be stored as *Illustration Watermarks*, the model would provide the whole information which is necessary to generate visualizations such as the illustration in Figure 7.9.

7.3.3 Future Directions

Particularly the inclusion of descriptive metadata deserves closer attention. Providing all the information necessary to generate the layout of an illustration (see Fig. 7.9) automatically is one potential scenario for future work. However, what such automatic layout generation cannot offer is to reconstruct a customized layout designed by a particular author which features aspects such as individual text selection and positioning as well as a

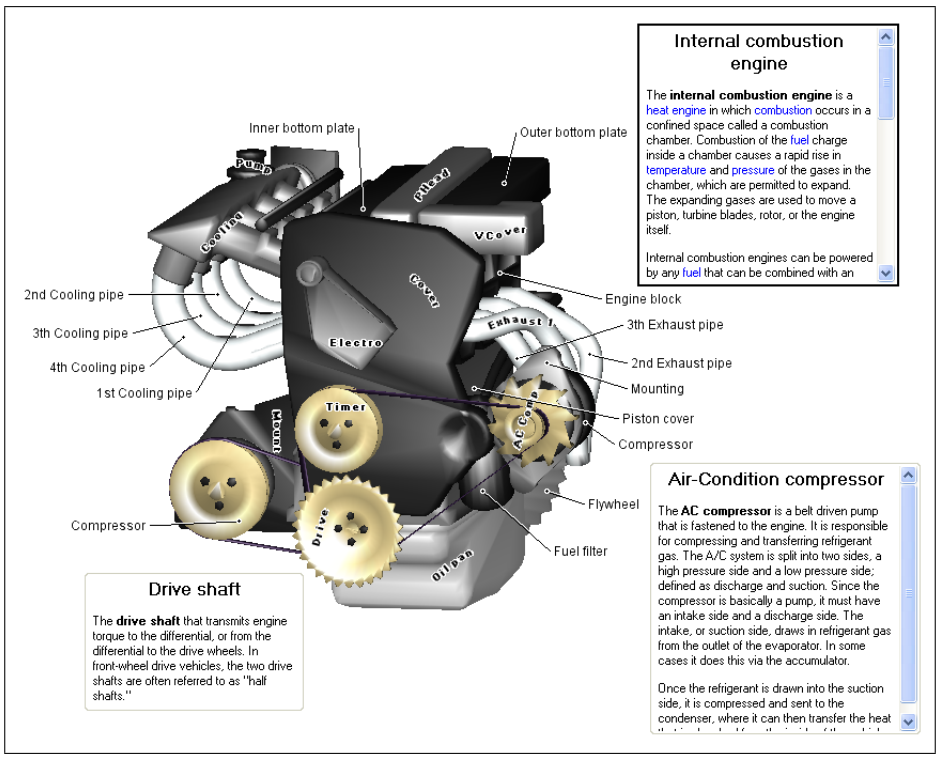


Figure 7.9: *A model of an engine which is enriched with different types of annotations (courtesy of* Timo Götzelmann*).*

specific view. Including only a few additional parameters in the watermark stream would allow for reconstructing customized layouts.

Also, maintaining an adequate layout during the exploration of a model could be further enhanced by defining certain behaviors and constraints. For example, each model component could be linked with a number of properties such as its importance within a particular context, the manner in which it is to be rendered when other components are focused, its sphere of influence, or the optimal position of its centroid. In addition, a certain space around or a certain region of a model component could be defined which must not be occluded by integrated metadata.

8 Conclusion

The picture which appears when a digital medium is visualized on a display device carries a certain amount of information which, according to a well-known proverb, can be worth more than a thousand words. But where are these words represented? In [SS97], the authors ask ironically whether these words are represented within the machine. Besides words which are directly linked with picture elements (e. g., "the tree beside the house") and words which are formed in the mind when previous knowledge gets involved (e. g., "the tree has green leaves, it must be summer"), some words can indeed be represented in the machine. These are those words which facilitate communicating the picture's real message or which provide additional knowledge concerning the picture's content that cannot be extracted from the present visualization. In this research, such words are referred to as *descriptive metadata*.

Providing descriptive metadata to the end-user requires software that can store, handle, and display the data. Existing software which can be used to store a digital medium and its descriptive metadata is manifold and associated with assets and drawbacks. Software tools, for example, which provide a wide range of functionalities typically require specific software for handling their data files due to lacking standardizations. Software specifications, on the other hand, such as those based on *XML* are standardized but do not allow for assigning the diversity of metadata types and interaction functionalities to the extent that software tools facilitate. Besides the internal representation of a digital medium and its descriptive metadata, the visual representation of the metadata is another important aspect that needs attention. Since the limitation of display space typically requires the metadata to be hidden, the challenge is to provide the metadata adequately and in its entirety to the end-user. Hence, appropriate techniques for both guiding the user to those regions which have the metadata associated the user currently seeks and integrating the metadata into the actual visualization are necessary.

In this research, techniques were analyzed which can be employed to store a medium—termed as *Background Layer*—and the data which is necessary to provide its associated descriptive metadata—termed as *Illustration Layer*. Digital watermarking is one of those techniques, even though its main application is in the field of copyright protection of digital media. However, accordingly parameterized, it is well-suited to store *Background Layer*, *Illustration Layer*, and the way they are linked with each other. Several *Illustration Watermarking* techniques for differing types of media were proposed and evaluated in this regard. The second part of this research dealt with techniques which can assist an end-user in exploring *Illustration Layer* data. Providing *Metadata-Related Information* in this context was discussed as a methodology for supporting the user in gathering the information sought. With respect to the particular scenario of integrating textual information in

interactive 3D visualizations, requirements were specified and techniques were proposed and evaluated.

8.1 Summary of Contributions

The research at hand addressed selective aspects of the huge area of the internal and visual representation of data. In this context, it analyzed and discussed existing methods, introduced new concepts and points of view, and it evaluated several proposed techniques.

8.1.1 A Medium's Basic Components and Techniques for their Storage

A digital medium for which an end-user can demand information in addition to what is currently displayed consists of two components: the *Background Layer* and the *Illustration Layer*. Storing the *Illustration Layer*, which is the component that allows for interaction, is associated with a number of challenges which were discussed in Chapter 2. *Illustration Layer* data storage comprises, for example, the inclusion of varying types of metadata, the appropriate representation of links between *Background* and *Illustration Layer* components, and descriptions of what can be triggered by what action.

In this regard, existing storage techniques were analyzed as to whether they can be employed to represent *Illustration Layer* data. To this end, the techniques were classified in techniques that store *Background* and *Illustration Layer* components in individual data files (*Separated Data Storage*), techniques that integrate *Background* and *Illustration Layer* components within the same data file (*Integrated Data Storage*), and techniques that use elements of both *Separated* and *Integrated Data Storage* (*Hybrid Data Storage*). Evaluating the techniques revealed that each technique had assets and drawbacks. Characteristics which were considered concerned for instance the variety of metadata types which can be assigned, the manner in which *Background* and *Illustration Layer* components can be linked, or the handling of the data files which are generated as to whether specific software is required or whether standard software can be used.

8.1.2 Illustration Watermarking for Illustration Layer Storage

One of the main issues emphasized in this work concerned the exploitation of digital watermarking techniques for data storage. In this regard, the concept of *Illustration Watermarking* was introduced. To store *Illustration Layer* data, existing watermarking techniques were adapted and novel concepts were developed which can be applied to raster and vector graphics as well as to 3D model data (see Chapters 3 and 4).

In the particular field of *Illustration Watermarking* for raster graphics, including *Capacity Maps* during the watermarking was introduced as a new methodology by which potential data rates which can be encoded in a pixel are suggested. These *Capacity Maps* result from analyzing image characteristics which can be exploited for watermarking. Conducting a user study should shed light on how efficient techniques that make use of *Capacity Maps* are in comparison with a traditional technique (see Chapter 5). Besides exploring differences

between the techniques, the maximum amount of data which can be invisibly embedded in an image region and the effects of watermarking varying image textures should be investigated. The evaluation revealed that the standard *LSB* method yields the best results when it includes *Capacity Map* suggestions. It also revealed that a minimum of 5, but up to 8 bits of the watermark data stream can be inserted into a uniformly textured image without the introduced changes being detected by a viewer.

To provide a selection of techniques which were developed to end-users, a software tool which is called *Smage* was implemented. Using this tool, descriptive metadata can be assigned to arbitrary image regions and encoded using *Illustration Watermarking*. Chapter 7 described several potential application scenarios in this regard, for instance the assignment of comments, funny notes, or images as well as the conversion of images into pictorial dictionaries. Another feature discussed is to add elements to an image which change their appearances over time and whose behaviors can be affected by end-users.

8.1.3 Visual Representation of Descriptive Metadata

Chapter 6 addressed aspects concerning the challenge of how 2D- and 3D visualizations which have metadata associated can be appropriately explored. Since the metadata is typically not visible in the picture's original view, information must be provided that signalizes which parts of the visualization are linked with metadata. In addition, providing information about the metadata itself can help to find the metadata which is sought more efficiently. Two approaches were proposed that provide such location- and content-related information, which has been termed *Metadata-Related Information*, to the viewer. The first is *Informative Cursor* which is a dynamic mouse cursor that changes its appearance to provide *Metadata-Related Information*. The second technique, which is called *Meta-Previewer*, employs a small window which shows a focus+context view with markers for augmented regions integrated that is permanently adapted according to the current mouse cursor position.

The second part of this chapter focused the challenging task of providing textual metadata in interactive 3D visualizations. Subsequent to giving an overview of the diversity of existing techniques, a novel approach was presented that attaches text labels directly to their associated 3D objects. In doing so, both shapes and extents of the labels can be adapted so that object linkage is clear and scene occlusion is limited. To facilitate scene exploration and to let the viewer control the labels, distortion lens functionality was exploited. This allows a viewer to create explosion diagrams of the scene, to change shapes of annotations, or to scroll the text displayed, with minimum effort.

To evaluate the usability of three classes of techniques for providing textual metadata in 3D visualizations, a user study was conducted. The study primarily aimed at exploring whether the correlation between a scene object and its text could be clearly identified and whether the text displayed was readable and understandable in spite of changing views and positions. Analyzing the study revealed that none of the techniques was perfect in all situations. However, each technique featured specific characteristics which were particularly applicable for certain tasks. Combining the techniques and applying them

with respect to task and metadata properties can yield an effective data visualization and exploration system.

8.2 Discussion and Future Directions

The methodology of embedding descriptive metadata in media using digital watermarking techniques, which is referred to as *Illustration Watermarking* in this research, and methodologies such as including metadata in certain sections provided by a few graphic file formats or using appropriate software tools (discussed in Sect. 2.2.2) can coexist. All these approaches have assets and drawbacks. *Illustration Watermarking* basically benefits from generating a close connection between a medium and its metadata and from allowing to assign any kind of digital data. Its most significant drawbacks, on the other hand, are the limited amount of data which can be embedded and the risk of metadata loss when the medium is modified.

The limited amount of data which can be encoded and the risk of metadata loss are two aspects which are intimately connected with digital watermarking. These problems are hard to overcome as long as watermark data is encoded by modifying the data of which the original medium consists. On the other hand, representing metadata as data structures based on *XML* and including it in data file headers sets limitations in terms of the diversity of metadata which can be assigned. However, using this methodology the amount of metadata is independent of the original medium's size and the metadata itself is not affected by potential operations applied to the medium. A solution which incorporates the advantages of both digital watermarking and metadata inclusion in certain file header sections could be to introduce an *Illustration Channel*. In case of raster graphics, the *Illustration Channel* could provide for each pixel a certain number of bits which can represent *Illustration Layer* data (comparable to the *Alpha Channel* that holds transparency information). Problems which have to be solved in this regard concern the manner in which the bits which are represented pixel by pixel can be appropriately merged so that they form the whole message which was assigned to a region of pixels. Also, since each region is typically associated with an individual amount of metadata, pixels can have varying bit rates assigned, which requires specific handling. Faced with numerous challenges, alternative approaches such as storing the metadata in *hash maps* (region-metadata pairs) have to be evaluated, as well.

Besides the objective of finding a standard for *Illustration Layer* data storage, another aspect to be dealt with in the future concerns classifying and structuring *Illustration Layer* data. Analyzing end-user tools which empower users to create and explore augmented media and investigating the functionalities desired in this regard will most-likely reveal that there is a limited set of techniques which are consistently in use. Examples are actions (e. g., selection with the mouse cursor), what they cause (e. g., display of a certain type of metadata), and in which way they cause it (e. g., methods for animating contents). Defining a standard for expressing and storing those functionalities would be a huge step towards a standardized multimedia data file format.

8.3 Concluding Remarks

Irrespective of the technique that is eventually used, *Illustration Watermarking* serves the purpose of augmenting a digital medium with descriptive metadata. This is necessary because humans often need explanations to understand what is really meant or hints which support perceiving pictures from new perspectives. An observation made by Œuvres Complétes de Voltaire [dV38] shows in this regard that this not only applies to digital media:

> Voltaire said that he once looked through a window at a building in the distance. He noticed several small statues that crowned the roof of that building. Suddenly, one of those statues moved, and from that moment on, he saw human beings instead of small statues.

A potential *Augmented Reality* scenario: the attribute "movement" assigned as descriptive metadata to a representation of a real object by employing an *Illustration Watermarking* technique ...

References

[AA99] Gennady L. Andrienko and Natalia V. Andrienko. Interactive Maps for Visual Data Exploration. *International Journal of Geographical Information Science*, 13(4):355–374, June 1999.

[AD86] Ikram E. Abdou and Nicolas J. Dusaussoy. Survey of Image Quality Measurements. In *Proc. of the Fall Joint Computer Conference*, pages 71–78. IEEE Computer Society Press, November 1986.

[ADME02] Nicolas Aspert, Elisa Drelie, Yannick Maret, and Touradj Ebrahimi. Steganography for Three-Dimensional Polygonal Meshes. In *Proc. of the SPIE 47th Annual Meeting, Applications of Digital Image Processing XXV*, volume 4790, pages 211–219. SPIE—The International Society for Optical Engineering, November 2002.

[Ado92] Adobe Developers Association. *TIFF 6.0 Specification*, June 1992.

[Ado99] Adobe Systems, Inc. *PostScript Language Reference*. Addison-Wesley Publishing Company, Inc., third edition, March 1999.

[Ala00] Adnan M. Alattar. Smart Images Using Digimarc's Watermarking Technology. In *Proc. of SPIE Vol. 3971, Security and Watermarking of Multimedia Contents II*, pages 264–273, May 2000.

[AP98] Ross J. Anderson and Fabien A. P. Petitcolas. On The Limits of Steganography. *IEEE Journal of Selected Areas in Communications*, 16(4):474–481, May 1998.

[AT04] Byron Antoniou and Lysandros Tsoulos. Converting Raster Images to XML and SVG. In *Proc. of the 3rd Annual Conference on Scalable Vector Graphics (SVG Open 2004)*, September 2004.

[Bar97] Lyn Bartram. Perceptual and Interpretative Properties of Motion for Information Visualization. In *Proc. of the 1997 Workshop on New Paradigms in Information Visualization and Manipulation*, pages 3–7. ACM Press, November 1997.

[BBP01] Mauro Barni, Franco Bartolini, and Alessandro Piva. Improved Wavelet-Based Watermarking through Pixel-Wise Masking. *IEEE Transactions on Image Processing*, 10(5):783–791, May 2001.

[BBRP99] Mauro Barni, Franco Bartolini, Alessia De Rosa, and Alessandro Piva. Capacity of the Watermark-Channel: How Many Bits Can Be Hidden Within a Digital Image. In *Proc. of SPIE Vol. 3657, Security and Watermarking of Multimedia Contents*, pages 437–448. SPIE—The International Society for Optical Engineering, April 1999.

[BCW00] Marko Balabanović, Lonny L. Chu, and Gregory J. Wolff. Storytelling with Digital Photographs. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 2000)*, pages 564–571. ACM Press, April 2000.

[Ben99] Oliver Benedens. Geometry-Based Watermarking of 3D Models. *IEEE Computer Graphics and Applications*, 19(1):46–55, January 1999.

[Ber83] Jacques Bertin. *Semiology of Graphics: Diagrams, Networks, Maps.* The University of Wisconsin Press, Ltd., 1983.

[BFH01] Blaine Bell, Steven Feiner, and Tobias Höllerer. View Management for Virtual and Augmented Reality. In *Proc. of UIST 2001, ACM Symposium on User Interface Software and Technology*, pages 101–110. ACM Press, November 2001.

[BFH02] Blaine Bell, Steven Feiner, and Tobias Höllerer. Information at a Glance. *IEEE Computer Graphics & Applications*, 22(4):6–9, July/August 2002.

[BG04] Patrick Baudisch and Carl Gutwin. Multiblending: Displaying Overlapping Windows Simultaneously Without the Drawbacks of Alpha Blending. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 2004)*, pages 367–374. ACM Press, April 2004.

[BGBS02] Patrick Baudisch, Nathaniel Good, Victoria Bellotti, and Pamela Schraedley. Keeping Things in Context: A Comparative Evaluation of Focus Plus Context Screens, Overviews, and Zooming. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 2002)*, pages 259–266. ACM Press, April 2002.

[BGML96] Walter Bender, Daniel Gruhl, Norishige Morimoto, and Anthony Lu. Techniques for Data Hiding. *IBM Systems Journal*, 35(3 & 4):313–336, 1996.

[BKM+02] Nikolaos V. Boulgouris, Ioannis Kompatsiaris, Vasileios Mezaris, Dimitrios Simitopoulos, and Michael G. Strintzis. Segmentation and Content-Based Watermarking for Color Image and Image Region Indexing and Retrieval. *EURASIP Journal on Applied Signal Processing*, 4:420–433, 2002.

[BLH04] Patrick Baudisch, Bongshin Lee, and Libby Hanna. Fishnet, a Fisheye Web Browser with Search Term Popouts: A Comparative Evaluation with Overview and Linear View. In *Proc. of the Working Conference on Advanced Visual Interfaces (AVI 2004)*, pages 133–140. ACM Press, May 2004.

[BLMO95] Jack T. Brassil, Steven H. Low, Nicholas F. Maxemchuk, and Lawrence O'Gorman. Electronic Marking and Identification Techniques to Discourage Document Copying. *IEEE Journal on Selected Areas in Communications*, 13(8):1495–1504, October 1995.

[BS95] C. Wayne Brown and Barry J. Shepherd. *Graphics File Formats: Reference and Guide*. Manning Publications Co., 1995.

[BSP+93] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and Magic Lenses: The See-Through Interface. In *Proc. of the 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'93)*, pages 73–80. ACM Press, August 1993.

[BWHR99] James F. Brinkley, Benjamin A. Wong, Kevin P. Hinshaw, and Cornelius Rosse. Design of an Anatomy Information System. *IEEE Computer Graphics & Applications*, 19(3):38–48, May/June 1999.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press Books, 1999.

[BZGM02] Niels Olof Bouvin, Polle T. Zellweger, Kaj Grønbæk, and Jock D. Mackinlay. Fluid Annotations through Open Hypermedia: Using and Extending Emerging Web Standards. In *Proc. of the 11th International Conference on World Wide Web*, pages 160–171. ACM Press, May 2002.

[Cag02] Kurt Cagle. *SVG Programming: The Graphical Web*. Springer-Verlag, 2002.

[Car99] M. Sheelagh T. Carpendale. *A Framework for Elastic Presentation Space*. PhD thesis, Simon Fraser University, Burnaby, British Columbia, Canada, March 1999.

[CCF95] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. 3-Dimensional Pliable Surfaces: For the Effective Presentation of Visual Information. In *Proc. of UIST 1995, ACM Symposium on User Interface Software and Technology*, pages 217–226. ACM Press, November 1995.

[CCF97] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Extending Distortion Viewing from 2D to 3D. *IEEE Computer Graphics & Applications(Special Issue on Information Visualization)*, 17(4):42–51, July/August 1997.

[CDF+86] Graham Campbell, Thomas A. DeFanti, Jeff Frederiksen, Stephen A. Joyce, and Lawrence A. Leske. Two Bit/Pixel Full Color Encoding. In *Proc. of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'86)*, pages 215–223. ACM Press, August 1986.

[CKLS97] Ingemar J. Cox, Joe Kilian, Tom Leighton, and Talal Shamoon. Secure Spread Spectrum Watermarking for Multimedia. *IEEE Transactions on Image Processing*, 6(12):1673–1687, 1997.

[CM01] M. Sheelagh T. Carpendale and Catherine Montagnese. A Framework for Unifying Presentation Space. In *Proc. of UIST 2001, ACM Symposium on User Interface Software and Technology*, pages 61–70. ACM Press, November 2001.

[CM02] Ingemar J. Cox and Matthew L. Miller. The First 50 Years of Electronic Watermarking. *EURASIP Journal on Applied Signal Processing*, 2:126–132, February 2002.

[CMB02] Ingemar J. Cox, Matthew L. Miller, and Jeffrey A. Bloom. *Digital Watermarking*. Morgan Kaufmann Publishers, 2002.

[Cos83] Max H. M. Costa. Writing on Dirty Paper. *IEEE Transactions on Information Theory*, 29(3):439–441, May 1983.

[CS02] Wallace Chigona and Thomas Strothotte. Contextualized Text Explanation for Visualizations. In *Proc. of the 2nd International Symposium on Smart Graphics*, pages 27–34. ACM Press, June 2002.

[CSRS03] Wallace Chigona, Henry Sonnet, Felix Ritter, and Thomas Strothotte. Shadows with a Message. In *Proc. of the 3rd International Symposium on Smart Graphics*, pages 91–101. Springer-Verlag, July 2003.

[Dau95] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1995.

[DC98] Marc Dymetman and Max Copperman. Intelligent Paper. In *Proc. of the 7th International Conference on Electronic Publishing*, pages 392–406. Springer-Verlag, April 1998.

[DIC04] DICOM Standards Committee. *Digital Imaging and Communications in Medicine (DICOM)*, October 2004. DICOM PS 3 - 2004 standard.

[Dig00] Digital Imaging Group, Inc. *Metadata for Digital Images*, August 2000. DIG35 Specification, Version 1.0.

[Dit00] Jana Dittmann. *Digitale Wasserzeichen: Grundlagen, Verfahren, Anwendungsgebiete*. Springer-Verlag, 2000.

[dV38] Œuvres Complétes de Voltaire. *Éléments de la Philosophie de Newton*. Voltaire Foundation, 1738.

[DVML02] Jean-François Delaigle, Christophe De Vleeschouwer, Benoit Macq, and I. Langendijk. Human Visual System Features Enabling Watermarking. In *Proc. of the IEEE International Conference on Multimedia and Expo (ICME 2002), Vol. 2*, pages 489–492. IEEE Computer Society Press, August 2002.

[EGD01] Dustin Eggink, Mark D. Gross, and Ellen Yi-Luen Do. Smart Objects: Constraints and Behaviors in a 3D Design Environment. In *Proc. of the Conference on Architectural Information Management*, pages 460–465. ECAADE, August 2001.

[FGD02] Jessica Fridrich, Miroslav Goljan, and Rui Du. Lossless Data Embedding for All Image Formats. In *Proc. of SPIE Vol. 4675, Security and Watermarking of Multimedia Contents IV*, pages 572–583. SPIE—The International Society for Optical Engineering, April 2002.

[Fiu89] Eugene L. Fiume. *The Mathematical Structure of Raster Graphics*. Academic Press, 1989.

[FP99] Jean-Daniel Fekete and Catherine Plaisant. Excentric Labeling: Dynamic Neighborhood Labeling for Data Visualization. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 1999)*, pages 512–519. ACM Press, May 1999.

[FvDFH96] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Publishing Company, Inc., 1996.

[GC99] Jaideva C. Goswami and Andrew K. Chan. *Fundamentals of Wavelets: Theory, Algorithms, and Applications*. John Wiley & Sons, Ltd., 1999.

[GC04] Vladimir Geroimenko and Chaomei Chen. *Visualizing Information Using SVG and X3D: XML-Based Technologies for the XML-Based Web*. Springer-Verlag, November 2004.

[GD03] Emmanuel Garcia and Jean-Luc Dugelay. Texture-Based Watermarking of 3-D Video Objects. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(8):853–866, August 2003.

[GE04] Emir Ganic and Ahmet M. Eskicioglu. Robust DWT-SVD Domain Image Watermarking: Embedding Data in All Frequencies. In *Proc. of the Multimedia and Security Workshop 2004*, pages 166–174. ACM Press, September 2004.

[GF04] Carl Gutwin and Chris Fedak. A Comparison of Fisheye Lenses for Interactive Layout Tasks. In *Proc. of the 2004 Conference on Graphics Interface*, pages 213–220. Canadian Human-Computer Communications Society, May 2004.

[GHS06] Timo Götzelmann, Knut Hartmann, and Thomas Strothotte. Agent-Based Annotation of Interactive 3D Visualizations. In *Proc. of the 6th International Symposium on Smart Graphics*, July 2006.

[GKH+99] Polina Golland, Ron Kikinis, Michael Halle, Chris Umans, W. Eric L. Grimson, Martha E. Shenton, and Jens A. Richolt. AnatomyBrowser: A Novel Approach to Visualization and Integration of Medical Information. *Computer Aided Surgery*, 4(3):129–143, 1999.

[Gol96] E. Bruce Goldstein. *Sensation and Perception*. Brooks/Cole Publishing Company, 1996.

[Goo02] Danny Goodman. *Dynamic HTML: The Definitive Reference*. O'Reilly Media, Inc., September 2002.

[GP90] Michael Gervautz and Werner Purgathofer. A Simple Method for Color Quantization: Octree Quantization. In Andrew S. Glassner, editor, *Graphics Gems I*, pages 287–293. Academic Press, 1990.

[GPSS01] Vladimir I. Gorodetski, Leonard J. Popyack, Vladimir Samoilov, and Victor A. Skormin. SVD-Based Approach to Transparent Embedding Data into Digital Images. In *Proc. of the International Workshop on Information Assurance in Computer Networks: Methods, Models, and Architectures for Network Security*, pages 263–274. Springer-Verlag, 2001.

[GS03] Carl Gutwin and Amy Skopik. Fisheye Views are Good for Large Steering Tasks. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 2003)*, pages 201–208. ACM Press, April 2003.

[GSG+02] Michael Gertz, Kai-Uwe Sattler, Fredric Gorin, Michael Hogarth, and Jim Stone. Annotating Scientific Images: A Concept-Based Approach. In *Proc. of the 14th International Conference on Scientific and Statistical Database Management*, pages 59–68. IEEE Computer Society Press, July 2002.

[Hal01] Terry Halpin. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Academic Press, 2001.

[HAPG00] Juan R. Hernández, Martín Amado, and Fernando Pérez-González. DCT-Domain Watermarking Techniques for Still Images: Detector Performance Analysis and a New Structure. *IEEE Transactions on Image Processing*, 9(1):55–68, January 2000.

[Hat98] Gary Hatfield. Attention in Early Scientific Psychology. In *Vancouver Studies in Cognitive Science: Visual Attention*, pages 3–25. Oxford University Press, 1998.

[HB02] Thomas Harte and Adrian G. Bors. Watermarking 3D Models. In *Proc. of the IEEE 2002 International Conference on Image Processing*, pages 661–664. IEEE Computer Society Press, September 2002.

[Hea99] Christopher G. Healey. Fundamental Issues of Visual Perception for Effective Image Generation. In *SIGGRAPH 1999 Course #6 Notes*, pages 1–42. ACM Press, August 1999.

[Hea01] Christopher G. Healey. Formalizing Artistic Techniques and Scientific Visualization for Painted Renditions of Complex Information Spaces. In *Proc. of the 7th International Joint Conference on Artificial Intelligence*, pages 371–376. Morgan Kaufmann Publishers, August 2001.

[Hec82] Paul S. Heckbert. Color Image Quantization for Frame Buffer Display. In *Proc. of the 9th Annual Conference on Computer Graphics and Interactive Techniques*, pages 297–307. ACM Press, 1982.

[Hec01] David L. Hecht. Printed Embedded Data Graphical User Interfaces. *IEEE Computer*, 34(3):47–55, March 2001.

[Her02] Cormac Herley. Why Watermarking is Nonsense. *IEEE Signal Processing Magazine*, 19(5):10–11, September 2002.

[Hod03] Elaine R. S. Hodges, editor. *The Guild Handbook of Scientific Illustration*. John Wiley & Sons, Ltd., second edition, may 2003.

[HPP+95] Karl Heinz Höhne, Bernhard Pflesser, Andreas Pommert, Martin Riemer, Rainer Schubert, and Ulf Tiede. A New Representation of Knowledge concerning Human Anatomy and Function. *Nature Medicine*, 1(6):506–511, June 1995.

[Hub02] Bill Huber. GIS & Steganography—Part 3: Vector Steganography. Directions Magazine, April 2002. http://www.directionsmag.com.

[HZR+92] Kenneth P. Herndon, Robert C. Zeleznik, Daniel C. Robbins, D. Brookshire Conner, Scott S. Snibbe, and Andries van Dam. Interactive Shadows. In *Proc. of UIST 1992, ACM Symposium on User Interface Software and Technology*, pages 1–6. ACM Press, November 1992.

[IF04] Edward W. Ishak and Steven K. Feiner. Interacting with Hidden Content Using Content-Aware Free-Space Transparency. In *Proc. of UIST 2004, ACM Symposium on User Interface Software and Technology*, pages 189–192. ACM Press, October 2004.

[JGD02] Thomas Jung, Mark D. Gross, and Ellen Yi-Luen Do. Annotating and Sketching on 3D Web Models. In *Proc. of the 7th International Conference on Intelligent User Interfaces (IUI 2002)*, pages 95–102. ACM Press, January 2002.

[JJ98] Neil F. Johnson and Sushil Jajodia. Exploring Steganography: Seeing the Unseen. *IEEE Computer*, 31(2):26–34, February 1998.

[Joi04a] Joint Technical Committee ISO/IEC JTC1. *Information Technology—JPEG 2000 Image Coding System: Extensions*, first edition, May 2004. International Standard ISO/IEC 15444-2.

[Joi04b] Joint Technical Committee ISO/IEC JTC1 SC24 and PNG Group (W3C). *Portable Network Graphics (PNG) Specification*, second edition, March 2004. International Standard ISO/IEC 15948:2003.

[JS01] Roland Jesse and Thomas Strothotte. Motion Enhanced Visualization in Support of Information Fusion. In *Proc. of the 2001 International Conference*

on Imaging Science, Systems, and Technology, pages 492–497. CSREA Press, June 2001.

[JW04] James S. Janosky and Rutherford W. Witthus. Using JPEG2000 for Enhanced Preservation and Web Access of Digital Archives—A Case Study. In *IS&T's 2004 Archiving Conference*, volume 1, pages 145–149. IS&T—The Society for Imaging Science and Technology, April 2004.

[Kai94] Gerald Kaiser. *A Friendly Guide to Wavelets*. Birkhäuser, 1994.

[KBE99] Martin Kutter, Sushil K. Bhattacharjee, and Touradj Ebrahimi. Towards Second Generation Watermarking Schemes. In *Proc. of the IEEE 1999 International Conference on Image Processing*, pages 320–323. IEEE Computer Society Press, October 1999.

[KDG99] Andreas König, Helmut Doleisch, and Eduard Gröller. Multiple Views and Magic Mirrors – fMRI Visualization of the Human Brain. In *Proc. of the Spring Conference on Computer Graphics and its Applications*, pages 130–139, April/May 1999.

[KDK98] Satoshi Kanai, Hiroaki Date, and Takeshi Kishinami. Digital Watermarking for 3D Polygons using Multiresolution Wavelet Decomposition. In *Proc. of the IFIP Working Group 5.2, Sixth International Workshop on Geometric Modelling: Fundamentals and Applications (GEO-6)*, pages 296–307, December 1998.

[KJB97] Martin Kutter, Frédéric D. Jordan, and Frank Bossen. Digital Signature of Color Images using Amplitude Modulation. In *Proc. of SPIE Vol. 3022, Storage and Retrieval for Image and Video Databases V*, pages 518–526. SPIE—The International Society for Optical Engineering, January 1997.

[Kle93] Stanley A. Klein. Image Quality and Image Compression: A Psychophysicist's Viewpoint. In Andrew B. Watson, editor, *Digital Images and Human Vision*, pages 73–88. The MIT Press, 1993.

[KLS00] Matthias Kreuseler, Norma López, and Heidrun Schumann. A Scalable Framework for Information Visualization. In *Proc. of the IEEE Symposium on Information Visualization 2000*, pages 27–36. IEEE Computer Society Press, October 2000.

[KM92] Charles Kurak and John McHugh. A Cautionary Note on Image Downgrading. In *Proc. of the Annual Computer Security Applications Conference*, pages 153–159. IEEE Computer Society Press, November 1992.

[KMH02] Robert Kosara, Silvia Miksch, and Helwig Hauser. Focus+Context Taken Literally. *IEEE Computer Graphics & Applications*, 22(1):22–29, January/February 2002.

[KP99] Martin Kutter and Fabien A. P. Petitcolas. A Fair Benchmark for Image Watermarking Systems. In *Proc. of SPIE Vol. 3657: Security and Watermarking of Multimedia Contents*, pages 226–239. SPIE—The International Society for Optical Engineering, January 1999.

[KPC+99] Allan Kuchinsky, Celine Pering, Michael L. Creech, Dennis Freeze, Bill Serra, and Jacek Gwizdka. FotoFile: A Consumer Multimedia Organization and Retrieval System. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 1999)*, pages 496–503. ACM Press, May 1999.

[KR97] T. Alan Keahey and Edward L. Robertson. Nonlinear Magnification Fields. In *Proc. of the 1997 IEEE Symposium on Information Visualization*, pages 51–59. IEEE Computer Society Press, October 1997.

[KS04] Seong-Whan Kim and Shan Suthaharan. An Entropy Masking Model for Multimedia Content Watermarking. In *Proc. of the 37th Annual Hawaii International Conference on System Sciences*, page 70182.2. IEEE Computer Society Press, January 2004.

[KT98] Konstantinos G. Kakoulis and Ioannis G. Tollis. A Unified Approach to Labeling Graphical Features. In *Proc. of the 14th Annual Symposium on Computational Geometry*, pages 347–356. ACM Press, June 1998.

[KZ95] Eckhard Koch and Jian Zhao. Towards Robust and Hidden Image Copyright Labeling. In *Proc. of the 1995 IEEE Workshop on Nonlinear Signal and Image Processing*, pages 452–455. IEEE Computer Society Press, June 1995.

[LA94] Ying K. Leung and Mark D. Apperley. A Review and Taxonomy of Distortion-Oriented Presentation Techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, June 1994.

[LC99] Wen-Nung Lie and Li-Chun Chang. Data Hiding in Images with Adaptive Numbers of Least Significant Bits based on the Human Visual System. In *Proc. of the IEEE 1999 International Conference on Image Processing*, volume 1, pages 286–290. IEEE Computer Society Press, October 1999.

[LH94] Maria M. Loughlin and John F. Hughes. An Annotation System for 3D Fluid Flow Visualization. In *Proc. of the Conference on Visualization '94*, pages 273–279. IEEE Computer Society Press, October 1994.

[Lip00] Herbert Lippert. *Lehrbuch Anatomie*. Urban & Fischer Verlag, fifth edition, 2000.

[LPS98] Jia Li, Catherine Plaisant, and Ben Shneiderman. Data Object and Label Placement for Information Abundant Visualizations. In *Proc. of the 1998 Workshop on New Paradigms in Information Visualization and Manipulation*, pages 41–48. ACM Press, November 1998.

[LS99] Clement H. C. Leung and Dwi Sutanto. Multimedia Data Modeling and Management for Semantic Content Retrieval. In *Handbook of Multimedia Computing*, pages 43–54. CRC Press LLC, 1999.

[Mar82] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* W. H. Freeman & Company, 1982.

[Mar98] Catherine C. Marshall. Making Metadata: A Study of Metadata Creation for a Mixed Physical-Digital Collection. In *Proc. of the 3rd ACM Conference on Digital Libraries*, pages 162–171. ACM Press, June 1998.

[Mas05] Mark A. Masry. A Watermarking Algorithm for Map and Chart Images. In *Proc. of SPIE Vol. 5681: Security, Steganography, and Watermarking of Multimedia Contents VII*, pages 495–503. SPIE—The International Society for Optical Engineering, March 2005.

[MB99] Fred Mintzer and Gordon W. Braudaway. If One Watermark is Good, are More Better? In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'99)*, pages 2067–2069. IEEE Computer Society Press, March 1999.

[MDB87] Bruce H. McCormick, Thomas A. DeFanti, and Maxine D. Brown. Visualization in Scientific Computing and Computer Graphics. *ACM SIGGRAPH Computer Graphics*, 21(6):1–14, November 1987.

[ME04] Yannick Maret and Touradj Ebrahimi. Data Hiding on 3D Polygonal Meshes. In *Proc. of the Multimedia and Security Workshop 2004*, pages 68–74. ACM Press, September 2004.

[MF97] Margaret W. Matlin and Hugh J. Foley. *Sensation and Perception.* Allyn and Bacon, fourth edition, 1997.

[Mia99] John Miano. *Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP.* Addison-Wesley Publishing Company, Inc., 1999.

[MLC00] Ira S. Moskowitz, Garth E. Longdon, and LiWu Chang. A New Paradigm Hidden in Steganography. In *Proc. of the Workshop on New Security Paradigms (NSPW'00)*, pages 41–50. ACM Press, September 2000.

[MSI01] Xiaoyang Mao, Makoto Shiba, and Atsumi Imamiya. Watermarking 3D Geometric Models through Triangle Subdivision. In *Proc. of SPIE Vol.4314, Security and Watermarking of Multimedia Contents III*, pages 253–260. SPIE—The International Society for Optical Engineering, August 2001.

[MU01] Peter Meerwald and Andreas Uhl. A Survey of Wavelet-Domain Watermarking Algorithms. In *Proc. of SPIE Vol. 4314, Security and Watermarking of Multimedia Contents III*, pages 505–516. SPIE—The International Society for Optical Engineering, January 2001.

[MvR96] James D. Murray and William van Ryper. *Encyclopedia of Graphics File Formats*. O'Reilly Media, Inc., second edition, 1996.

[MW87] Carol Bergfeld Mills and Linda J. Weldon. Reading Text from Computer Screens. *ACM Computing Surveys*, 19(4):329–357, December 1987.

[NP01] Athanasios Nikolaidis and Ioannis Pitas. Region-Based Image Watermarking. *IEEE Transactions on Image Processing*, 10(11):1726–1740, November 2001.

[NS02] Moira C. Norrie and Beat Signer. Web-Based Integration of Printed and Digital Information. In *Proc. of the 2nd Workshop on Data Integration over the Web (DiWeb'02)*, pages 71–85. University of Toronto Press, May 2002.

[NS05] Moira C. Norrie and Beat Signer. Overlaying Paper Maps with Digital Information Services for Tourists. In *Proc. of the 12th International Conference on Information Technology and Travel & Tourism (ENTER 2005)*, pages 23–34. Springer-Verlag, January 2005.

[OMA97] Ryutarou Ohbuchi, Hiroshi Masuda, and Masaki Aono. Watermarking Three-Dimensional Polygonal Models. In *Proc. of the ACM International Conference on Multimedia 1997*, pages 261–272. Addison-Wesley Publishing Company, Inc., November 1997.

[OMA98] Ryutarou Ohbuchi, Hiroshi Masuda, and Masaki Aono. Watermarking Multiple Object Types in Three-Dimensional Models. In *Proc. of the Multimedia and Security Workshop at ACM Multimedia*, pages 83–91. ACM Press, September 1998.

[OMT02] Ryutarou Ohbuchi, Akio Mukaiyama, and Shigeo Takahashi. A Frequency-Domain Approach to Watermarking 3D Shapes. *Computer Graphics Forum*, 21(3):373–382, 2002.

[OUE03] Ryutarou Ohbuchi, Hiroo Ueda, and Shuh Endoh. Watermarking 2D Vector Maps in the Mesh-Spectral Domain. In *Proc. of the International Conference on Shape Modelling and Applications (SMI 2003)*, pages 216–225. IEEE Computer Society Press, May 2003.

[PAK99] Fabien A. P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn. Information Hiding—A Survey. *Proc. of the IEEE, Special Issue on Protection of Multimedia Content*, 87(7):1062–1078, July 1999.

[Par97] James R. Parker. *Algorithms for Image Processing and Computer Vision*. John Wiley & Sons, Ltd., 1997.

[PHF99] Emil Praun, Hugues Hoppe, and Adam Finkelstein. Robust Mesh Watermarking. In *Proc. of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'99)*, pages 49–56. ACM Press, August 1999.

[PHP+01] Andreas Pommert, Karl Heinz Höhne, Bernhard Pflesser, Ernst Richter, Martin Riemer, Thomas Schiemann, Rainer Schubert, Udo Schumacher, and Ulf Tiede. Creating a High-Resolution Spatial/Symbolic Model of the Inner Organs Based on the Visible Human. *Medical Image Analysis*, 5(3):221–228, 2001.

[PR02] Bernhard Preim and Felix Ritter. Techniken zur Interaktiven Hervorhebung von Objekten in Medizinischen 3D-Visualisierungen. In *Simulation and Visualization 2002*, pages 187–200. SCS European Publishing House, March 2002.

[PRS97] Bernhard Preim, Andreas Raab, and Thomas Strothotte. Coherent Zooming of Illustrations with 3D-Graphics and Text. In *Proc. of the 1997 Conference on Graphics Interface*, pages 105–113. Canadian Information Processing Society, March 1997.

[PTD05] Bernhard Preim, Christian Tietjen, and Christina Dörge. NPR, Focussing and Emphasis in Medical Visualizations. In *Simulation and Visualization 2005*, pages 139–152. SCS European Publishing House, March 2005.

[RBH99] Antony Rix, Alex Bourret, and Mike Hollier. Models of Human Perception. *BT Technology Journal*, 17(1):24–34, January 1999.

[RD04] Robert Reinhardt and Snow Dowd. *Macromedia Flash MX 2004 Bible*. John Wiley & Sons, Ltd., 2004.

[RDB96] Joseph J. K. Ó Ruanaidh, William J. Dowling, and Frank M. Boland. Watermarking Digital Images for Copyright Protection. *IEE Proceedings - Vision, Image and Signal Processing*, 143(4):250–256, August 1996.

[RM93] George G. Robertson and Jock D. Mackinlay. The Document Lens. In *Proc. of UIST 1993, ACM Symposium on User Interface Software and Technology*, pages 101–108. ACM Press, November 1993.

[RMC91] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone Trees: Animated 3D Visualizations of Hierarchical Information. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 1991)*, pages 189–194. ACM Press, April/May 1991.

[RSHS03] Felix Ritter, Henry Sonnet, Knut Hartmann, and Thomas Strothotte. Illustrative Shadows: Integrating 3D and 2D Information Displays. In *Proc. of the 2003 International Conference on Intelligent User Interfaces*, pages 166–173. ACM Press, January 2003.

[RSV02] Philippe Rigaux, Michel Scholl, and Agnès Voisard. *Spatial Databases—With Application to GIS*. Academic Press, 2002.

[RV04] Rakhi Rajani and Alex Vorbau. Viewing and Annotating Media with MemoryNet. In *CHI '04: Extended Abstracts on Human Factors in Computing Systems*, pages 1517–1520. ACM Press, April 2004.

[RVEP04] Yuriy Rytsar, Sviatoslav Voloshynovskiy, Frederic Ehrler, and Thierry Pun. Interactive Segmentation with Hidden Object-Based Annotations: Toward Smart Media. In *Proc. of SPIE Vol. 5307, Storage and Retrieval Methods and Applications for Multimedia*, pages 29–37. SPIE—The International Society for Optical Engineering, January 2004.

[SB92] Manojit Sarkar and Marc H. Brown. Graphical Fisheye Views of Graphs. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 1992)*, pages 83–91. ACM Press, May 1992.

[SCE01] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The JPEG 2000 Still Image Compression Standard. *IEEE Signal Processing Magazine*, 18(5):36–58, September 2001.

[SCS04] Henry Sonnet, M. Sheelagh T. Carpendale, and Thomas Strothotte. Integrating Expanding Annotations with a 3D Explosion Probe. In *Proc. of the Working Conference on Advanced Visual Interfaces (AVI 2004)*, pages 63–70. ACM Press, May 2004.

[SCS05] Henry Sonnet, M. Sheelagh T. Carpendale, and Thomas Strothotte. Integration of 3D Data and Text: The Effects of Text Positioning, Connectivity, and Visual Hints on Comprehension. In *Human-Computer-Interaction (INTERACT 2005), IFIP TC13 International Conference Proceedings*, pages 615–628. Springer-Verlag, September 2005.

[SFB94] Maureen C. Stone, Ken Fishkin, and Eric A. Bier. The Movable Filter as a User Interface Tool. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 1994)*, pages 306–312. ACM Press, April 1994.

[Shn96] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualization. In *Proc. of the 1996 IEEE Conference on Visual Languages*, pages 336–343. IEEE Computer Society Press, September 1996.

[Shn03] Ben Shneiderman. Why Not Make Interfaces Better than 3D Reality? *IEEE Computer Graphics & Applications*, 23(6):12–15, November/December 2003.

[SIDS03] Henry Sonnet, Tobias Isenberg, Jana Dittmann, and Thomas Strothotte. Illustration Watermarks for Vector Graphics. In *Proc. of Pacific Graphics 2003*, pages 73–82. IEEE Computer Society Press, October 2003.

[SK00] Ben Shneiderman and Hyunmo Kang. Direct Annotation: A Drag-and-Drop Strategy for Labeling Photos. In *Proc. of the International Conference on Information Visualization*, pages 88–95, July 2000.

[SKT98] Mitchell D. Swanson, Mei Kobayashi, and Ahmend H. Tewfik. Multimedia Data-Embedding and Watermarking Technologies. *Proc. of the IEEE, Special Issue on Multimedia Signal Processing*, 86(6):1064–1087, June 1998.

[SL05] Henry Sonnet and Silvio Lange. Data Storage: Carrier Objects as Illustration Watermarks for 3D Polygonal Models. In *Simulation and Visualization 2005*, pages 305–316. SCS European Publishing House, March 2005.

[SP04a] Ben Shneiderman and Catherine Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Publishing Company, Inc., fourth edition, 2004.

[SP04b] Vassilios Solachidis and Ioannis Pitas. Watermarking Polygonal Lines Using Fourier Descriptors. *IEEE Computer Graphics & Applications*, 24(3):44–51, May/June 2004.

[Spe06] Michael Specht. Using Perceptually Uniform Color Spaces for Image Steganography. An Enhancement of the Least Significant Bit Method. Technical Report 2, Otto-von-Guericke-Universität Magdeburg, Faculty for Computer Science, April 2006.

[SS97] Christine Strothotte and Thomas Strothotte. *Seeing Between the Pixels: Pictures in Interactive Systems*. Springer-Verlag, 1997.

[SS00] Stefan Schlechtweg and Thomas Strothotte. Generating Scientific Illustrations in Electronic Books. In *Smart Graphics. Papers from the 2000 AAAI Spring Symposium*, pages 8–15. AAAI Press, March 2000.

[ST97] Gaurav Sharma and H. Joel Trussell. Digital Color Imaging. *IEEE Transactions on Image Processing*, 6(7):901–932, July 1997.

[Sto03] Maureen C. Stone. *A Field Guide to Digital Color*. A K Peters, Ltd., 2003.

[Sum04] SumTotal Systems, Inc. *ToolBook Instructor 2004—User Guide*, March 2004. Part No. 070119.

[SUS+06] Henry Sonnet, Andrea Unger, Lothar Schlesier, Thomas Vogel, Tobias Isenberg, and Thomas Strothotte. Interactive Images using Illustration Watermarks: Techniques, Study, and Applications. Technical Report 7, Otto-von-Guericke-Universität Magdeburg, Faculty for Computer Science, June 2006.

[SWK99] Po-Chyi Su, Houng-Jyh M. Wang, and C.-C. Jay Kuo. Digital Image Watermarking in Regions of Interest. In *Proc. of the Conference on Image Processing, Image Quality and Image Capture Systems*, pages 295–300. IS&T—The Society for Imaging Science and Technology, April 1999.

[TLRA03] Kentaro Toyama, Ron Logan, Asta Roseway, and P. Anandan. Geographic Location Tags on Digital Images. In *Proc. of the 11th ACM International Conference on Multimedia*, pages 156–166. ACM Press, November 2003.

[TM02] David S. Taubman and Michael W. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers Group, 2002.

[Tre85] Anne Treisman. Preattentive Processing in Vision. *Computer Vision, Graphics, and Image Processing*, 31(2):156–177, August 1985.

[TRvS+93] Andrew Z. Tirkel, G. A. Rankin, Ron G. van Schyndel, W. J. Ho, N. R. A. Mee, and Charles F. Osborne. Electronic Water Mark. In *Proc. of the 2nd Conference on Digital Imaging, Computing Techniques, and Applications (DICTA-93)*, pages 666–672, December 1993.

[Tuf01] Edward R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, second edition, 2001.

[UCB04] Francesca Uccheddu, Massimiliano Corsini, and Mauro Barni. Wavelet-Based Blind Watermarking of 3D Models. In *Proc. of the Multimedia and Security Workshop 2004*, pages 143–154. ACM Press, September 2004.

[VB02] Michael Voigt and Christoph Busch. Watermarking 2D-Vector Data for Geographical Information Systems. In *Proc. of SPIE Vol. 4675: Security and Watermarking of Multimedia Contents IV*, pages 621–628. SPIE—The International Society for Optical Engineering, April 2002.

[VCWP96] John Viega, Matthew J. Conway, George Williams, and Randy Pausch. 3D Magic Lenses. In *Proc. of UIST 1996, ACM Symposium on User Interface Software and Technology*, pages 51–58. ACM Press, November 1996.

[VD05] Thomas Vogel and Jana Dittmann. Illustration Watermarking: An Object Based Approach for Digital Images. In *Proc. of SPIE Vol. 5681: Security, Steganography, and Watermarking of Multimedia Contents VII*, pages 578–589. SPIE—The International Society for Optical Engineering, January 2005.

[vDD02] Marc van Droogenbroeck and Jórôme Delvaux. An Entropy Based Technique for Information Embedding in Images. In *Proc. of the 3rd IEEE Benelux Signal Processing Symposium (SPS-2002)*, pages 81–84, March 2002.

[VYB04] Michael Voigt, Bian Yang, and Christoph Busch. Reversible Watermarking of 2D-Vector Data. In *Proc. of the Multimedia and Security Workshop 2004*, pages 160–165. ACM Press, September 2004.

[Wal92] Gregory K. Wallace. The JPEG Still Picture Compression Standard. *IEEE Transactions on Consumer Electronics*, 38(1):18–34, February 1992.

[Wal96] Karl K. Walther. *Lexikon der Buchkunst und Bibliophilie.* Weltbild Verlag GmbH, 1996.

[Wat93] Andrew B. Watson. Image Coding and Compression. In *Digital Images and Human Vision*, pages 1–2. The MIT Press, 1993.

[WBT97] Andrew B. Watson, Robert Borthwick, and Mathias Taylor. Image Quality and Entropy Masking. In *Proc. of SPIE Vol. 3016, Human Vision and Electronic Imaging II*, pages 2–12. SPIE—The International Society for Optical Engineering, February 1997.

[Wem03] Faithe Wempen. *Microsoft Office—PowerPoint 2003 Bible*. John Wiley & Sons, Ltd., 2003.

[Wen99] Robin Wendler. LDI Update: Metadata in the Library. *Harvard University, Library Notes*, (1286):4–5, July/August 1999.

[Wer94] Josie Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2*. Addison-Wesley Publishing Company, Inc., 1994.

[WKLH00] Jian K. Wu, Mohan S. Kankanhalli, Joo-Hwee Lim, and Dezhong Hong. *Perspectives on Content-Based Multimedia Systems*. Kluwer Academic Publishers Group, 2000.

[WL03] Min Wu and Bede Liu. *Multimedia Data Hiding*. Springer-Verlag, 2003.

[Wor03] World Wide Web Consortium (W3C): SVG Working Group. *Scalable Vector Graphics (SVG) 1.1 Specification*, January 2003.

[Wor05] World Wide Web Consortium (W3C): SYMM Working Group. *Synchronized Multimedia Integration Language (SMIL 2.0)*, second edition, January 2005.

[XBA98] Xiang G. Xia, Charles G. Boncelet, and Gonzalo R. Arce. Wavelet Transform Based Watermark for Digital Images. *Optics Express*, 3(12):497–511, December 1998.

[Yan98] Steven Yantis. Objects, Attention and Perceptual Experience. In *Vancouver Studies in Cognitive Science: Visual Attention*, pages 187–214. Oxford University Press, 1998.

[YPSZ01] Kangkang Yin, Zhigeng Pan, Jiaoying Shi, and David Zhang. Robust Mesh Watermarking Based on Multiresolution Processing. *Computers & Graphics*, 25(3):409–420, June 2001.

List of Figures

A List of Implementations

The following two tables provide an overview of the experimental studies (first table) and implementations (second table) which were done during this work. They list all persons directly involved, their contributions to this work, and the estimated times they spent (m = person months, w = person weeks, d = person days). However, there are many other persons not included in the tables who also contributed by helpful discussions.

Experiment	*Section*	*Contributors*	*Description*	*Time*
Image watermarking	5.1	Henry Sonnet	Study design, technique implementation, testing, analysis of results	3m
		Andrea Unger	Study design, technique implementation, testing, analysis of results	3m
		Thomas Vogel	Implementation of test program, testing	1m
3D scene and text integration	6.3.4	Henry Sonnet	Study design, technique implementation, testing, analysis of results	3m

Application	*Section*	*Authors*	*Description*	*Time*
Vector graphic watermarking	3.1	Henry Sonnet	PDF file parsing and watermark insertion	2w
	3.1	Tobias Isenberg	Line graphic generation	4d
3D model watermarking	3.2.2	Henry Sonnet	Watermark insertion into texture images	1d
	3.2.3	Silvio Lange	Carrier object generation, watermark insertion, and positioning inside the model	3m
Raster graphic watermarking	4.3.2	Michael Specht	LSB watermarking approach based on the CIELab color space	1m
	4.3.3	Andrea Unger	LSB watermarking approach that generates and uses capacity maps (based on entropy and color analysis)	2m
	4.4.2	Henry Sonnet	Applying the Wavelet transform for watermarking	3w
SMAGE	4.2.3	Henry Sonnet	Framework and library (*libSmage*)	2m
	7.1	Henry Sonnet	User interface, icon design, dictionary representation, documentation	4m
	7.1	Lothar Schlesier	User Interface, browser plugins, setup framework	4m
Images with dynamic contents	7.2	Henry Sonnet	Basic framework	2w
	7.2.2	Sadiq Sowdagar	Animation of image elements and event handling	3m
Metadata previewer	6.2.3	Nguyen D. Quyen	Cursor widget with focus+context view, user interface	3m
LENSEXPLORER	6.3.3	Henry Sonnet	Label layout affected by distortion and explosion diagram creation	3m

B List of Supervised Works

Internships

Name:	Andreas Oppermann
Title:	A Method for Interactively Creating Wax Masks for Batik
Submission:	January 2004
Advisor:	Dr. Sheelagh Carpendale
Name:	Lothar Schlesier
Title:	Eastern Perspective – Multi Projection Images
Submission:	January 2004
Advisor:	Dr. Sheelagh Carpendale
Name:	Michael Specht
Title:	Using Perceptually Uniform Color Spaces for Image Steganography. An Enhancement of the Least Significant Bit Method
Submission:	November 2004
Name:	Stefan Habelski
Title:	Realisation of Territory-Based Interaction Techniques for Supporting Tabletop Collaboration
Submission:	December 2004
Advisors:	Dr. Sheelagh Carpendale, Stacey D. Scott
Name:	Jan Oelze
Title:	Integration von Tool-Programmen zur Modifikation von Objektgeometrien in ein Interaktives Visualisierungssystem
Submission:	July 2005
Advisor:	Dr. Axel Hintze
Name:	Sebastian Kärsten
Title:	Sequentielle und Parallele Berechnung sowie Visualisierung von Wind- und Wellendaten der Irischen Küste
Submission:	July 2005

Master Theses

Name:	Wanchun Luo
Title:	Object-Related Illustration Watermarks in Cartoon Images
Submission:	February 2004
Advisor:	Dr. Knut Hartmann

Name: Venkata Sai Kishore Nagisetti
Title: Robust Illustration Watermarks
Submission: April 2005
Advisor: Dr. Knut Hartmann

Name: Dingcao Liu
Title: Region-Based Illustration Watermarks using Wavelet Decomposition
Submission: April 2005

Name: Sowdagar Sadiq Ali Baig
Title: Embedding of Computer Games through Watermark Illustrations
Submission: July 2005
Advisor: Dr. Knut Hartmann

Name: Cheng Qian
Title: Metadata Exploration in Images supported by an Adaptive Cursor
Submission: July 2006

Name: Nguyen Dinh Quyen
Title: Metadata Previewer: Exploiting Distortion Techniques for the Exploration of Watermarked Images
Submission: July 2006

Diploma Theses

Name: Silvio Lange
Title: Trägerobjekte als Illustrations-Wasserzeichen für 3D-Modelle
Submission: June 2004

Name: Andrea Unger
Title: Evaluation of an Illustration Watermarking Approach based on Entropy and Color Analysis
Submission: March 2006

Name: Alexander Opel
Title: Enhancing Watermarking Algorithms for Raster Images by Template based Synchronization
Submission: August 2006
Advisor: Wolfgang Funk

C Smage's Internet Page: www.smage.de

The software tool *Smage* which was developed in the course of this research has its own Internet page (see Fig. C.1). It provides basic information about this tool including a short documentation as well as a download area.

Figure C.1: *Internet page of the software project* Smage.

D Glossary

A selection of terms and their meanings within the scope of this research. Terms marked with a '(⋆)' were introduced in the course of this work.

2D Vector Graphic, pp. 24, 37

2D Vector Graphics are images which are descriptively represented (see *Descriptive Representation*). Their *Data Representations* consist of geometric primitives such as points, lines, and curves in two dimensions.

3D Geometric Model, pp. 25, 47

A *3D Geometric Model* is a *Data Representation* of a real-world or fictional object whose characteristics are specified in three dimensions. The geometry of the object is typically represented as a polygonal mesh. Besides the geometry, included textures and material properties can influence the appearance of the object.

3D Scene, p. 123

A *3D scene* is a virtual environment which consists of *3D Geometric Models* arranged in a certain manner and which can be explored by camera movement or by changing other viewing parameters.

Authoring Tool, pp. 15, 18, 31

An *Authoring Tool* is a software tool which assists a user in creating or modifying digital data. The data can, for example, be Web pages, games, presentations, or other digital media which allow for end-user interaction, which means the user can trigger certain events by which certain actions are executed.

Background Layer (⋆), p. 11

The *Background Layer* of a medium contains the medium's visual information which is displayed in the initial state. In this state, it is not augmented with *Descriptive Metadata*.

Background Layer++ (⋆), pp. 15, 31

Background Layer++ refers to a storage technique that integrates *Background* and *Illustration Layer* data. The *Data Representation* of the medium typically provides some space for metadata storage at a certain position in its file format structure. Images whose file headers can include metadata are an example for this concept.

Blind Detection, p. 22

Blind Detection means that the original, unwatermarked medium is not required for decoding the watermark data encoded in the watermarked medium.

Capacity

See *Data Payload*.

Capacity Map (⋆), pp. 80, 86

Some *Illustration Watermarking* techniques analyze the properties of the *Cover Medium* before the watermark data is encoded. With respect to *Raster Graphics*, for example, entropies of pixel blocks are computed and different color spaces are exploited to determine *Capacity* suggestions for each pixel. These are then stored as intensity values in the *Capacity Map* which is a *Raster Graphic* and whose size corresponds to the size of the *Cover Medium*. During the actual watermarking, *Capacity Map* values can affect the amount of data which is embedded into a particular pixel's color channel.

Cover Medium, p. 21

The *Cover Medium* is the original medium into which watermark data is to be inserted. After watermarking, the medium is referred to as watermarked medium.

Data Payload, p. 22

Data Payload is the amount of data which is encoded as *Digital Watermark* in the *Cover Medium*.

Data Representation, pp. 13, 38, 48

Data Representation is the manner in which a certain digital medium is represented on a storage device. The medium is represented according to specific file format specifications which can vary depending on the application, the aim, and the inventor.

Descriptive Metadata, p. 11

Descriptive Metadata is information which has been added to a medium to assist the viewer in understanding the content of the medium, to describe and comment what is depicted, or to provide knowledge that cannot be extracted from the visualization alone.

Descriptive Representation, p. 16

A medium which is represented by means of a sequence of well-confined text descriptions is associated with the class of *Descriptively Represented* graphics. The display of those media depends on the manner in which the descriptions are eventually interpreted. Examples for those media are *2D Vector Graphics* or *3D Geometric Models*.

Digital Watermark, p. 20

A *Digital Watermark* is a sequence of bits which are to be embedded in the *Data Representation* of a digital medium using a *Digital Watermarking* technique.

Digital Watermarking, p. 20

Digital Watermarking is a methodology of hiding data which is in some relation with a digital medium within that medium itself. For this purpose, the *Data Representation* of the medium is typically slightly modified in a way so that the introduced changes are imperceivable to a viewer.

Entropy, p. 76

Entropy is a statistic measure which can be used to express the information content of a message. With respect to *Raster Graphics*, it can be used to characterize the texture of an image.

Fidelity, p. 22

Fidelity refers to the perceptual similarity between the original *Cover Medium* and the medium with the *Digital Watermark* embedded. *Transparency* means in this regard that neither the presence of embedded data nor any modifications introduced to the *Cover Medium* because of watermarking can be noticed by the viewer.

Gameable Image (⋆), p. 141

Gameable Images are standard *Raster Graphics* which allow for gaming. The necessary information such as controllable game characters, their behaviors, and how they can be affected by the player is included in the *Illustration Layer*. This layer is then stored as *Illustration Watermark* in the original image which serves as game background.

Hybrid Data Storage (⋆), p. 17

Hybrid Data Storage refers to a combination of *Integrated* and *Separated Data Storage*. Media, for example, which have parts of their *Illustration Layer* integrated in the *Background Layer* while other parts are separately stored are associated with this storage class.

ID-Buffer, p. 63

The *ID-Buffer* is a *Raster Graphic* which serves as reference during the watermarking. Its color depth of 8 bits per pixel allows to specify up to 256 different regions, each with a unique color. These regions mark those regions within the *Cover Medium* which are associated with *Descriptive Metadata* and which hence are to be watermarked.

Illustration Layer (⋆), p. 11

The *Illustration Layer* comprises the data which is necessary to display information in addition to what is shown by the *Background Layer*. *Illustration Layer* data is typically invisible in the initial view and can be requested by the end-user. The *Illustration Layer* can consist of *Descriptive Metadata*, specify interrelations between *Background* and *Illustration Layer* components, and contain information about which actions are to be triggered by what events.

Illustration Watermark (⋆), p. 29

Illustration Watermarks are *Digital Watermarks* with certain properties. Their purpose is to add value to a medium from which an end-user can benefit. Typically, *Illustration Watermarks* provide information that enables end-user interaction with the watermarked medium. The procedure of embedding an *Illustration Watermark* in a *Cover Medium* is referred to as *Illustration Watermarking*.

Integrated Data Storage (⋆), p. 15

Integrated Data Storage refers to storage techniques by which *Background Layer* and *Illustration Layer* data are included in the same data file.

Layout, pp. 120, 123, 127

Layout is the manner in which graphic elements are arranged in limited space. With respect to the *Illustration Layer*, *Layout* refers to positioning *Illustration Layer* components on the *Background Layer* according to specific requirements (e. g., prevention of occlusion, readability, and unambiguous correlations).

LSB, p. 26

LSB is the *L*east *S*ignificant *B*it (first bit) of a byte. With respect to RGB color channels each represented by one byte, the *LSB* of the blue color channel is typically the first bit which is modified during watermarking.

Magic Lens, pp. 114, 123

The term *Magic Lens* refers to operators which are placed in front of a visualization to display alternative information within the region that is confined by the lens.

Metadata-Related Information (⋆), pp. 107, 114

Metadata-Related Information is information which is provided to the viewer during the exploration of a medium which has *Descriptive Metadata* associated. It comprises information regarding the location (which regions of the medium are linked with metadata) and content (e. g., type, amount, and keywords) of the metadata which is available.

Raster Graphic, pp. 26, 59

A *Raster Graphic* is a digital image which is represented as a 2D matrix of pixel values. Each pixel value defines a color value at a specific position within the matrix.

Robustness, p. 22

If the embedded *Digital Watermark* can be detected and decoded even though signal processing operations were applied to the watermarked medium, the watermark is said to be robust. Such operations can include, for example, spatial filtering, lossy compression, printing and scanning, and geometric distortions such as rotation, translation, cropping, or scaling.

Separated Data Storage (⋆), p. 16

Separated Data Storage refers to storage techniques by which *Background Layer* and *Illustration Layer* data are stored in individual data files. Certain expressions are necessary to link *Background* and *Illustration Layer* components.

Smage (⋆), p. 135

Smage is the name of a software tool for end-users which allows to embed *Descriptive Metadata* in arbitrary *Raster Graphics*. To this end, regions in the image can be selected and metadata such as simple texts, text balloons, or images can be specified. These

metadata items are then encoded as *Illustration Watermarks* into their assigned image regions.

Spatial Domain Watermarking, pp. 24, 67, 71, 75

In the *Spatial Domain*, the *Digital Watermark* is embedded directly into the *Data Representation* of the *Cover Medium*, that is, the *Cover Medium* is not transformed into a different representation (e. g., by applying the *Wavelet Transform*) before watermarking.

Transformed Domain Watermarking, pp. 24, 82

Transformed Domain Watermarking techniques transform the *Data Representation* of the *Cover Medium* before the *Digital Watermark* is inserted. Such a transformation can, for example, result from applying the *Wavelet Transform*.

Transparency

See *Fidelity*.

Visual Representation, pp. 38, 48, 61, 119

Visual Representation is the actual visualization of a digital medium on a display device. It depends on the manner in which the medium's *Data Representation* is interpreted.

Wavelet Transform, p. 82

The *Wavelet Transform* is a mathematical transform which can be used to split up a signal into different time-scale representations using wavelets which are translated and dilated versions of the original wavelet function (mother wavelet).

Résumé

Personal Data

Name:	Henry Sonnet
Address:	Jakobstrasse 25 39104 Magdeburg Germany
Date of birth:	October 28, 1975
Place of birth:	Magdeburg, Germany
Nationality:	German

Education

05/2003 – 09/2003:	Research activities at the Dept. of Computer Science, University of Calgary, Canada
since 01/2003:	Ph.D. studies at the Dept. of Simulation and Graphics, Otto-von-Guericke University of Magdeburg
09/2002 – 12/2002:	Research assistant at the Dept. of Simulation and Graphics
08/2002:	Diploma degree in Computational Visualistics from the Otto-von-Guericke University of Magdeburg, graduation with distinction
12/1998 – 08/2002:	Graduate studies at the Faculty of Computer Science, Otto-von-Guericke University of Magdeburg
03/2000 – 08/2000:	Internship at "MeVis" in Bremen, Germany
10/1996 – 11/1998:	Undergraduate studies at the Faculty of Computer Science, Otto-von-Guericke University of Magdeburg
10/1994 – 12/1995:	Civilian service at the hospital "Altstadt" in Magdeburg
08/1990 – 06/1994:	High school education at the "Gymnasium Otto-von-Guericke" in Magdeburg (special emphasis on natural sciences and English)
09/1981 – 06/1990:	Basic school education in Magdeburg

Wissenschaftlicher Lebenslauf

Persönliche Daten

Name:	Henry Sonnet
Adresse:	Jakobstraße 25 39104 Magdeburg Deutschland
Geburtsdatum:	28. Oktober 1975
Geburtsort:	Magdeburg
Nationalität:	Deutsch

Ausbildung

09/1981 – 06/1990:	Allgemeinbildende Oberschule in Magdeburg
08/1990 – 06/1994:	Abitur am "Gymnasium Otto-von-Guericke" in Magdeburg (Schwerpunkte Naturwissenschaften und Englisch)
10/1994 – 12/1995:	Zivildienst im Krankenhaus "Altstadt" in Magdeburg
10/1996 – 08/2002:	Studium der Computervisualistik an der Otto-von-Guericke-Universität Magdeburg
11/1998:	Vordiplom in der Computervisualistik
03/2000 – 08/2000:	Berufspraktikum bei der Firma "MeVis" in Bremen
08/2002:	Diplom in der Computervisualistik, Abschluss mit Auszeichnung
09/2002 – 12/2002:	Wissenschaftlicher Mitarbeiter am Institut für Simulation und Graphik an der Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg
seit 01/2003:	Doktorand am Institut für Simulation und Graphik
05/2003 – 09/2003:	Forschungsaufenthalt am "Dept. of Computer Science" der Universität in Calgary (Kanada)