OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG

FACULTY OF COMPUTER SCIENCE
DEPARTMENT OF SIMULATION AND GRAPHICS

MASTER THESIS

Metadata Previewer: Exploiting Distortion Techniques for the Exploration of Watermarked Images

by

Nguyen Dinh Quyen

B.Sc., HCMC University of Natural Sciences, Vietnam, 2001

Supervisors

Prof. Thomas Strothotte
Henry Sonnet

OvG-Universität Magdeburg, Germany

July 2006

To my parents,

Nguyễn Hưỡng and Trần Thị Phụng

玉不琢，不成器。人不學，不知義。

Ngọc bất trác bất thành khí
Nhân bất học bất tri nghĩa

If jade is not polished,
it cannot become a thing of use.
If a man does not learn,
he cannot know his duty towards his neighbour.

TAM TỰ KINH [*]

[*] San Zi Jing, 三字經, or The Three Character Classic, was probably written in the 13th century and attributed to Wang Yinglin (王應麟, 1223-1296). It was a distillation of the essentials of Confucian thought expressed in a way suitable for teaching young children.

Abstract

This work deals with techniques for the exploration of augmented images which are images that have associated metadata encoded using illustration watermarking techniques. By exploiting the theories of image distortion and GUI design, an interactive tool for such exploration is developed in this thesis. The tool is built with the goal that it uses advanced visualization techniques to represent the metadata. To this end, a small widget is designed and implemented which is integrated in the original image and which moves with the mouse cursor. This widget which is called Metadata Previewer provides hints for watermarked regions as well as information concerning the metadata. For the demonstration, the tool is implemented in C++ using Troltech Qt® toolkit.

Keywords: *Illustration watermarking, augmented image, image distortion, focus+context view, fisheye lens, graphical-user interface design.*

Acknowledgements

First and foremost, I would like to express my sincere attitude to the supervisors, Henry Sonnet and Professor Thomas Strothotte, for the very creative ideas, the very helpful instructions, and the very fruitful discussions that I benefit for this thesis. It is the honor and luck for me to work with them.

Some backgrounds for the work come from the Hiwi job and the laboratory project that I worked with Thomas Vogel at the Research Group Multimedia and Security, OvG-Universität Magdeburg. I would like to thank him for the nice time and the useful studying.

I also would like to show my gratefulness to KAAD, the Catholic Academic Exchange Service in Germany, for the scholarship I got. They do always support, sympathize with me, and encourage me to finish this master degree in Germany.

Thanks also to all my friends, who are always by my side when I need them. Thank the close friend Vo Dinh Thanh and many friends at Duc An church dormitory, Pleiku, Vietnam. Thank uncle Tran Uy Nghi, a very nice man who has helped me so much in life and studying. Thanks to my colleagues at the University of Information Technology, Vietnam National University - Ho Chi Minh City. To Nguyen The Uy, a nice friend for always talking and playing games with me whenever I faced stresses and difficulties doing the thesis. And it is inattentive to forget Ho Le Ngan, for things, past and present.

Finally, my thanks are to my family – my parents, my brothers and my sisters for their endless love, care, and support. My parents are always the figure of hard working and the motivation for my studying. And last, but not least, I would like to put the acknowledgements to my aunt, Tran Thi Loan, who could never see my result in life and studying.

Contents

List of Figures

List of Abbreviations

EPS	Elastic Presentation Space
GUI	Graphical User Interface
HCI	Human-Computer Interaction
IWM	Illustration Watermarking
MIWS	Multilevel Illustration Watermarking System
MP	Metadata Previewer
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Motivation

Visualization plays a prominent role in our lives and activities where pictures and the whole visual world are indispensably used for our everyday communication and expression[1]. With the same manner, the development and use of information technology in our social and scientific lives are connected with the study of Computer Graphics and Human Computer Interaction; in which, the ultimate goal of Computational Visualization is "to make pictures a more flexible medium of presentation of information, and even to give pictures some of the degrees of freedom which language enjoys"[2]. In other words, computational visualization is importantly investigated and employed for the purposes of making the visual world on computer screen no longer just a set of pixels but the means of communication.

With the idea that "visualization offers methods for seeing the unseen,"[3] it gives chances for us to explore and enjoy the external world not only as its natural existence, but also with its connection to human knowledge. Visual thinking makes images the tools for communication and even tools for complex information space presentation. Pictograms, annotations, watermarks, among many others, are instances of knowledge dealt with in images. Here comes my matter for building systems that can be used for the exploration of visual data in

---

[1] I am writing this thesis during the time FIFA World Cup 2006 happens in Germany. All over the world, people do really enjoy it, express their feelings, show their cultures, and get communication through a visual world. I could not imagine how it would be if it is without visualization!

[2] See "*Computational Visualization: Graphics, Abstraction, and Interactivity*" of T. Strothotte [35] for more concepts.

[3] See "B. McCormick, T. DeFanti, and R. Brown (Editors), *Visualization in Scientific Computing and Computer Graphics*, ACM SIGGRAPH 21, 6, (November 1987)".

complex information spaces in dynamic ways. Let us examine them through the example of advanced world map exploration. It would be of benefit for us to develop a computer system that can support the users in their ways to interactively navigate over a map image and get the information about the history, politics, geography, transportation, climate, etc. of the different countries or geographical places in the map. Here, the map is by default a normal image, and the information data are unknown, i.e. invisible to the viewers; and the data are just shown on the map whenever they are requested by the users.

Illustration watermarking, the computer-related field which deals with the techniques for embedding, extracting, and displaying the watermarks information inside the images - focused by the researchers at the Otto-von-Guericke University of Magdeburg, Germany - is therefore applicable to that purpose. In this thesis, the development of an interactive interface for the presentation of the metadata presented as illustration watermarks is carried out as the main goal to be achieved.

From the very first GUI design[4], interactive applications have been developed and changed much, both with advanced techniques and wider range of applications. To adapt this complicated world of users and applied fields, more and more attractive, friendly, and beneficial visual interfaces have been established in an increasing number of computer systems. With that trend, visualization techniques are supposed to be investigated for the implementation of a visual tool, namely the *Metadata Previewer*, for the exploration of the annotation data which is invisibly attached to images. The challenge is to find solutions which are the improvements of traditional methods for the presentational task. And based on the characteristics of the system, it should tackle the following problems: (i) the explored image is a watermarked image but it is supposed to be seen as a normal one, (ii) the attached metadata can be any medium and is typically invisible to viewers, but it is exptected to be extracted in abstract forms and (iii) reasonably presented on the previewer following the criteria of GUI design.

1.2 Approaches

The aim of the work is to develop a tool that can be used for the exploration of the augmented images which have annotation metadata encoded using the

[4] NLS – oNLine System, developed by Douglas Engelbart and researchers at the Augmentation Research Center - Stanford Research Institute in the 1960s.

illustration watermarking techniques. Following the problems mentioned above, the proposed approaches to the work will be introduced in this section. For adequate navigation, the previewer is designed as a small cursor which can be moved around the image, and the visual data are generated and displayed on the cursor. There, the two main tasks needed to be done for the accomplishment of the previewer are (1) the identification of the watermarked region(s), and (2) the presentation of the annotation data.

The first task deals with the visual presentation of watermarked region. A specific view is created which integrates the markers and the highlight contour for the identification of watermarked places of the objects in the augmented image. Together with them, the original image is distorted for emphasizing the region which is focused. Distortion views are chosen as the advanced techniques for the representation of the visual data in complex information spaces, where the EPS model developed by Sheelagh Carpendale is proposed for the implementation of the tool. The visual decoration for objects' identification is suggested, following the concepts of GUI design in HCI.

The second task concerns techniques for designing and creating the graphical user interface for abstract metadata presentation. Extra widgets or geometric models can be used for such presentation; but in my case, small toolbars, namely the *information bars*, with icons and abstract texts are developed, since they adapt well the requirements for the task of interactive navigation of the previewer as a small cursor on the watermarked image.

Specifically for the two main tasks just mentioned, the interactive interface for the previewer is done through the following topics that I would like to resume once again here:

- Specifying the positions for the watermarked objects and representing them with the markers to assist the navigation process.
- Highlighting the selected watermarked object whenever it is explored.
- Using distortion techniques for the magnification of the selected region while the whole context remains.
- Developing the GUI templates for presenting the explored metadata.

1.3 Structure

The remaining parts of this document are structured as follows:

Chapter 2 presents the *fundamentals* and deals with the approaches in *related work* as the foundation for the later tasks. This includes the basic definitions of Steganography and Digital Watermarking, and the concept for Illustration Watermarking is then presented. Since distortion techniques are used for the creation and presentation of the previewer, the overview to computational visualization theories is given, and the selected Elastic Presentation Space is mentioned. The chapter ends with some Human Computer Interaction issues for Graphical User Interface design that will be employed in the work of generating and displaying the abstract metadata.

Chapter 3 comes after the fundamental chapter as my own work. This *concepts* chapter provides the analysis and the solutions to the problems having been addressed. It begins with the discussion to the model for creating and updating the previewer; and the detail methods for lens creation, objects identification and metadata presentation are then mentioned.

Chapter 4 talks about the *implementation* of the Metadata Previewer, and shows the demonstration results. They are done based on the discussed methods presented in chapter 3.

Chapter 5 goes on to the *evaluation* of the works, and opens some *future directions* as the result for the assessments and discussions. The chapter ends with the model of Multilevel Illustration Watermarking System as an overall architecture for the future development.

Chapter 6 finally ends the thesis with a short summary for the whole work.

Chapter 2

Fundamental and Related Work

In this chapter the state of the art of related work concerning this thesis will be presented. Section 2.1 gives the general introduction to the field of *information hiding* and the concept behind *illustration watermarking*. Section 2.2 covers the theory of *computational visualization* with respect to *distortion* techniques for image displaying. And finally, section 2.3 talks about the ideas of *visual interface design* in *human-computer interaction* study.

2.1 Introduction to Information Hiding

The rapid development of new media technologies and their applications and the spreading use of Internet in the last few decades have placed *information security* to be an active and growing research area in computer and related sciences, with two main directions of *cryptology* and *information hiding*. During a long period of more than 45 years which reached until the last decade of the previous century security area was just mainly about data encryption techniques, while information hiding came later in the 1990s.

The newer branch of data security is about the research for methods to conceal some secret data into the other data files. This happened due to the unavoidable requirements for copyright protection and state surveillance of entertainment industry, game distribution, distance learning, and other e-commercial applications. More and more research works and applications have been carrying out within the short period of about fifteen years up till now. The statistics from INSPEC, Jan. 1999, quoted in [33] shows that “103 papers dealing with watermarking appeared in 1998, whereas two appeared in 1992," for which, according to S. Katzenbeisser and F. Petitcolas, provides clear "evidence for the growing importance of steganography and watermarking," the two common disciplines of this research field.

Since the theory of information hiding does not make much sense to my work, I just want to give a general introduction to such computer security branch here, with concepts about *steganography* and *watermarking* as its main subdisciplines in section 2.1.1, some types of applications in section 2.1.2, and go on to the concept about *illustration watermarking* as the most related subject for the thesis work in section 2.1.3. For a good knowledge about the field, available and updated resources should be considered such as the books [1][7][33] with good definitions, techniques, and bibliography; websites [13][26][29] for links to conferences, community discussion; or theses such as [16][27]; among many other documents.

2.1.1 Steganography and Digital Watermarking

Based on the characteristics of hidden information and cover data, as well as the purposes of confidential communication in different kinds of applications, various subdisciplines of information hiding have been defined and applied for information security. Among that, the two main disciplines which got most attention from research are steganography and digital watermarking.

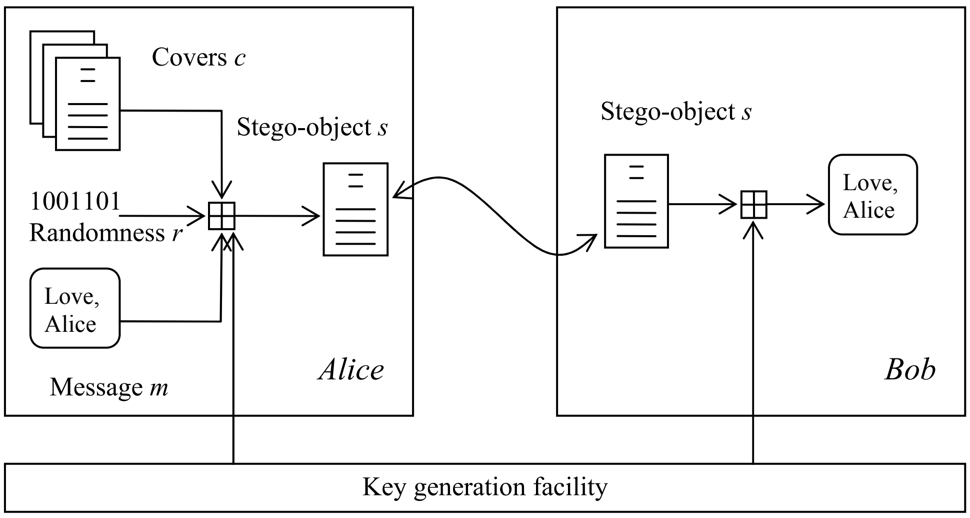


Figure 2.1 Process of hiding and extracting data in steganography. Courtesy of S. C. Katzenbeisser [33].

In *Steganographia*, the first treatise on steganography written by Johannes Trithemius in 1606, "*steganography*" stems from the Greek *στεγανός γραøειν* which means "covered writing". The idea is making data invisible by hiding (or embedding) them into another data, which is known as the cover, the host, or the carrier. The process of hiding and extracting data in a typical steganographic application can be illustrated in figure 2.1. A secret message *m* expected to be privately delivered from Alice to Bob is embedded into a harmless cover *c*, using a key *k*, called the *stego-key*. The modified cover *s*, including the hidden data, known as the *stego-object*, is then stored and transmitted to Bob in such secret communication. Using a relevant method and a key *k* that he knows before, Bob can then extract the message Alice wants to give him.

Historically, one of the first steganographic methods is the well-known story that Histiæus shaved the head of his most trusted slave, tattooed it with a message which disappeared after the hair had grown. The purpose was to instigate a revolt against the Persians [33]. Together with many others, Petiticolas also gives examples about steganographic techniques such as "sending a message to a spy by marking certain letters in a newspaper using invisible ink, adding subperceptible echo at certain place in an audio recording" as the recently secret communication ways. With the same purpose, new methods have been defined and applied for new digital media. The research in the scientific field that we mention here is about such methods as the result for this time of information technology.

The main focus of such typical steganographic techniques are the secret messages, while the covers that carry the messages are not meaningfully taken into account. It is therefore not necessary to consider the relationship between the cover data and the hidden messages. However, in recently applications, as presented in section 2.1.2, it is serious to consider the host data as well. That leads to the requirement for differentiating information hiding techniques into two main subdisciplines. The methods that just require the concentration on the secret message, but not the cover data, keep the definition as *steganography*; and on the other hand, the methods that care about the cover data are known as *digital watermarking*. The idea of watermarking comes from the use watermarked techniques in papermarking industry, especially in bank notes and stamps, where watermarks are used to protect the worthiness of the notes and the stamps.

For the investigation of information hiding techniques, a list of requirements should be tackled, based on the characteristics of the digital media and the applications, as followings - according to [7][1][30][33]:

Embedding Capacity: known as *payload*, is the amount of data that can be hidden in a cover, compared to the size of the cover. The feature can be measured numerically, for instance, in units of bit-per-byte (bpb).

Invisibility: termed *perceptual transparency*, or *algorithm quality* as well, is the measure of the amount of distortion to the cover. A large embedding capacity is useless if it causes large distortions to the cover. This feature is not numerically measured, but tied to human visual or auditory perception. For instance, in image watermarking, the visibility of the watermark may increase if the image is scaled.

Undetectability: or *security***,** is the feature about the ability that whether a hidden message is detectable or not. A steganographic technique is truly secure if knowing the exact algorithms for embedding and extracting hidden data does not help an unauthorized party to detect their presence.

Robustness: The embedded data can be altered or destroyed if the host data carrying them are processed in some ways. The measure of the ability that a steganographic algorithm retains the embedded data is known as *robustness*. In reality, there is no perfect method proposed so far, and it is not clear yet whether an absolutely steganographic method exists at all. Since the robustness just normally works against some specific processing tasks, but there are so many ways that the host data would be processed, intentionally or unintentionally, in computer applications.

In the case that an attacker intentionally tries to alter the embedded data in a cover rather than destroy them, the feature is specifically mentioned as *tamper resistance*, instead of robustness [7].

Oblivious vs. Non-Oblivious: In some steganographic methods, the original cover can be used for the retrieval of the embedded data. Such method is known as using *non-oblivious cover*. But in many other cases, for instance in watermarking applications for copy protection or indexing, the algorithms do not have access to the original data, and they are referred to as *blind* or *oblivious* methods.

Each of the requirements has its own importance in a specific algorithm. There is no method which can satisfy all requirements. In particular, embedding capacity, robustness, and undetectability are mutually conflicting and cannot be achieved by one algorithm. Figure 2.2 describes the relationships between these requirements. It shows that naïve steganographic methods can achieve large embedding capacity, but at the expense of robustness and undetectability. Advanced algorithms can achieve a high degree of undetectability, but offer small embedding capacity and insufficient robustness. Methods for embedding a watermark are normally designed to be robust, but result in small embedding capacity and questionable undetectability [7].

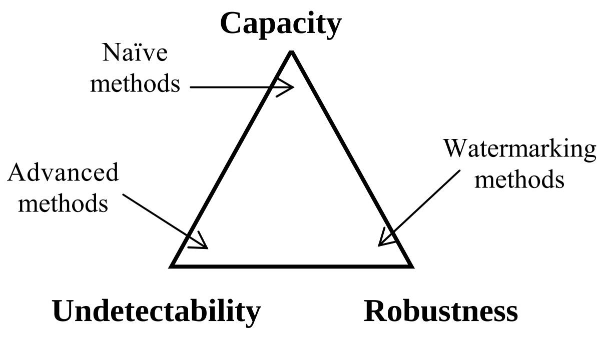


Figure 2.2 Conflicting requirements for data hiding, according to D. Salomon [7].

The detail investigation in information hiding techniques with their own requirements, for security applications, is really complicated but promising. We can find a general survey about those techniques in the the tutorial paper of P. Moulin [30], or M. Mihçak's thesis [27].

2.1.2 Applications of Information Hiding

Depending on the variety of multimedia technology and the development of steganographic methods, information hiding techniques have been investigated and applied to almost all fields of communication which require security. For that, applications in steganography and watermarking are innumerable and updated by time, and therefore, there would be no exact list about such applications. For example, in the book published in 2000 about information hiding techniques [33], M. Kutter and F. Hartung mention just four common watermarking applications. But in the tutorial paper presented last year in Proceedings of IEEE about "Data-Hiding Codes" [30], P. Moulin and R. Koetter give a review list of 11 modern applications. In general, I can summarize the applications of information hiding into four main categories as followings:

The first and most obvious applications of information hiding are *steganography* applications. As mentioned in the previous section, the embedded messages are the important data in consideration. One instance for this type of applications is doing secret communication over networks. In such application, people normally embed hidden messages into popular cover data and post them to public forums. Then the data are downloaded and the messages are extracted by the other people. Today such communication can be found in military and intelligence systems, or terrorists.

The second category is about applications deal with digital signature or identification in the cover data. They are purely *watermarking* applications. For instance, in *copyright protection*, watermark signature is embedded to valuable documents such as text, audio, image or video files, as the identification of the ownership, authorship, or other kinds of relationship between the data and a person or an organization. The important requirement for this type of application is the robustness of the algorithm against attacking. Another kind of watermarking applications is *copy protection*. Different with copyright protection, the main idea of copy protection is placing a watermark into the multimedia distribution which disallows unauthorized copying of the media. For monitoring or tracing back the unauthorized use of a document, another kind of applications to be considered in this category is *fingerprinting* or *traitor tracing*.

The third category is applications about *tamper-proofing*, or *forgery detection*, of cover data. Since modification of the cover would lead to the change of the hidden message, steganographic techniques can be used in authenticating multimedia data. The main consideration here is the detection of the attacking, not the processing which supposed to be already applied to the signal. For extracting information about *how* the signal has been altered, the application is known as *media forensics*.

Finally, dealing with the combination of steganography and watermarking techniques in computer science with the other fields, some other applications can be suggested and applied, which are to be grouped into this last category. For instance, the applications in *feature tagging* and *database annotation*, where captions, annotations, names of persons or places associated with the data are embedded to the cover, and later used in *data indexing* and *retrieval*. Though these last two categories can be used for both steganography and watermarking, they are just normally applied in watermarking applications. In the next section, we can get a clear concept about *illustration watermarking* as an instance for this last category of information hiding domain.

2.1.3 Illustration Watermarking

As presented, digital watermarking is used for a variety of security applications with different kinds of media covers, from text, to image, audio and video. Among them, digital image place the most important position. In many documents and publications, e.g. all cited resources used in this thesis, image is always the first subject for almost all watermarking techniques. The reason for that may come from the ideas that (i) watermarking was originally used in visual objects such as securing paper or money, and (ii) visualization can be seen as the very important mean for public communication. By the way, the hypothesis is out of scope of the thesis, and I do mention it here just for the grounds of illustration watermarking image concept.

Among the three main categories of digital image watermarking applications – copyright protection for cover data, forgery detection in content integrity, and data annotation, the last category promises the necessary for investigating further techniques and applications for displaying the annotation data. While in the first two groups, the crucial issue for the techniques is the secure communication and the means for data protection; *illustration watermarking*, the research field of the last category, recently defined by our researchers [15][36][38], on the other hand, opens a topic for the investigation of techniques for embedding and exploring the watermarked data.

Literally presented [15][38], illustration watermarking is the technique used to watermark hidden messages into distinct objects, instead of spreading them over the media as in normal watermarking techniques. For security purpose, the advance of this method in comparison to normal watermarking is its robustness against image cropping and therefore reserves the semantic characteristics of image objects. For computer graphics domain, illustration watermarking deals with the problems of retrieving the hidden information, by default, and displaying them according to the requests of the users. In this thesis, I will cope with the second direction for data presentation. Creating a system that can explore the watermarked data that can be used in interactive communication systems of e-learning, map exploration, medical illustrations, and other graphical-illustrating applications would be a beneficial work!

A model for illustration watermarking is illustrated in figure 2.3 with typical framework consists of three main modules: the *authoring tool* and the *illustration encoder* for the embedding part, and the *annotation browser*, or the *viewer* in the retrieving part. The process can be explained as followings. The authoring tool gets the cover image *c*, selects the objects to be watermarked, and chooses the information to be embedded. The most important part of the authoring tool is the task for selecting the objects. Applying to possible applications, such task can be done through a segmentation tool. An example is my BlobContours [37]. There, the user is supported to select the region of interest with the task of circling and pointing for segmentation refinement. After that, the illustration encoder is called to merge the annotation messages to relevant objects in the cover. Here, different techniques for image watermarking, for instance the *m*-band wavelet [39], can be applied. The resulting watermarked image *c'* can be then distributed to the users who use the annotation browser or viewer to explore the annotation information.

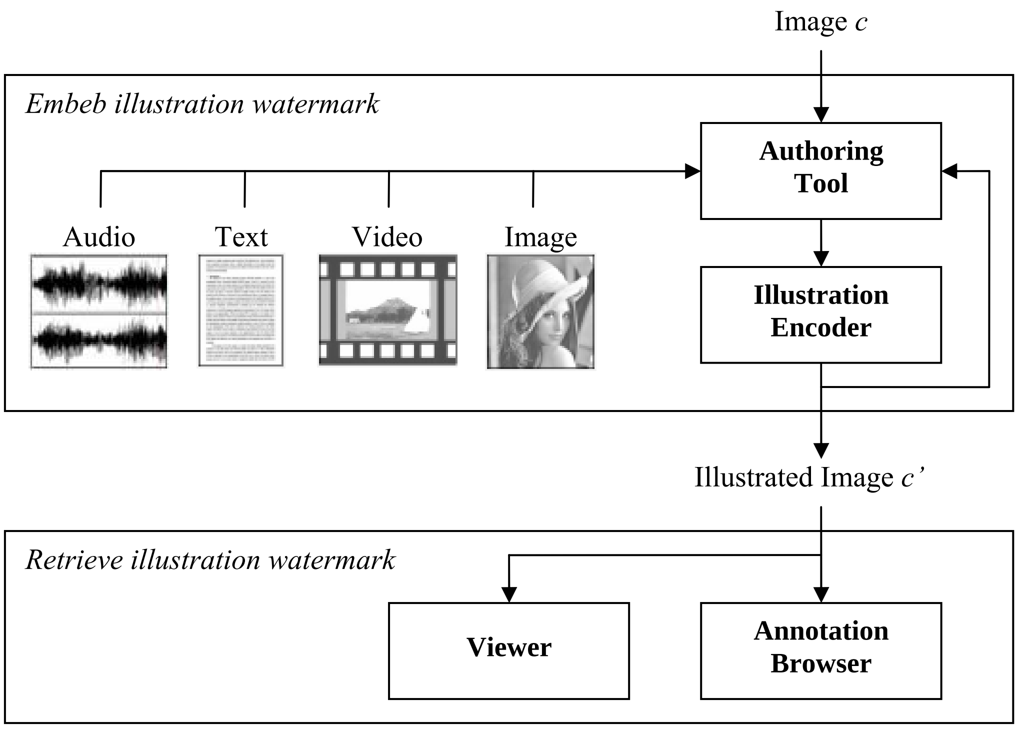


Figure 2.3 Model for illustration watermarking, according to T. Vogel [38].

Due to the requirements for adequate exploring and displaying the watermark data, the theories and techniques for computational visualizing and human-computer interaction design will be the central work of the thesis, as mentioned in the introduction chapter; and they will be discussed and presented from this point to the end of the document. And now, we will go on to the fundamental concepts for lens creation of the *Metadata Previewer*.

2.2 Distortion Techniques in Computational Visualization

The existence of recently high-tech displays, graphics cards and other peripheral devices and the investigation of Computer Graphics have put everything of our visual world onto computer screen. Nowadays, a user can easily enjoy the photographic world, manipulate his computer-aid works, and perform many other activities though numerous visual applications. For that, the demand for more and more interactive systems for comprehensible presentation and communication gets a great attention from the computer developers and researchers. Computer games, illustration graphics, and simulation visual models are some scientific fields cope with that direction which takes place through the research and studying program of Computational Visualistics at the University of Magdeburg, Germany.

Let us begin this section with the definition of Computational Visualization. According to Thomas Strothotte [35], it is the branch of computer-based techniques dealing with the process of exploring *complex information spaces* by generating the *abstractions* and *interactively representing* them through the communication between a user and an application. “The ultimate goal of Computational Visualization is to make pictures a more flexible medium of presentation of information, and even to give pictures some of the degrees of freedom which language enjoys”.

Information space is the central concept in that theory. It indicates that “the information available in any given scene or situation (needed) to be visualized is typically abundant and highly complex, even contradictory”, thus general model for graphical rendering of relevant information based on specific (interactive) application is necessary. Information data for that rendering include two main types: the data that define the shape and physical appearance of objects, referred to as *geometric model*, and the data that do not directly influence the appearance of the objects, known as *symbolic model*. And an information space is the *space* that includes both two models.

Based on the interactive exploring process from the user, data for geometric model and symbolic model should be adjusted as the basic process for computational visualization. The fundamental requirements in connection to the adjustment are the level of data abstraction and the relations of the rendered data. Generally, we do often need to represent the more important data and hide the less important one, and the data should be managed and displayed in hierarchical structures. By doing so, one can explore the necessary information in structure-based navigating process. Applying to my problem of exploring the watermarked images, we should create a model that can show the abstract information of the selected watermark region in the whole space, together with its metadata information. By the way, it is not very obvious to specify the geometric model and the symbolic model in this case. In my proposal, geometric model for the exploration of illustration watermarks image is the shapes of the presented image and objects, and the symbolic model is the types of widget and visual annotation with images and texts, though they can also be seen as the geometric model extracted from the metadata.

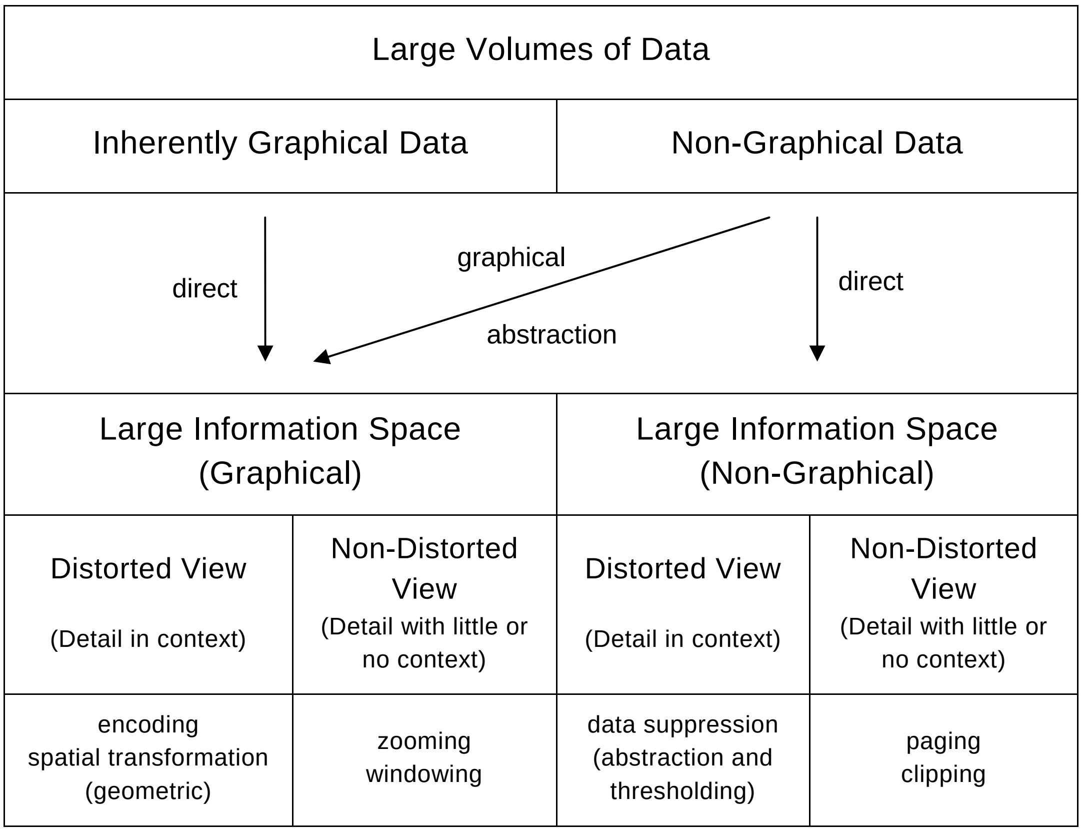


Figure 2.4 A taxonomy of presentation techniques for large graphical data spaces [40].

Basically, the idea for such abstraction and interactive model is driven by the theory that distortion views, among different presentation techniques, are better used for the exploration of complex information spaces. That theory comes from the concept of visual perception that our retinas get much more resolution at the center points, and lesser at the outer regions, relevant to the way we concentrate on an object when observing it in visual space. They are really promising, but applications of fisheye techniques are nowadays just mainly found in research laboratories, and almost not in the commercial domains. In the next subsection, I will give a short talk to the definitions of panning-and-zooming presentation, then go on to the basic theories for distortion views creation and representation in section 2.2.2. The Elastic Presentation Space framework of Sheelagh Carpendale - the main reference for my work - is then briefly given in the last subsection 2.2.3 to end this part.

2.2.1 Panning, Scrolling and Zooming

It is needless to say, we are all quite familiar with the task of using the mouse device to scroll a large image over the computer screen for watching. The common techniques of panning, scrolling and zooming are nowadays applied as the basic for graphical user interface design in almost all visual applications. In figure 2.5, the screenshot from Abode Acrobat® 6.0 is demonstrated. On the left *Pages* panel the "panning" map view is show as the indication for the visual content presented in the main client area. The user can zoom in or out to get the detail or the overall view of the screen page, and scroll over the page to get the other visual data.

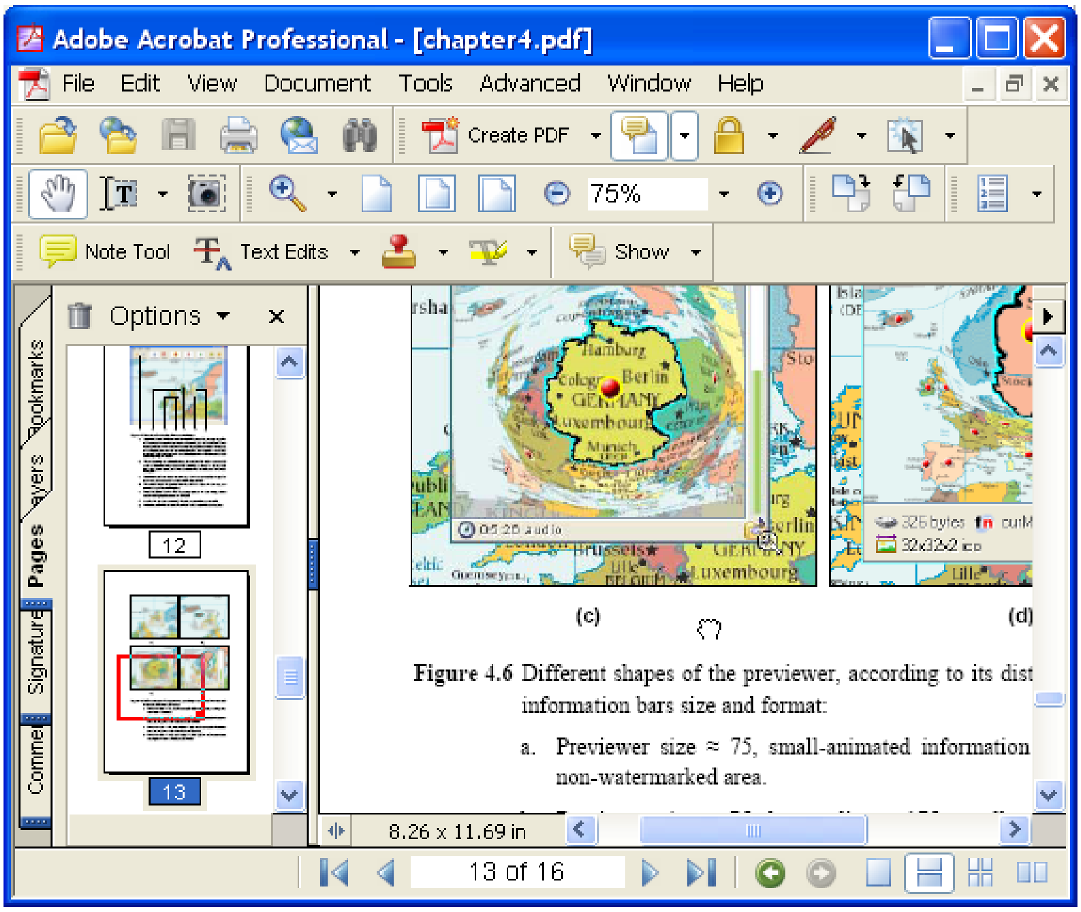


Figure 4.6 Different shapes of the previewer, according to its dist information bars size and format:

a. Previewer size ≈ 75, small-animated information non-watermarked area.

Figure 2.5 Adobe Acrobat® - An example of panning-and-scrolling GUI application.

The crucial issues of these interface techniques are the matter of pixels rendering and representing, and the matter of GUI elements for visual controlling. To give a good presentation, vector graphics with TrueType fonts, SVG (Scalable Vector Graphics) images, and other graphical objects are investigated and implemented. For raster graphics, techniques for anti-aliasing are required for a nicer appearance of the (scaled) images. They do together with the design of mouse cursors, toolbars, scroll bars and previewer widgets make complete the systems, depending on the requirements of the applications.

2.2.2 Distorting View

The normal techniques of panning-and-scrolling can generally be used for the presentation of any visual data on computer screen. But as we have mentioned, they are not the most sufficient techniques. In domains for complex information spaces, such of large hierarchically clustering networks, maps, or structured databases, pan-and-zoom techniques seem inconvenient in interactive exploring the information elements. In [9], Doug Schaffer et al. made experiments to show that distortion techniques can help to overcome such limitations. In comparison to full-zoom model for exploring the nodes and lines in hierarchical clustering graph, the feedbacks from their testers show that the user can quicker complete his tasks, and make fewer unnecessary navigational steps through the model of fisheye views.

Fisheye views are recognized as the basic for distortion views techniques, together with Bifocal Display [32], Perspective Wall [19], and Polyfocal Display [28]. The generalized fisheye views were originally introduced by George Furnas [14]. His idea is that we can assign to each element in an information space a *degree of interest DOI* as a number telling how interested the user is when he sees or explores that point in the space. The generalized fisheye views decompose the *DOI* into two components: *a priori importance API* and a dynamic distance *D*, and formularized as follow:

$$DOI_{fisheye}(x\,|\,.=y) = API(x) - D(x,y) \tag{2.1}$$

where *API*(*x*) is the global *Priori Importance* value which is statically predefined in the system, and *D*(*x*,*y*) is the distance between *x* and the current point *y*.

The *API* values and distances *D* are particularly specified depending on the fisheye applications. For instance, in an information space of map exploration, the priori importance of a city might rely on its inhabitants, political function (the capital city, for example), or industry zone, and the distance function is defined through the spatial, conceptual or temporal metrics to the others.

By applying the equation to each element in the exploring space, whenever we change the focus, the DOIs of the elements are generated and we can then get the new space presentation. Here, we face a matter of how to represent the DOI elements. In the model of a clustering network, a node can be scaled to some sizes, added with some detail data, or representative by an icon, based on the DOI value. Different ways can be tackled for such problem of emphasizing the important elements and deemphasizing the others of minor importance. In [10], Emmanuel Noik categorizes the emphasis techniques into three groups:

- *Filtering*: The elements with low DOI are simply left out. The process of filtering is almost always based on a hierarchical structure imposed on the information space, where entire substructures are left out.
- *Distorting*: This type of fisheye views reflects exactly the photographic nature of the fish-eye. It distorts the graphical presentation of visual data by manipulating the size, position, and shapes of objects in accordance with their DOIs.
- *Adorning*: The presentational elements are emphasized or deemphasized through the modification of their colors, line styles, fonts, transparency, or other visualizing parameters.

In his introduction to fisheye views, Furnas illustrates the theory with simple examples for navigating tree structures and source code texts using filter techniques. The concept for source code navigation is that the importance of a code line is based on the hierarchical structuring of a (procedural) program. For instance, the first and last lines of procedures and statements are important, while the other codes are less important if they are out of scope of the focus region. In figure 2.6, a “distorted” C code program is demonstrated to show that fisheye technique makes the view more compact (to a half), and therefore, easier to explore.

Based on the generalized model, different fisheye view techniques have been investigated and applied to some application domains [8][17][23][24][4][5]. Among that, the *graphical fisheye views* of M. Sarkar and M. Brown [23] made the first contribution to the graphical presentation with the mathematical formalism for element positions computing in displaying plane, as the essential concept to my problem of image distortion.

```
   1 #define DIG 40
   2 #include <stdio.h>
...4 main()
   5 {
   6     int c, i, x[DIG/4], t[DIG/4], k = DIG/4, noprint = 0;
...8     while((c=getchar()) != EOF){
   9        if(c >= '0' && c <= '9'){
...16       } else {
   17          switch(c){
   18             case '+':
...27             case '-':
...38             case 'e':
 >>39                for(i=0;i<k;i++) t[i] = x[i];
   40                break;
   41             case 'q':
...43             default:
...46          }
   47          if(!noprint){
...57          }
   58       }
   59       noprint = 0;
   60    }
   61 }
```

Figure 2.6 A fisheye view of the C program. Line numbers are in the left margin. “>>” indicates the focus point and “...” indicates the missing lines [14].

Starting from the undistorted layout (the normal view), two types of transform functions was suggested for the reposition of the visual points in the distortion views: one based on a Cartesian coordinate transformation and the other on polar system. The transformation and magnification functions for 1-dimensional fisheye view are presented in figure 2.7a and 2.7b, with *M* is the derivative function of *T*.

$$T(x) = \frac{(d+1)x}{dx+1} \quad \text{and} \quad M(x) = \frac{(d+1)}{(dx+1)^2} \tag{2.2}$$

where *d* is called the distortion factor and *x* is the normalized distance from a point under consideration to the point of focus.

The larger *d* is, the bigger the magnification and the amplitude of the peak in the magnification function. This shows the characteristics of focus and context in distortion views. The value of *x*, and similar *y* for coordinates, can be normalized into interval [0,1] where if x=0, the point under consideration is at the focus point, and if x=1, it is at the a position furthest away from the point of focus on boundary.

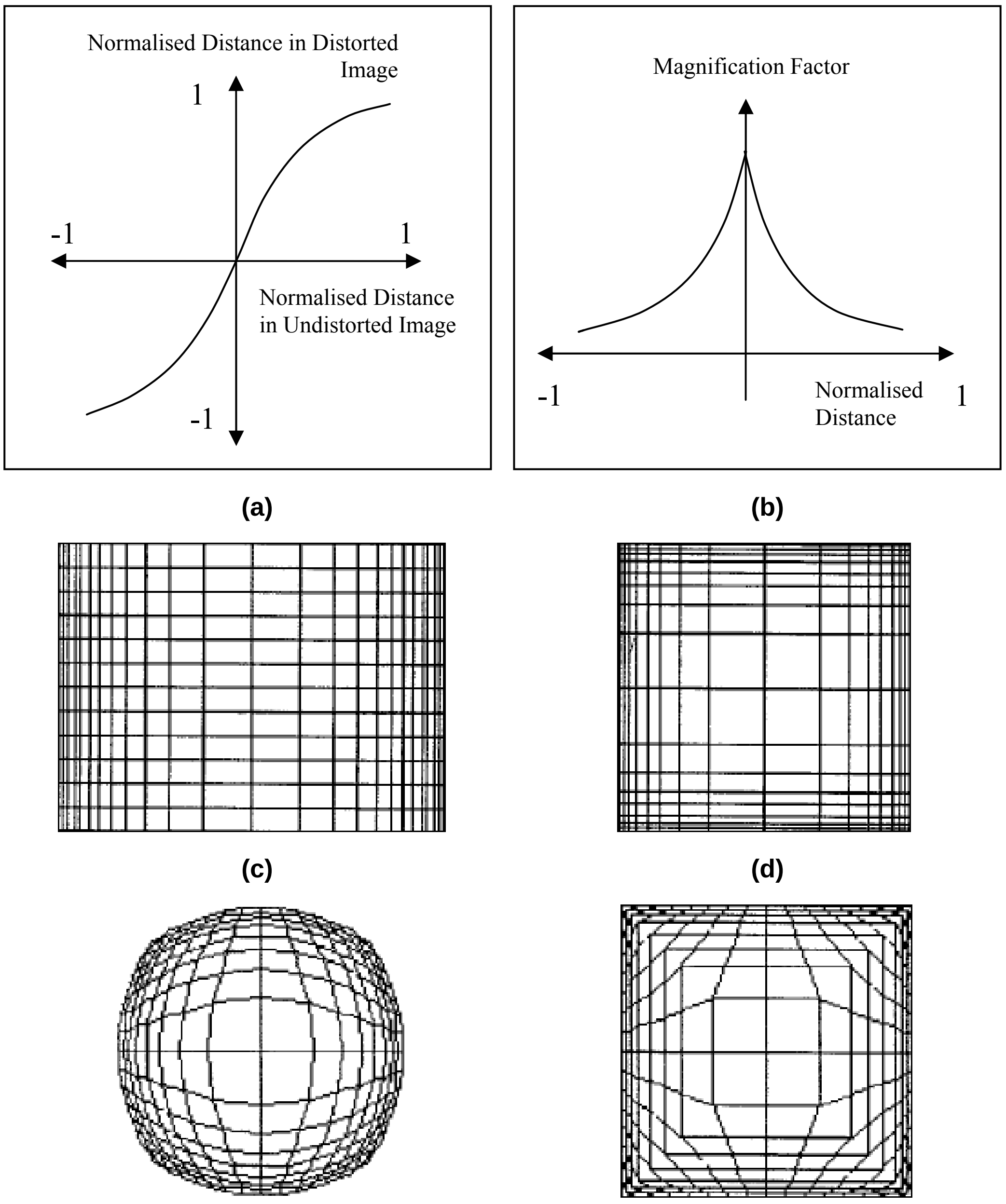

Normalised Distance in Distorted Image
1
-1
1
Normalised Distance in Undistorted Image
-1
Magnification Factor
-1
Normalised Distance
1
(a)
(b)
(c)
(d)
(e)
(f)

Figure 2.7 The Fisheye View: (a) a typical transformation function; (b) the corresponding magnification function; (c) the application of Fisheye View in one dimension; (d) a Cartesian Fisheye View in two dimensions; (e) a polar Fisheye View; (f) a normalized polar Fisheye View [40].

Figure 2.7c and 2.7d show the applications of fisheye views in one and two dimensional with Cartesian coordinate system, and figure 2.7e with the transformation based on a polar coordinate system. Here, we see that the polar fisheye view produces a rounded appearance which unfortunately does not provide a nature look when implemented on a rectangular screen, known as the *screen real estate problem*. Sarkar and Brown proposed further that the rounded appearance of the polar fisheye view can be remapped on a rectangular space. The result for such modified transformation is illustrated in figure 2.7f.

This method of Sarkar and Brown can be applied to the problem of image distortion for watermarked objects as the main work in my thesis. However, it is inadequate in providing methods for adjusting the different presentation forms of the Metadata Previewer (see the next chapter for the options of lenses creation). Besides, the analysis of smoothly integrating the discrete points in the focus and context regions of the image, and non-linear transformation functions are also some other matters in consideration for the implementation. The Elastic Presentation Space of Sheelagh Carpendale [24], on the other hand, provides flexible techniques to all those matters. Thus, it is chosen for my analysis and implementation of the demonstration tool.

2.2.3 The Elastic Presentation Space

Different from the above approaches which purely consider the distortion views as the two-dimensional transformations of visualizing planes, a distortion view in the *elastic presentation space* (EPS), according to Carpendale, is defined as a perspective projection of a surface constructed in three-dimensional space onto a two-dimensional plane through a viewpoint and a view plane. The general model for that idea is presented in figure 2.8. What we want to see for the complex information space is a visual plane, supposed to be the *base plane*, which is larger than the displaying area that we have, i.e. the *view plane*. Putting them together into the same view-point, one can imagine that they have the same reference viewpoint as illustrated in the drawing. Traditionally, by shifting the view volume in that space paralleling to the base plane, the visualizing data presented inside will be scaled into a similar version with relevant ratio. The view plane would be the scaled view of the base plane. Keeping that idea, but instead of scaled shifting, we distort the visual points in the base plane by computing their new values, using some geometric functions to get the expected presentation.

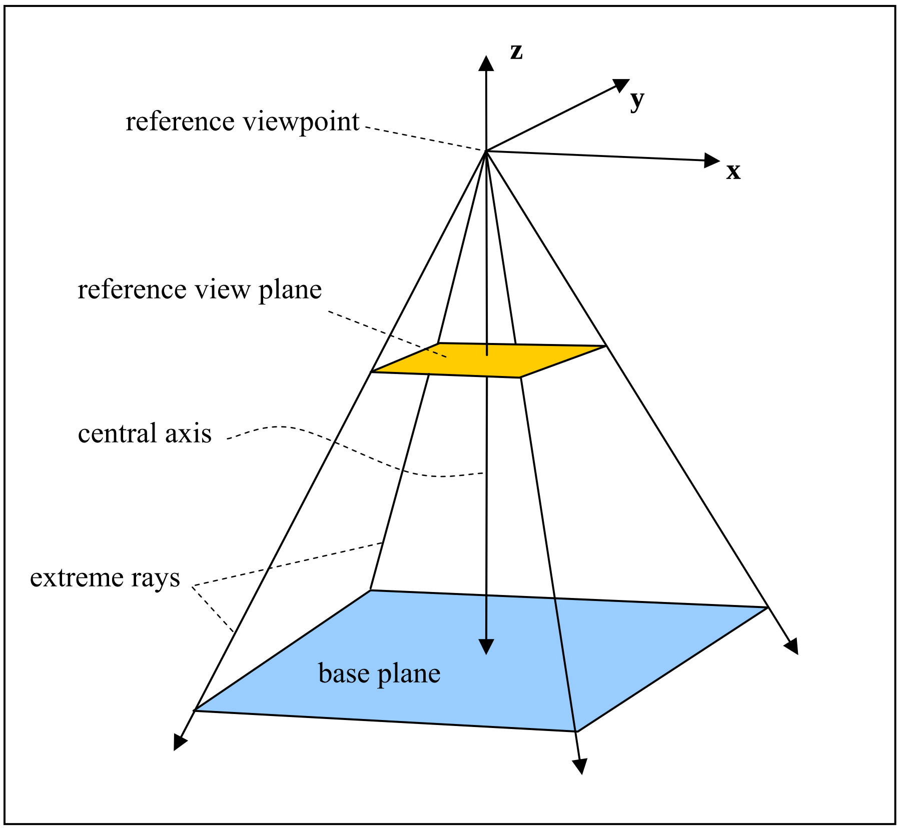


Figure 2.8 Three-dimensional model for elastic presentation space [24].

A cross-section for that distortion manipulation is presented in figure 2.9. At the top of the triangle is the reference view point (RVP) and at the bottom is the base plane. At the left of the triangle is the central axis and on the right is an extreme ray or the edge of the viewing volume. Considering that the visualization data is presented on a surface. In its normal position, the surface is the base plane, and there is no distortion in the presentation. What we need to do for the distortion view is magnifying a chosen focus, and compressing the surrounding area in the surface for the necessary space.

The task for that magnification is quite simple. Translating the surface in *z* towards the RVP will lessen its distance and increase the magnification. If raising the whole surface uniformly with the central point, what we see from the RVP is a zoomed visualization and the outer area is lost. Thus, we need to manipulate that surface in the way that the focus point is raised but the whole context connected to the base plane is remained to get the distortion presentation. Here, the most important factor that affects the shape of the distortion view in the EPS model is the drop-off function connecting from the focus point to the outer context (the base plane).

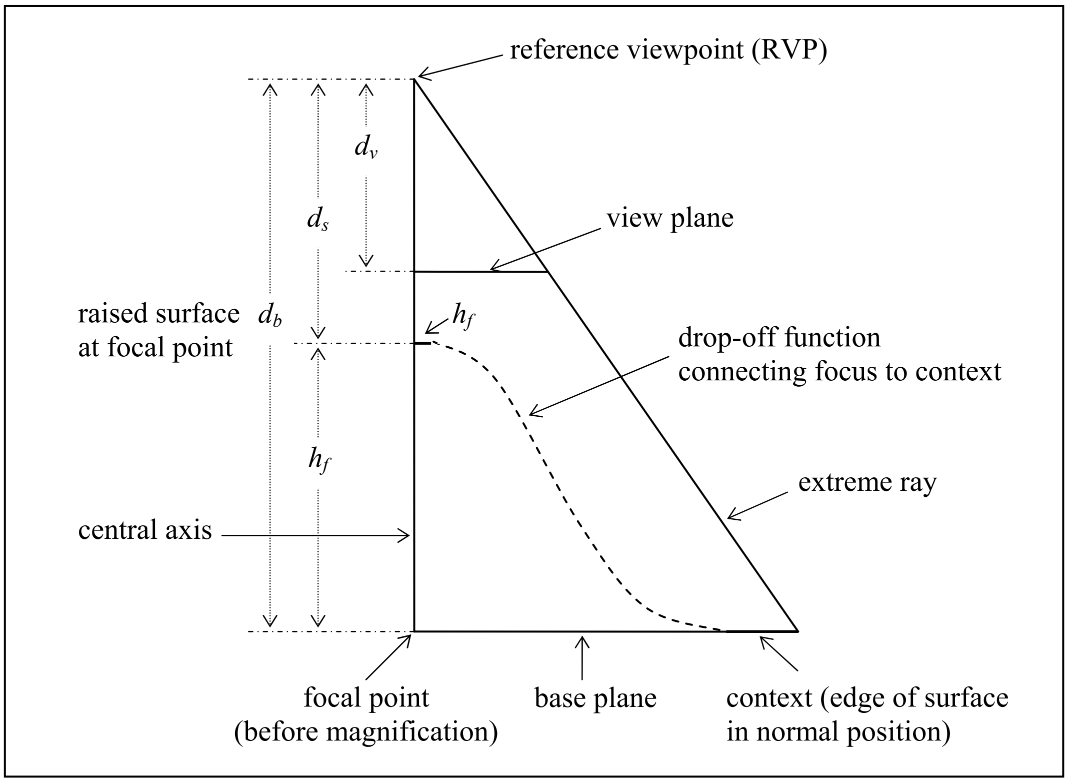

reference viewpoint (RVP)
d_v
d_s
view plane
raised surface at focal point
d_b
h_f
drop-off function connecting focus to context
h_f
extreme ray
central axis
focal point (before magnification)
base plane
context (edge of surface in normal position)

Figure 2.9 A cross-section diagram showing the raised central focal point, the profile of the drop-off function and the edge of the surface in normal position [24].

In detail, at the central focal point, the surface is translated in z to a height h_f, and on the base plane, the edge of the surface is still in its normal position. They are the two end-points of the dash line of the drop-off function shown in the above figure. Changing the drop-off function will change the appearance of the surface, and therefore, change the distortion view. In figure 2.11, several drop-off functions with their distorted lenses are shown to clarify that idea. Among that, the Gaussian drop-off function is the one that produces the best shape for the distortion view, since "it combines the advantages of gentle focal integration with those of gradual integration into the remaining context" [24, §5.5]. In my illustration metadata previewer, this function is used for the distortion lens creation.

The general formularizing model to visualize a normal image as a distorted image is a 2D-to-2D transformation (in fact, it is 2D-to-3D-to-2D transformation). It means that what we have on the displaying device is always a two-dimensional matrix of pixels, and the concept of three-dimensional surface is used just for the data manipulation. The explanation of that transformation is as followings:

Translating a point (x_i,y_i) from the base plane to the elastic surface, the new position will be (x_h,y_h), as presented in figure 2.10. h_p is the height from (x_i,y_i) to

(x_h,y_h), and d_p is the radical distance for (x_i,y_i) from the focus centre f_c. The formula for such Gaussian drop-off height is:

$$h_p = h_f \cdot \exp^{-\left(\frac{(d_p)^2}{\sigma}\right)} \tag{2.3}$$

where σ is the Gaussian standard deviation.

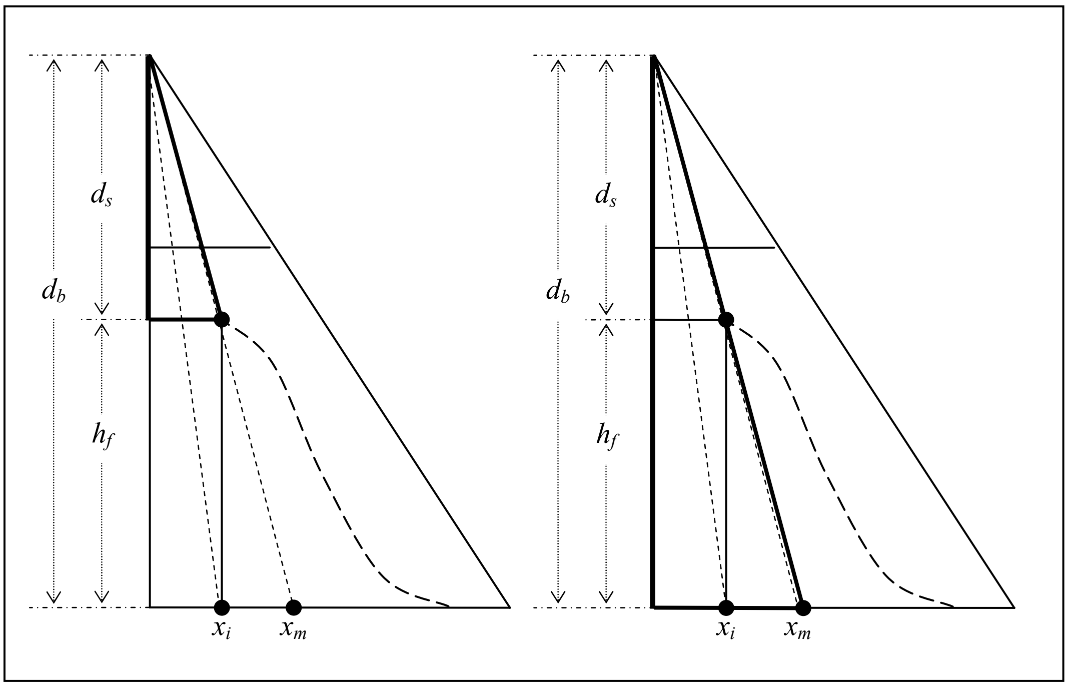


Figure 2.10 Similar triangles show the relationships between z-translation and magnification [24].

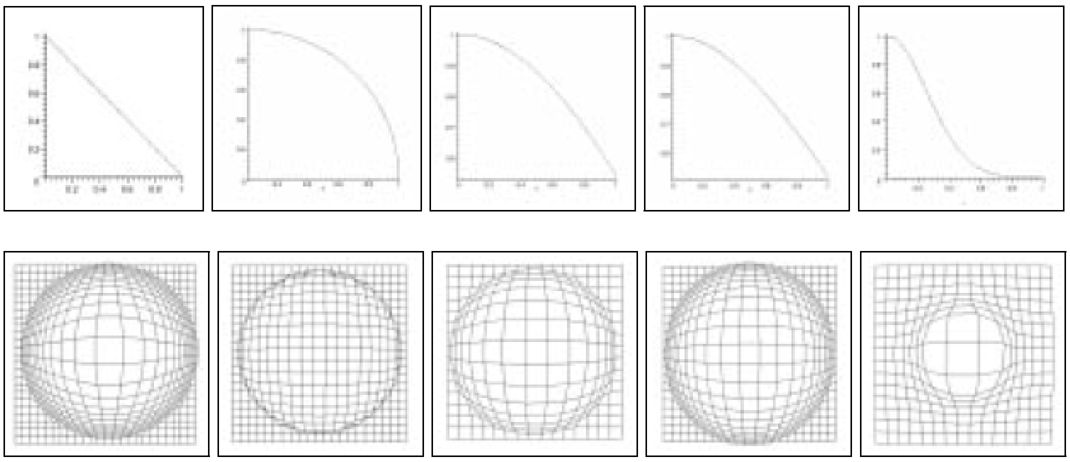
When viewed in perspective from the RVP, the focus appears magnified. The point (x_h, y_h) will be seen at position (x_m, y_m) on the base plane. It means that the projection moves a point in the original image from (x_i, y_i) to (x_m, y_m) in its distorted version. Having an input image with a matrix of pixels, we can compute their new positions, considering the similar triangles in the figure with formulae:

$$\frac{x_m}{d_b} = \frac{x_i}{d_s} \quad \text{and} \quad \frac{y_m}{d_b} = \frac{y_i}{d_s} \tag{2.4}$$

or

$$x_m = x_i \cdot \frac{d_b}{d_b - h_p} \quad \text{and} \quad y_m = y_i \cdot \frac{d_b}{d_b - h_p} \tag{2.5}$$

where d_b is a constant in the view, and h_p is the drop-off height for (x_i,y_i), computed according to equation 2.3.



(a) Linear drop-off function (b) Hemisphere drop-off function (c) Cosine drop-off function (d) Hyperbolic drop-off function (e) Gaussian drop-off function

Figure 2.11 Graphs of the drop-off functions and the top view of their lenses [24].

In Carpendale's thesis, full investigation and discussion about lenses creation and manipulation of EPS is addressed. The lenses can be changed with a freedom of focal shapes and magnification levels. The focus can be a single one or a multiple foci, with circular or arbitrary shape, and flexible in the location projection. The magnification level can be adjusted in some possible manners to give a better usage for the distortion lenses applications. Applying to my Metadata Previewer, together with some implementation issues, the selected magnification level and the use of scaled-only focal region are the matters for studying as presented in the next chapter.

2.3 Visual Interface Design

Together with the techniques for fisheye lenses creation as the basis for the watermarked region specification, the background of visual interface design is the second crucial point needed to be mastered in this thesis work. As presented in the next chapter, the *Metadata Previewer* is used for the displaying of the explored information, thus it is supposed to be designed as well-looked, friendly, and informative as possible. They are the criteria for a navigating tool used in complex information spaces.

In theories of complex information space, visual data are the means of communication, in which abstract images and other visual formats can be used to give more useful information than the abundant data from which they are derived. Without caring about their models (geometric or symbolic model), *abstractions* are employed in my thesis as the matters of visual interface design. Since the abstract information is so different in form to its original data, detail investigations on the extraction process should be considered. Information science, data management techniques, human factors, psychology, philosophy, user case study, image science, and of course computer science, are all needed for the full work. This is a big job. Thus, within the scope of the studying, I do give just some concepts about abstract visual data for the design of GUI applications, but not go into details about the way the data are extracted (from the metadata files), how the data fit the user requirements, or whether they are in the best form.

The next subsections are organized as follows. In 2.3.1, the general theory of visual interface design with the basic requirements for the GUI, and the matters of graphical abstraction are presented. Later, I go on to talk about some visual objects that will be used for the presentation of the Metadata Previewer's information bars. Section 2.3.2 shows the concepts for multiple items creation and management. There, menu, piemenu, and toolbar, and tabs are presented. The later subsection 2.3.3 is about icons and abstract texts as the detail objects for information data displaying. Quantitative visualization is the matter for the presentation of the comparative searching data, thus mentioned in section 2.3.4. And finally, section 2.3.5 ends the chapter with the ideas about interactive presentation, with the matter of display rate for toolbars animation.

2.3.1 Graphical User Interface for Data Exploration Domain

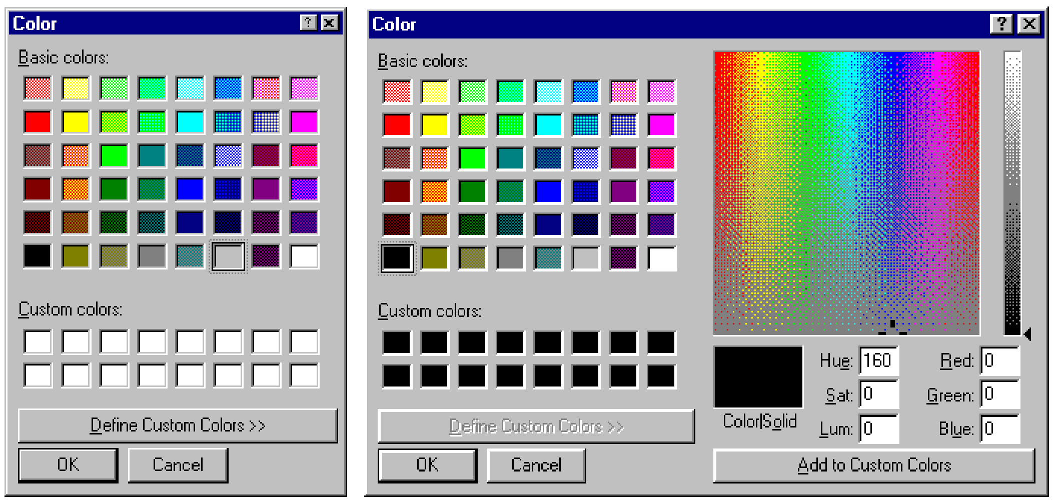
According to Ben Shneiderman [3], the motivations to the interface design for interactive systems vary in different application domains. In life-critical systems of air traffic control, power utilities, military operations, etc. the most crucial requirements are the reliability and effectiveness of the interface; in industrial and commercial systems, costs shape, ease for learning and speed are the criteria for consideration; and so forth. Thus, an application for data exploration like the Metadata Previewer should also have its own constraints to be achieved. In my opinion, the following principles should be taken into account concerning the *content*, *interactivity* and *navigability* of the system:

- *Efficiency*: This is the basic requirement of an exploration system. The interface is designed to supply a better way for the user to get the necessary information in a huge, and hidden – in the case of the MP, data collection, instead of manually finding the data.

- *Correctness and completeness*: Normally, a searching domain does temporarily find the similar data for the expected result, where relative metrics are applied for the similarity measuring. For that, correctness and completeness are not so important for the system, but they are expected to be achieved as much as possible. This principle adapts the concepts of graphical abstraction in visual presentation.

- *Friendliness*: From the very first GUI applications, friendliness is the main goal of visual designing. The more convenient the application can support, the more popular it is. However, this criterion is not fixed, since it depends on different situations of the user communities, current techniques, and its life line.

The visual content, as mentioned, should be compact in the complex information spaces. Compactness supports the user to get the necessary information out of the complex structure of the data that he or she may be tired of if watching in their original form. However, visual abstracting is a hard process. The content may be changed or lacked of information if represented in new format. Therefore, choosing a possible type of visual data is the first step to be tackled. Icons, pictograms, quantitative charts, diagrams are some solutions that can be used. In the subsections 2.3.3 and 2.3.4 we will learn some more about those visual objects.



Hue: 160 Sat: 0 Lum: 0
Red: 0 Green: 0 Blue: 0

(a) (b)

Figure 2.12 Example for (a) normal and (b) custom color selection in MS Paint®.

With the necessary visual data, the second point will come is the matter of decorating them on the interface. Visual layout is in consideration. We should find solution for the position and template for the arrangement of the design items. Here, the requirements of compact form are also remained for the decoration. For instance, in many dialog boxes, the *advanced* options are often added beside the *normal* interface with the meaning that just the frequently used options are presented so that the user can easily choose them, but the advanced options should also be amended for the further adjustments. In figure 2.12, an example of normal and advanced custom dialog for color selection in MS Paint® is illustrated. The arrangement of UI widgets is done based on the widgets themselves, where windows, icons (for abstraction, as mentioned), menus and a pointing device, which typically is a mouse, for short WIMP, are known as the basic elements which make concepts of GUI design for decades.

At this moment, the works for such WIMP GUIs are just mainly about "making the visual interface *realism*". It is friendliness feature. The idea is to make the applications easy to use, easy to learn, easy to imagine and easy to remember. Inventions for new UI widgets are no longer the major tasks, but the tactics for refining the items are carried out. Shadow, 3D effect, highlighting, animation, etc. which aim to increase the user attention are the goals for mostly nowadays GUI designers. An application for data exploration, therefore, should also not neglect this issue.

2.3.2 GUI Layout and Widgets for Multiple Items Selection

In fact, the task of laying out the visual interface does not take place just when the visual items exist, but they are concurrently analyzed and established in a GUI design process. A circle of *conceptual*, *semantic*, *syntactic* and *lexical* levels is carried out to fulfill the layout model [3]. There, the system designers have to get the user's mental for conceptual level, analyze the meanings of what he or she can benefit from the input commands and the output display, define the syntactic structure for visual items that will perform those tasks, and make real the layout based on the devices and the working mechanisms. In the case of the Metadata Previewer, the analysis for such process is not clearly mentioned, since the visual interface is just specifically defined for the exploration of hidden metadata accompanying with the distortion view.

With an analyzed visual model, different interaction styles can be chosen for the visual form, which are *menu selection*, *form fillin*, *command language*, *natural language*, and *direct manipulation* [3]. Menu selection and direct manipulation are the two styles that will be used in the design of the MP. In the following paragraphs, I will shortly introduce some UI widgets for the menu selection style.

Pull-down menu is the most common type for multiple items selection used in almost all GUI applications. A menu simply consists of a list of texts and symbols combined together, representing the command actions. By clicking or keyboard-accelerating one of the actions, the system will perform the instruction that the symbol represents. The advantages of menu are shortening the interaction between the application and the user and managing the actions in hierarchical structure.

A shorter, simpler form for menu item selection is the **toolbar**. A toolbar normally contains just the icons representative for the command actions, thus it looks nicer and occupies a smaller space on the screen. In some applications, when there are so many tool actions, pull-down list for the tool items is supported for other selections. The idea is illustrated in figure 2.13b.

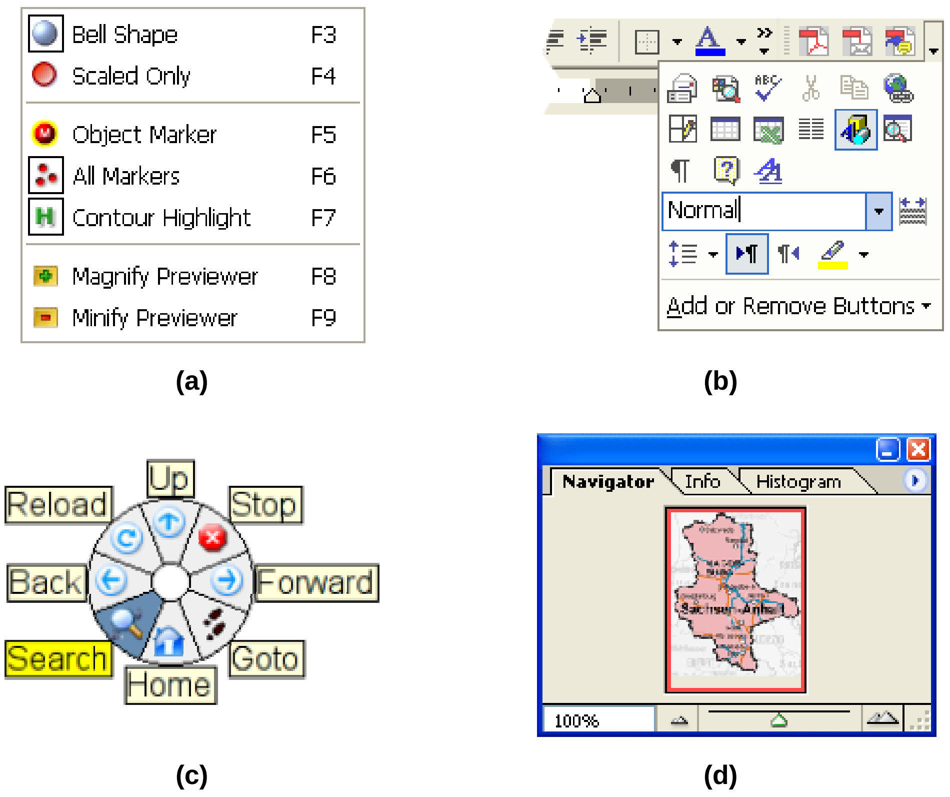


Figure 2.13 Example for GUI widgets (a) pull-down menu, (b) toolbar with extended list for other actions, (c) piemenu, and (d) tabs and view panes.

Different with those rectangular and linear order menus, another kind of visual menu that places action items in a circle around the clicked point, namely

piemenu, is supposed to provide better navigation for the user. To access an item, the user moves the mouse at different angles to reach different options, rather than moving it different distances, therefore the navigation is faster. Angles are also easier than distances to hold with muscle memory as characteristic of friendliness characteristic. A normal piemenu often consists of 4 or 8 option-items. For more items, nested piemenus or piemenus with popup-linear ones can be efficiently used. In figure 2.13c, a simple 8-options piemenu for internet browser is demonstrated.

The above mentioned menu widgets are used for the interaction commands without considering the presentation of the visual data. Thus, they are not completely applied for my MP, since what I inherit from them are just the ideas of visual layout. With this issue, **tabs** for multiple view panes with short texts and icons for actions selection can be a good solution for the presentation of the complex extracted metadata. They will be mentioned again in the next chapters for the detail discussions of MP's design as my central work, and of Cascading Previewers as the suggestion for future work.

2.3.3 Graphical Abstraction: A Way of Visual Thinking

In [6] Christine Strothotte and Thomas Strothotte classify pictures into three main relational categories of presentational pictures, abstract-graphical pictures and pictograms. Here, picture, according to The American Heritage® Dictionary, is defined as "a visual presentation or image painted, drawn, photographed, or otherwise rendered on flat surface". In detail, "presentational pictures present properties and relations in reality (including virtual reality and imagination) which are visible to humans". They are, normally, photo-realism. Abstract-graphical and pictograms, on the other hand, are quite different. "In abstract-graphical pictures, properties of and relations in reality which are invisible to humans can be presented in an abstract way". Graphical symbols such as geometric primitives (rectangles, ellipses, cubes, etc.), arrows, lines, and text labels are combined together, based on the appearance of the normal presentational pictures to form the abstract-graphical pictures. And, "picto-

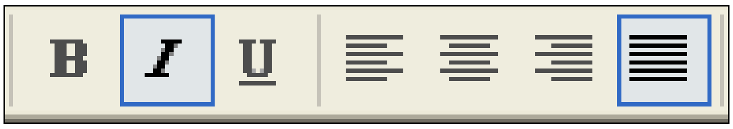


Figure 2.14 Icons, the way for visual pictogram that effectively showing the necessary information of text types (bold, italic, underline) and different text alignments for paragraph in MS Word®.

grams represent something more abstract than what they can actually show". They are used with the purposes of giving some specific visual information, and can be seen as a language for communication.

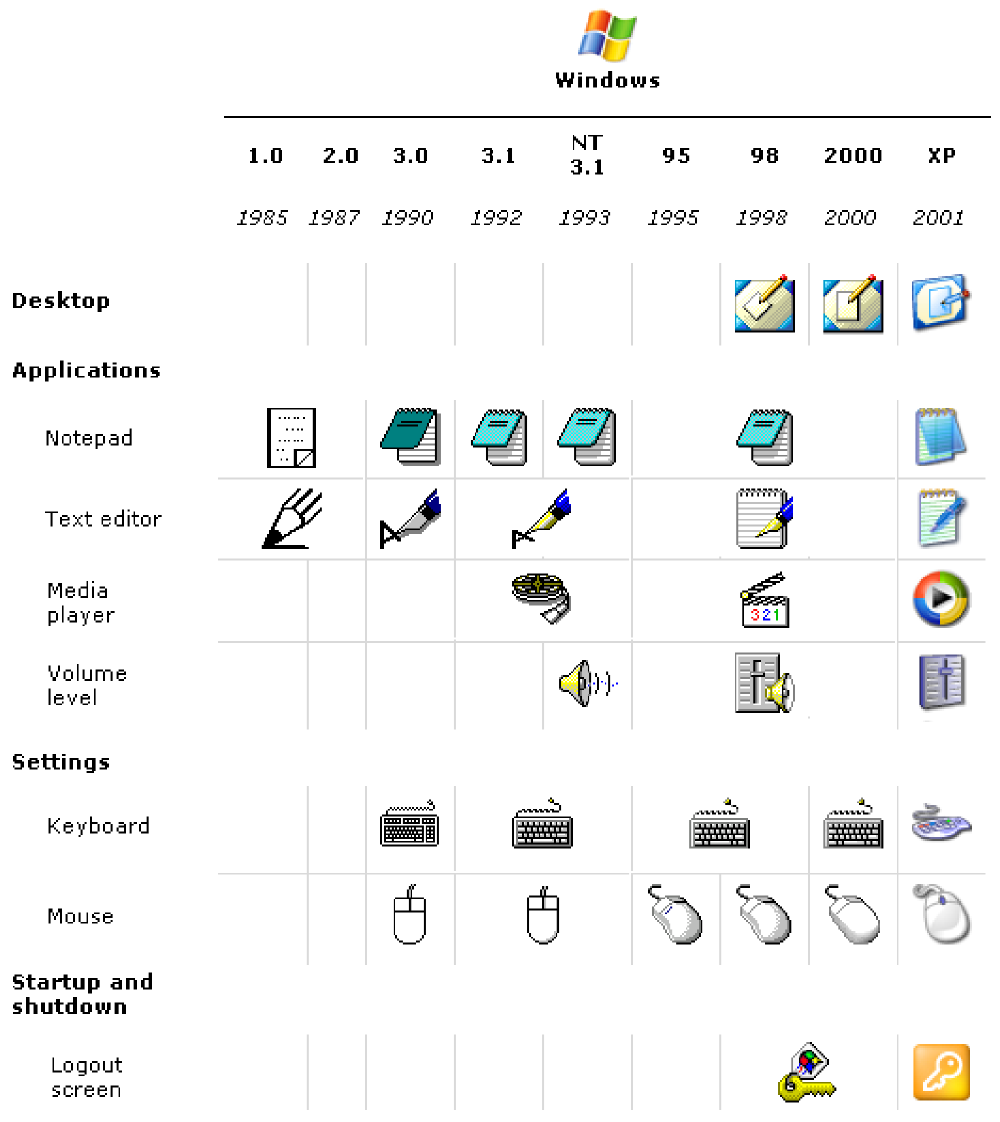


Figure 2.15 A small collection of icons historically used in MS Windows® [25].

Visual thinking, as presented in [3], is therefore the concept behind such graphical abstractions. Applying to the context of exploring complex information space, the visual presentation in form of abstraction gives a better interface for the interaction communication than showing a huge amount of explored metadata. Thus, pictograms, short texts, and icons are the suggestions for the design of their GUIs. Since the creation of pictograms and icons do normally deal with art and designing tactics, it is not the topic for my discussion. Anyway, a general look to the history of icons used in computer systems would be necessary for the icon shapes created in my MP. In figure 2.15, a small collection of some popular system icons used in Microsoft Windows® is presented. From their very first versions, Windows icons look like pictograms, where representative information is the main goals. But later, with the support of better hardware and based on the interest of users, the icons look more and more *realistic*.

2.3.4 Quantitative Visualization

In many documents, the presenting data are not shown in their conventional forms of texts, numbers, or discrete images, but in forms of graphical charts, maps, or graphs. They are known as data graphics - a kind of abstract, non-presentational pictures. According to E. Tufte [11], data graphics are instruments for reasoning about quantitative information. A well-designed data graphic is usually the simplest and at the same time the most powerful way for describing, exploring and summarizing the complex and abundant normal data. Designing data graphics, similar to any visual abstracting process, is therefore not an easy task at all. In this small section, some main issues concerning the characteristics of data graphics are shortly mentioned through an example of chart bar drawing, as followings:

- *Complexity*: The most crucial point for data graphics, in comparison to their conventional forms is the complexity of the data. The data are always expected to be designed as simple as possible. To do that, the normal way is reflective mapping the data into a common visual form of data maps, time-series, space-time narrative designs, or relational graphics. The redundant data-ink is always supposed to be removed. For instance, in figure 2.16, the altitude of a bar is redundantly shown in "(1) height of left line, (2) height of shading, (3) height of right line, (4) position of top horizontal line, (5) position (not the content) of the number at bar's top, and (6) the number itself" [11]. But which redundancies are not important in this case?

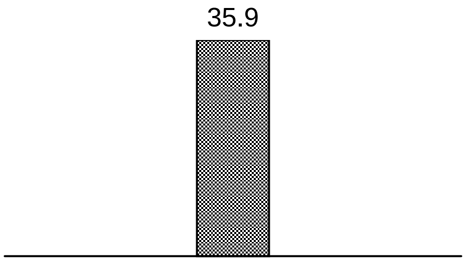


Figure 2.16 Redundancy in this graphical chart? [11].

- *Comprehensibility*: The answer is the remaining bar should show the basic visual information of the chart. We may need just a line (vertical or horizontal) for the altitude of the bar, but it might cause misperceptive to the viewer. The graphics would be miscommunicative. As the characteristics of visual presentation, the comic shown in figure 2.17 artfully expresses that idea. That comes to a principle: "Erase redundant data-ink, within reason."

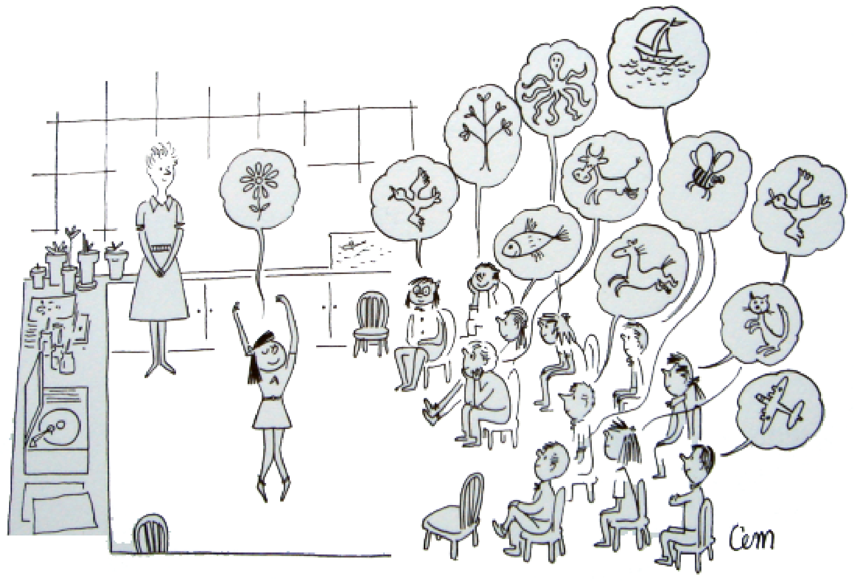


Figure 2.17 Visual communication may be misperceptible [11].

- *Representationary*: As we have just seen, it would be difficult to ideally represent all conventional data in their graphical forms as characteristic for visual abstractions. Thus, what we should base on to make the visual graphics is giving them just an amount of similar quantity for visual observation. The idea is mentioned in [11], graphical design is not "dependent on direct analogy to the physic world", thus "any variable quantity can be placed in relationship to any other variable quantity, measured for the same units of observation".

It is the main problem for the displaying of comparison data using quantitative visualization for the metadata in MP. But I will not come to that point in the thesis. What I want to cope with the topic is whether or not, and how we can make quantitative visualization to be simplest. The example of redesign of chart bar / histogram presented in [11] would be helpful, considering the above criteria, as shortly presented in figure 2.18.

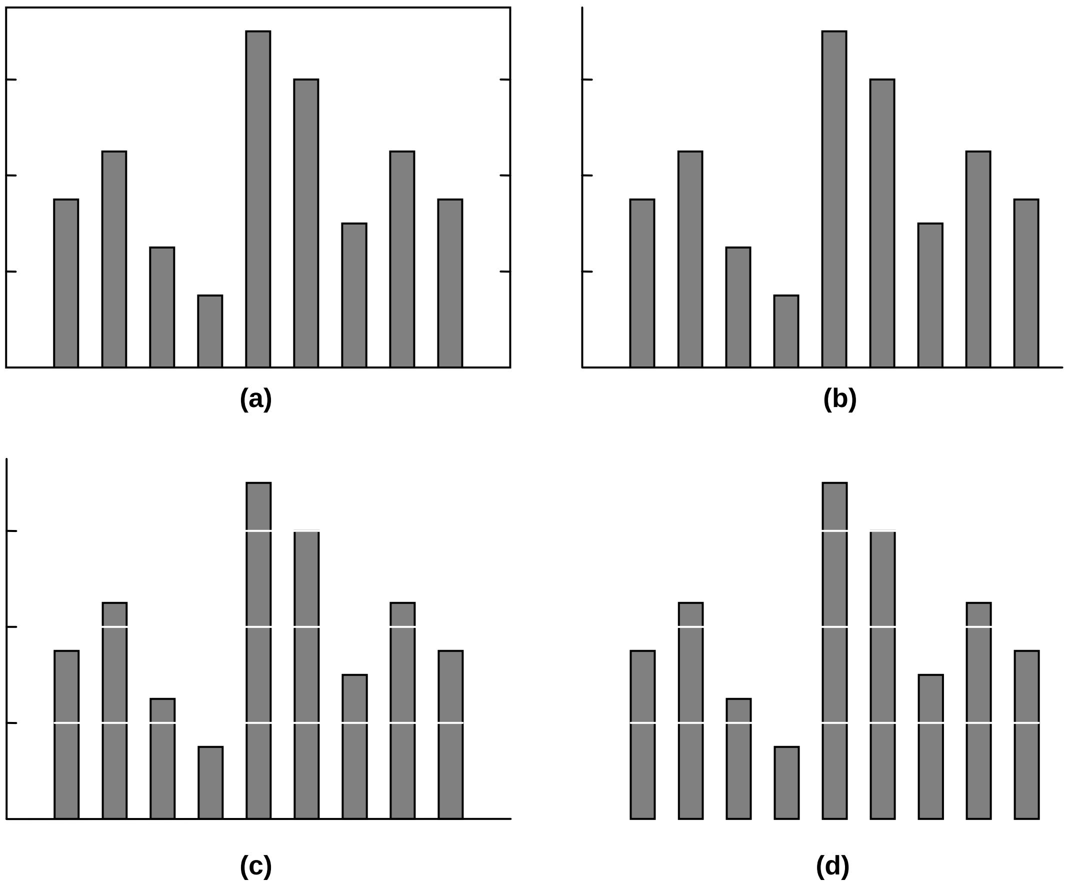


Figure 2.18 Redesign of the bar chart: (a) standard model bar chart normally found in statistical and scientific publications, (b) the box can be erased, (c) while grid can show the coordinate lines more precisely than ticks alone, and (d) the ticks are no longer no longer necessary for a nice and informative appearance [11].

2.3.5 Response Time and Display Rate for Interactive GUI

In subsection 2.3.2, tab panes are referred to as the widget that can be used to display different data sets on a limited visual area. It is the solution for the problem of exploring multiple data on the complex information space. In the case of the MP, depending on the user actions, alternative methods can better be applied. Animation is the method for presenting different abstract data which does not require extra user's actions, thus it is more efficient for the navigation task over the visual space. In this section, basic knowledge about response time and display rate as the fundamental for the discussion of the animation's implementation is mentioned, based on the theory presented in [3].

In his writing, Ben Shneiderman shows that cognitive and perceptual abilities of the users are the main issues that affect the interactive GUI that will be designed. A user can perceive the visual information which is slightly changed on computer system, and process it in different manners. Short-term memory, long-term memory and learning, problem solving, etc. are the classifications of such *human central processes*. A well-designed interface must cope with those problems for "user productivity, error rates, working style and satisfaction". For instance, the users do normally prefer the applications with shorter response times, which are 2 to 4 seconds for the common tasks, about which, the error rates reach the minimums, and the users' satisfaction is adequate.

Response Time Guidelines

- User prefer shorter response times.
- Longer response times (greater than 15 seconds) are disruptive.
- Users change usage profile with response time.
- Shorter response times lead to short user think times.
- A faster pace may increase productivity, but error rates also may increase.
- Error-recovery ease and time influence the optimal response time.
- Response time should be appropriate to the task:

  Typing, cursor motion, mouse selection: 50 to 150 milliseconds.

  Simple frequent tasks: less than 1 second.

  Common tasks: 2 to 4 seconds.

  Complex tasks: 8 to 12 seconds.
- Users should be advised of long delays.
- Modest variability in response time is acceptable.
- Unexpected delays may be disruptive.
- Empirical tests can help to set suitable response times.

Figure 2.19 Response time guidelines for interactive GUI design [3].

However, the preferred times are set up depending on the kinds of users, their experiences, and the specific tasks. For a given user and a task, there is a preferred response time. Long response times lead to wasted effort and more errors when a solution plan is reviewed continually. Shorter response times may generate a faster pace in which solution plans are prepared hastily and incompletely. Satisfaction generally increases as the response time decreases, but there may be a danger from stress induced by a rapid pace. For the simple data, a high display rate is acceptable, but it needs to be lower for the more complex data. In hard-copy terminals, typical rates vary from 10 to 160 characters per second (cps), but in the normal display terminals, the display rates best suit in between 15 and 30 cps.

Those ideas should be taken into consideration for the designing of icons and abstract texts for the information bars of the MP. However, the work just can be carried out with the integration of all the above mentioned subjects for the visual design and presentation of the MP, together with the user study of the demonstrated results. In the next chapters, we will come into details some proposal methods and discussions for them again.

Chapter 3

Concepts

After having taken a general view to the fundamental theories of *image distortion* and *visual interface design* in the previous chapter, we are now going on to discuss the concepts for the creation of the *Metadata Previewer*, as the basis for the implementation shown in the next chapter. The content of chapter 3 is structured as follows. In section 3.1, the process for illustration watermarks exploration is outlined, sketching the main tasks that will be mentioned in the later sections. Section 3.2 talks about the ways the watermarked objects are specified, according to its input data. Section 3.3 deals with the tactics for generating the distortion view of an image, based on the EPS model. Section 3.4 gives discussions to the problem of creating the toolbars for abstract metadata presenting. And finally, section 3.5 ends the chapter with the overall summarization.

3.1 Outlining the Tasks

Augmented images are images which have illustration watermarks[1] embedded in different places which are the segmentation objects inside. The goal of this thesis work is developing an *annotation browser* tool – see figure 2.3 – for metadata exploration. Loading an augmented image, the aim of a *viewer* is generating and displaying the embedded information for the watermarked object according to the request from the user, for instance when he moves the mouse cursor over the object in the image. In general, such interactive metadata exploration process is carried out considering these three main issues:

[1] See section 2.1.3 for the concepts of *illustration watermarking*.

1. The illustration watermarks are by default invisible to the user, thus the tool must be able to locate the watermarked objects at any time he wants.

2. Whenever a watermarked object is determined, as the user moves the cursor to and selects - for instance, the relevant watermarked region must be specifically recognizable on the screen.

3. The associated metadata are then shown in some predefined ways that the tool supports. New windows or GUI widgets can be the suggestions for the design of the "previewer" which represents the metadata content.

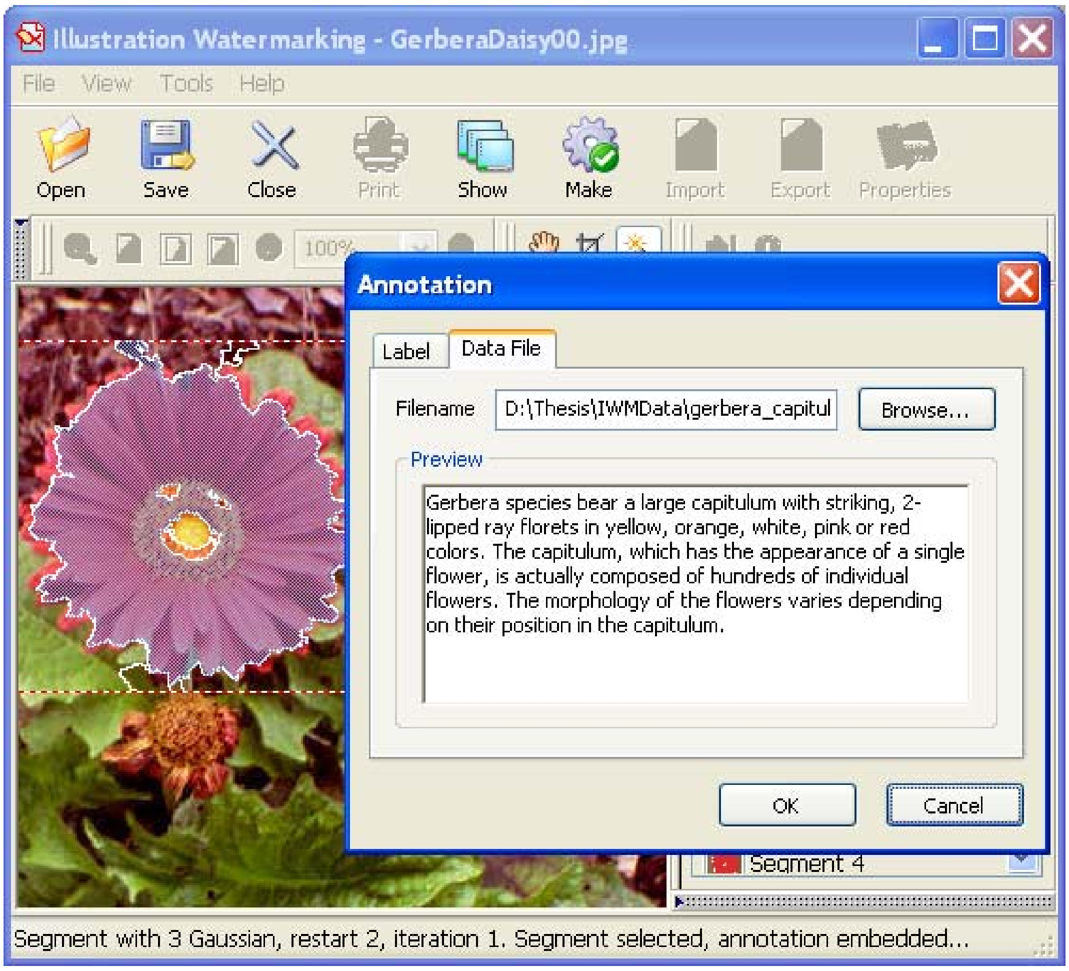


Illustration Watermarking - GerberaDaisy00.jpg
File View Tools Help
Open Save Close Print Show Make Import Export Properties
100%
Annotation
Label Data File
Filename
Browse...
Preview
OK Cancel
Segment 4

Filename: D:\Thesis\IWMData\gerbera_capitul

Gerbera species bear a large capitulum with striking, 2-lipped ray florets in yellow, orange, white, pink or red colors. The capitulum, which has the appearance of a single flower, is actually composed of hundreds of individual flowers. The morphology of the flowers varies depending on their position in the capitulum.

Segment with 3 Gaussian, restart 2, iteration 1. Segment selected, annotation embedded...

Figure 3.1 A simple way to show the selected watermarked object - by opaquely highlighting it, and the annotation data - in popup window. The screenshot is taken from the Illustration Watermarking Toolkit [10].

Basically, different techniques can be applied for the implementation of such metadata exploration process. For instance, the simplest way to indicate a selected watermarked object is to create an opaque mask for the object in the original image or highlight it using the object contour. And the embedded information is displayed on the application status bar, tooltips, or shown on popup or extra windows outside the displaying canvas. That simple approach can be illustrated through the example of the authoring tool IWM shown in figure 3.1. Though it is easy to implement, the method has some limitations considering the criteria of GUI design and interactive exploration for complex information space (see chapter 2 for more details). For example, as an exploration system, the tool must support the user ways to realize all watermarked objects in the image so that he or she can easily navigate through the image to explore them. In that case, a mapping window with markers indicating the objects would be a possible solution - see figure 3.2. But that kind of window is not really a complete recommendation since it occupies some space on the application screen just for showing the watermarks positions and nothing else. Another matter is the problem of displaying the embedded metadata. Whenever a watermarked object is explored, a new window with relevant data is popped up, which may cause interruption to the navigation process. Thus, the task seems also not adaptable to a friendly interaction application.

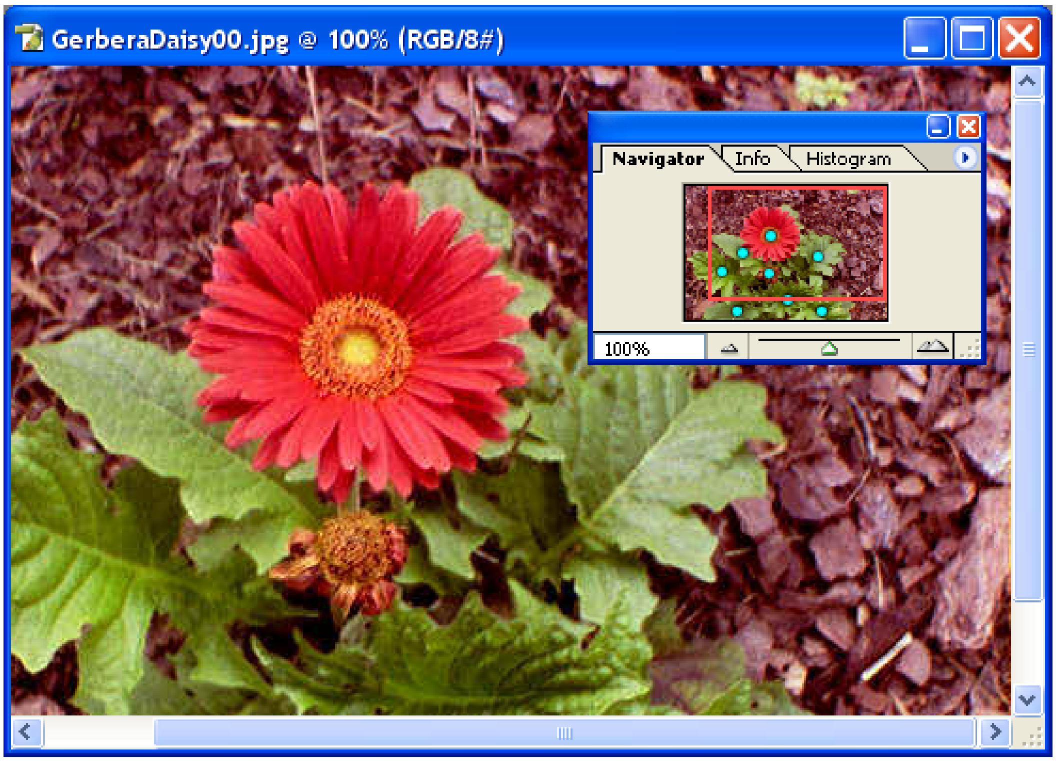


Figure 3.2 *Navigator* – the mapping window for image presentation and navigation in Adobe Photoshop®. The markers that I add to the preview image on the *Navigator* are just for the illustration purpose.

Unlike those methods, a distortion-based viewer that I establish in this master work would satisfy the requirements of an interactive metadata exploration application. The idea is that we can build a fisheye cursor that dynamically navigate over the watermarked image and automatically update its appearance - *the previewer* - on the application canvas without changing the original image. Whenever a mouse event occurs, the previewer gets the cursor position, determines the relevant object pixels and the embedded metadata of the watermarked object, then generates the distortion view and the toolbars for metadata presentation and display them on the application screen. The process for such metadata previewer creation is illustrated in figure 3.3, with the three main steps reflecting the three main issues mentioned above. Here, (a) *inserting the object markers* and (b) *highlighting the focus object* are implemented for the tasks of locating the watermarked objects; (c) *enlarging the focus region* is done for the determination of the selected watermarked region; and (d) *adding the information about the embedded metadata* are accomplished in forms of toolbars for abstract visualization to show the explored data.

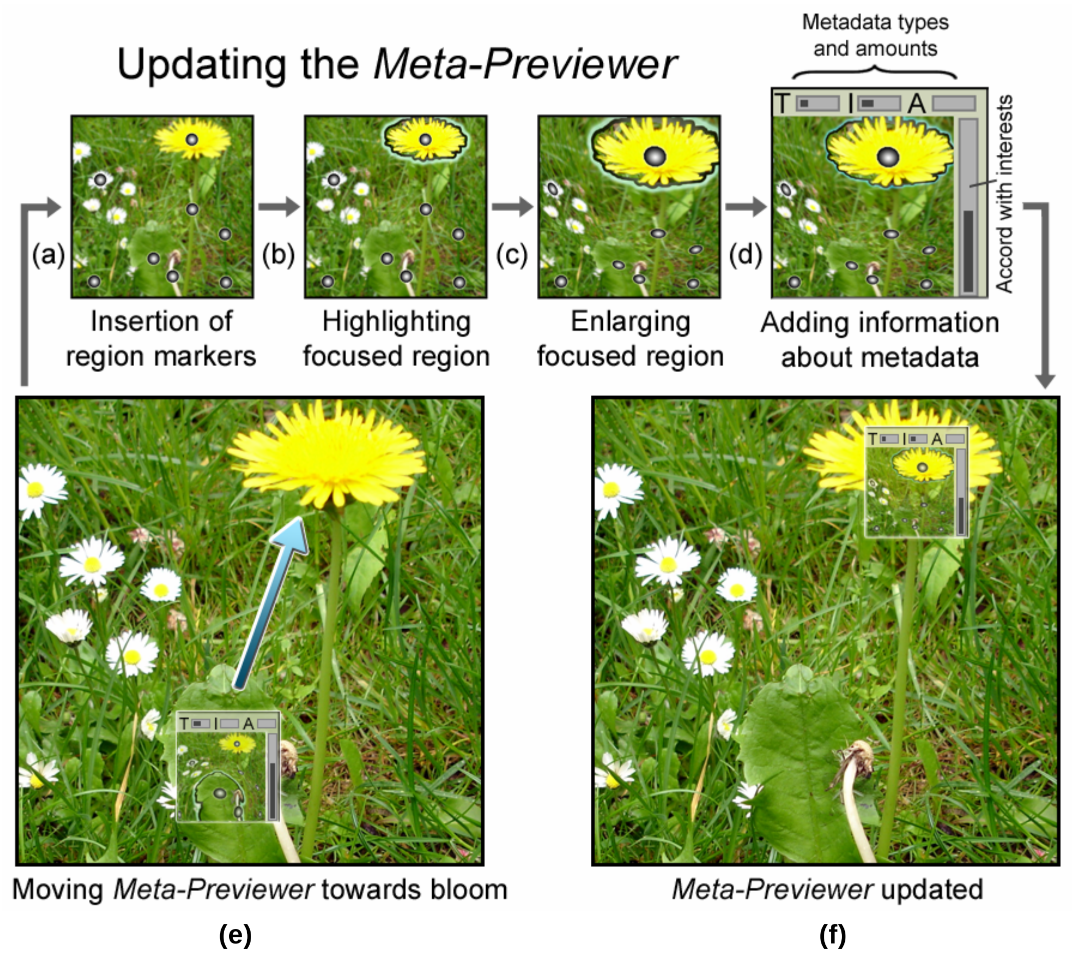

Updating the *Meta-Previewer*
Metadata types and amounts
T I A
Accord with interests
(a)
(b)
(c)
(d)
Insertion of region markers
Highlighting focused region
Enlarging focused region
Adding information about metadata
Moving *Meta-Previewer* towards bloom
Meta-Previewer updated
(e)
(f)

Figure 3.3 Process for updating the *Metadata Previewer*. Courtesy of Henry Sonnet.

All steps for this visual interface implementation are done on the previewer itself, and we should take care of this knowledge during the presentation of the thesis work. In detail, the object markers are included in the previewer in form of small dots, and the focus object is highlighted with the contrast colors contour. They are designed to attract the viewer attention. The more discussions for such *object displaying* tasks will be taken place in the next section. For the focus object enlargement, distortion techniques using EPS model of Sheelagh Carpendale are applied, presenting in section 3.3. Fisheye lenses are used to emphasize the watermarked object while the whole context is remained for the further navigating action. The associated metadata embedded in the image segmentations can be in any format of text, image, audio, video, or other data types, which semantically annotate the image objects content. For instance, the illustration watermarking for a world map image are supposed to contain information about population, short text about history, images for flag, coat of arms, detail map, and other statistics in the regions indicating the countries. The embedded metadata are, therefore, quite huge and complex. Thus, loading all such *raw* metadata and represent them in a small space of a previewer cursor seem not to be a workable solution. To tackle it, abstract information for metadata presentation is suggested. Those data are represented in form of toolbars, namely the *information bars* – as shown in figure 3.3d. By the way, it is the matter of section 3.4, and we will now learn about the tasks of creating and highlighting the objects.

3.2 Displaying Objects: Markers and Highlighting

As mentioned, the first issue needed to be fulfilled in the Metadata Previewer is the visual interface that supports the user ways to identify the watermark positions and easily locate them through the navigating action. Since the previewer is just a small GUI cursor moving on the explored image, the visual items designed inside must be tiny and informative. Two main points needed to be solved in this section for watermarked objects displaying are *creating the objects markers* and *highlighting the focus object*. They are included to the previewer with the aim of attracting the user's attention.

The markers used for objects identification are suggested to be generated and displayed on the previewer in the form of small dots. Different methods can be applied for such objects positions determination, and what we need to accomplish is deciding the most suitable technique that can indicate well the objects. Choosing the position of marker does also affect the matter of creating the distortion lenses mentioned in the later section. For highlighting the focus object, object contour is used to disconnect the focus region from the whole image, and it is highlighted by putting the colors into contrast.

By the way, the first topic needed to be presented here is the concept of the illustration watermarks objects. To understand the techniques for objects displaying, we first should get the general idea about the (data) structure of the watermark area as the basis for the configuration of the positions for object's markers, as well as the contour of the object presented in the next subsections.

3.2.1 Object Description

The illustration watermarked object is defined as a unique segmentation inside the watermarked image. Every object is structured with a mask buffer indicating the watermarked region, and some associated metadata files. In detail, an illustration watermarks image contains at least these data:

The original image: This is the image that will be used for the creation of the distorted view – see section 3.3. It is also the image that is displayed on the client area of the application. The image can be seen as a copy of the cover image in which the watermarks are supposed to be invisible to the user.

The object IDs buffer: This is the mask image corresponding to the watermarked pixels of the objects in the original image. They are unknown to the end-user, but will be used for the determination of the explored object when the previewer moves over the original image. In some applications, for instance the IWM [10], different mask image is generated for each object, or they can be merged together to make a *full* image. In the detail implementation, we will learn about the detail structure of this buffer which I use for the Metadata Previewer, as shown in the next chapter.

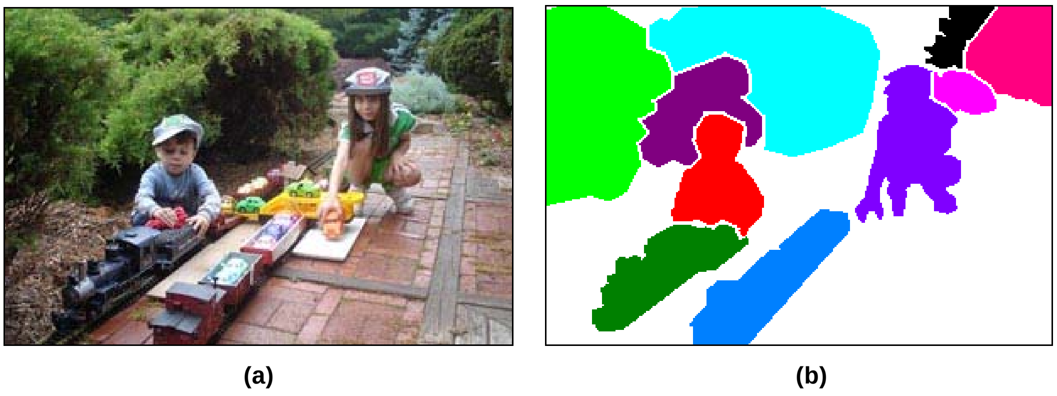


Figure 3.4 An illustration watermarking image (a) and its object IDs buffer (b).

The associated metadata: Associated with each ID in the object IDs buffer is the embedded metadata, which can be in any data format: a short text, a binary code, or even a large video file. They may be completely embedded into the object area, or just partly embedded as hyperlinks. The approach for embedding (and then extracting) the metadata depends on the capacity of the watermarked region itself. Since such data can be complex, they are expected to be partly shown on the previewer with just the abstract information, as presented in section 3.4.

Complex data structures and file formats can also be used to describe the watermarked objects and their metadata. For instance, instead of using the object IDs buffer with pixels indication, vector objects in SVG (Scalable Vector Graphics) format, or geometric models with triangles can be some other solutions for the mask objects. Different data structures will lead to different approaches for creating the visual forms of the previewer, and a well-constructed structure will help the programmer easier in implementing the application. For a well-designed system, the data structure must be investigated in depth, since with it, the tasks for selecting, manipulating, and representing, or storing objects and metadata are much more flexible and standardized.

3.2.2 Markers For Objects Determination

The markers are used to drive the viewer's attention to the watermarked objects in the original image. A marker that reflects well the object should adapt these two requirements for visualization:

i. *The marker must be at the center of the watermarked region*: It is the best position to get the focus from the viewer. Normally, when we look at an object without paying attention to any special point, we need to care about the whole object. Therefore, the center point is the convergence of our focus. Beside, in the case of the metadata previewer, the watermarked region will be scaled into any size. Thus, placing the marker at the center point will not affect the scale displacement.

   This center position is also used for the creation of the fisheye view, known as the focal point of the lens, as mentioned in section 2.2.3 and 3.3. Adjusting the height and the lens radius from this center point, we will have different views of the distortion region.

ii. *The marker must lie inside the object area*: The easiest way to determine the center point for a region is considering it as the center of the boundary rectangle that covers the region. However, a watermarked region may have arbitrary shape, thus such center point may lie outside the object area. In this case, it will not reflect well the object, and can not be a good solution. In figure 3.5 we see that the center of the boundary rectangle of the object in (a) lies inside the region, while the object in (b) is not, which comes to the solution with the result in (c).

The formula for computing the center of the boundary rectangle of a region is:

$$x_{C1} = \frac{x_{\min} + x_{\max}}{2}, \text{ where } x_{\min} = \min_{i=1}^{N} x_i \text{ and } x_{\max} = \max_{i=1}^{N} x_i, \tag{3.1}$$

with i={1,…,N} and N is the number of pixels in the object region. The same formula is also applied for the coordinate y_{C1}.

The better solution is computing the average moment of the object pixels. Taking a closer look, the expected center (x_{C1}, y_{C1}) in (3.1) is the average value of all pixels - both objects and background one - in the cover rectangle. Dropping all unnecessary pixels that do not belong to the object, we come to the average moment:

$$x_{C2} = \frac{1}{N}\sum_{i=1}^{N} x_i, \text{ and } y_{C2} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{3.2}$$

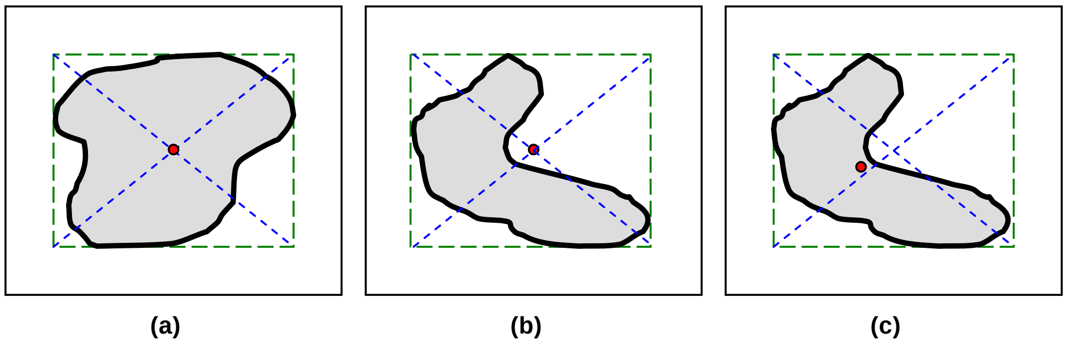


Figure 3.5 Marker positions for the segment in image. In (a) the center point of the boundary rectangle lies inside the object, while not in (b). To tackle that problem, the average moment can be used to compute the new point, as shown in (c).

The two formulae above are simply implemented in my Metadata Previewer, though they do not completely satisfy the second requirement. For demonstrating, the simplest way can also be applied is using an authorizing tool, for instance the IWM, to specify the markers. A clear example for the data describing those hand-made points can be found in the XML structure mentioned in the next chapter.

Advanced than them, another technique for automatically computing the markers, based on the technique for object anchor generation for interactive labeling of complex 3D model [34], is suggested.

The coordinates of the marker computing for an object following the formula 3.2 may still lie outside the object region. The reason for that is each coordinate *x* or *y* is just the average of all the coordinate-points, thus the marker may not coincide with any point of the object. With that matter, I come to the idea of selecting the point that always lies inside the object by testing each of them. The point that has the minimum *distance* to the other points is chosen for the object marker, which can be computed through this formula:

$$C_3 = P_i \mid \min\left(\sum_{j=1}^{N} d(P_i, P_j)\right) \tag{3.3}$$

where i={1,…,*N*} and *N* is the number of pixels in the object region. The distance from the point P_i to P_j can be computed with the Euclidean distance.

However, comparing the minimum distance for all the points in a large watermarked region would be a slow task, and the result point may not lie “in the middle” of the region. The skeleton, which remains the geometric form of the region, is the better option for such computing. It keeps the idea of an inside point, and also laying in the *middle* of the region. The overall algorithm for creating a marker is as followings:

1. Generate the skeleton of the watermarked region, for instance using the Hilditch's algorithm, resulting with a set of points S_i, *i*={1,…,*M*}, where *M* is the number of skeleton points.

2. Compute and select the marker among the skeleton points which has the minimum distance to the rest points, using formula 3.4 with the similar parameters to formula 3.3 for the set of skeleton points S_i.

$$C_4 = S_i \mid \min\left(\sum_{j=1}^{M} d(S_i, S_j)\right) \tag{3.4}$$

The concepts can be illustrated in figure 3.6 with the marker in (c) is the most exact position among the mentioned techniques. It is therefore chosen for my MP. However, the other techniques are also implemented for demonstration. Furthermore, beside the concepts of specifying the position, the matters for size and color of the marker are also in consideration for the designing. Since the previewer is a small cursor moving on the image, the marker must be designed as small as possible. As the result of my testing, the size of 4 to 8 pixels would be the best solution, since it is not so small to be unrecognizable, but not so big to occupy so much space on the previewer. For the marker color, two contrast colors are chosen to draw it, with the idea is that they can highlight the marker, as discussed in the next subsection of watermarked object highlighting.

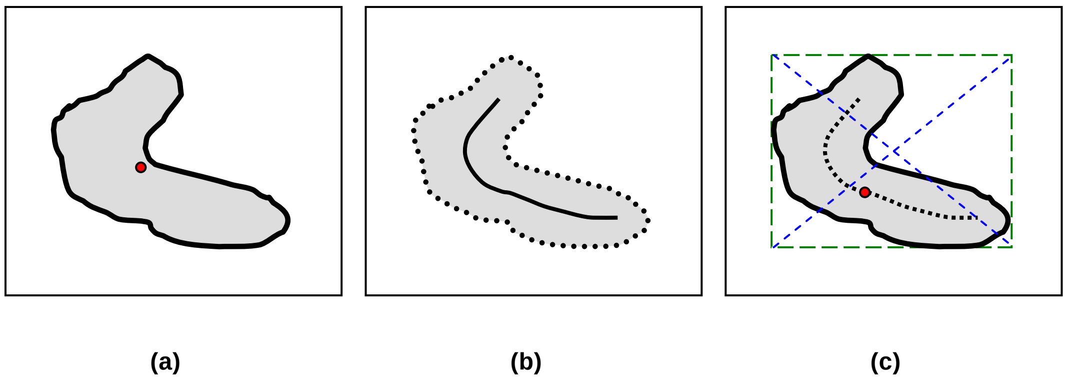


Figure 3.6 Marker position using the skeleton - minimum distance technique. In (b), the skeleton of object in (a) - also figure 3.5c - is generated, and the selected marker is shown in (c) as the one that has minimum distance to the other skeleton points.

3.2.3 Highlighting the Focus Object

The markers can help the user to determine the watermarks' places in the image, but they do not let him know the exact areas that were watermarked, which are also important for the visual exploring of illustration watermarks images. In the next section, we will learn about the ways that distortion techniques can be used to focus a selected object region in an image. But that is not enough. In the case of not well-seen or undetermined object's region, the user may still not identify the object just with the fisheye view. Therefore, highlighting the object region for a clearer appearance is a necessary task. The theory of perception indicates that we can determine the visual objects through the physical reflection of their colors and shapes to our retinas and brains, as well as the psychological dispute of our minds. The more differentiating the objects are, the easier we can recognize them. Since the visual objects in our world are so complex in colors, shapes, and sometime mixed together, the mot suitable ways for us to separate an object from the whole image, without caring about their visual properties, are making a contrast line between it and the other objects, changing its colors or pattern from the normal view, or changing those of the rest image. As illustrated in figure 3.7, I would suggest using contrast object contour, object opaque mask, or lightening background image to differentiate a selected object from the whole image.

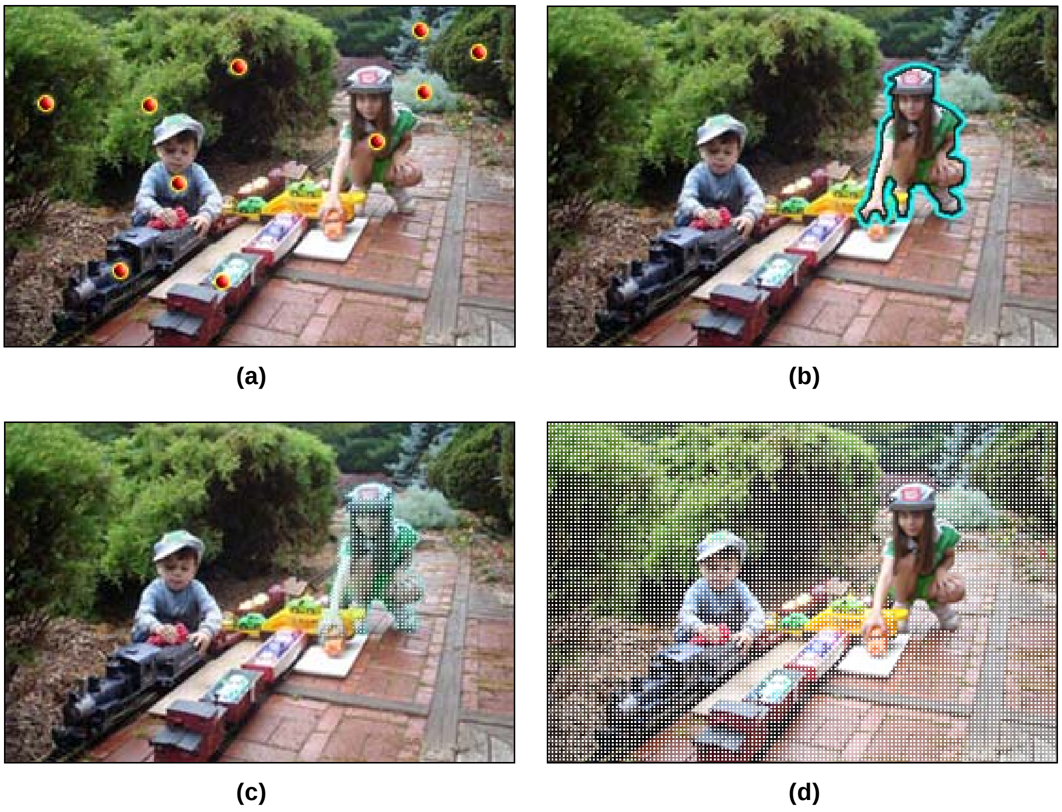


Figure 3.7 Object markers (a) and one high-light object using the contrast contour (b), mask grid for the object (c), and mask grid for the rest area (d) of the watermarked image shown in figure 3.4.

In any case, the algorithm for such manipulations on the image is passing each pixel or pattern in the image and changing their data values. For instance, suppose that we have a matrix M_{ij} as the mask image of object IDs buffer, *ID* is the ID value of the selected object, and I_{ij} is the matrix of pixels in image, the algorithm for opaque mask creation over the object is as simple as followings:

1. Go through the mask matrix *M* of the selected object with *i* and *j* reflect the *x* and *y* coordinates (basically, *i* and *j* just run from the left to the right and from the top to the bottom of the object cover rectangle),

2. Get the mask value M_{ij} for each pixel which is an object ID value,

3. Update a new value for the pixel according to its ID value. If M_{ij} is equal to *ID* and its surrounding pixels are not the grid color value, change I_{ij} to the grid color value; where grid color value is a predefined color that "opaquely" masks over the region. I call it the *grid color value* since the mask area is seen as a grid of that color – see figure 3.7c for more concepts.

By creating the mask grid, we still can see the selected object but discern it from the whole image. However, in both cases of specifying the object – figure 3.7c, and lightening the rest area – figure 3.7d, the techniques are still not so adequate, in comparison to the method of using the object contour, as shown in figure 3.7b. The reason is that using an opaque mask makes much change to the image content than the way we use the highlighted contour. For detail implementation, the matter of creating the contour for object can be worked out by a number of techniques. From a seed inside the object region, algorithm for seeded region growing, or chain codes with size growing can be used to make the line. With the idea of region growing, I suggest to use a simple algorithm that work with the object IDs buffer to create the contour, as mentioned in the implementation chapter, section 4.2.2.

3.3 Generating the Distortion View

Distortion techniques are used to relatively display the visual information in a limited information space by emphasizing the most interesting[2] area, which is the focus region, and deemphasizing the less important surroundings, which are the viewing context. Different methods about such lenses creation with their own purposes and applications have been introduced in the previous chapter. Among them, the Elastic Presentation Space (EPS) proposed by Sheelagh Carpendale [24] was chosen for the implementation of the Metadata Previewer. The reasons for that are the EPS model is a general framework that supports different ways to manipulate many types of fisheye views, which are necessarily implemented for my application, and also easy to apply for the raster images. The basic concepts for EPS model was presented in the §2.2.3, and in this chapter I will give discussions to the employment of the model into the Metadata Previewer.

The detail applying of 2D-to-2D transformation for an image using the equation 2.5, which results in a new image that has the "appearance" of a distortion view, is not transparently used in my implementation. In its *raw* computation, the result image is not a well-seen distortion. The problem is illustrated in figure 3.8. The more adjacent the pixels are to the focal point, the less density for their visual data to be shown on the image. That leads to my proposal for generating a fine, well-seeing image following these main steps:

1. *Data preparation:* Determine the expected size for the distortion image - with small dimensions as the requirement of the previewer cursor - and scale the original image to a suitable size based on it,

[2] See chapter 2, section 2.2.2 for the definition of DOI, the *degree of interest*.

2. *Distortion Computing:* Generate the matrix of raster pixels for that scaled image using the given EPS formulae as mentioned in section 2.2.2,

3. *Updating the view:* “Resize” the resulting matrix of pixels to an expected size to get the distortion view in the previewer, depending on the resulted pixels’ density (the number of pixel in a predefined region).

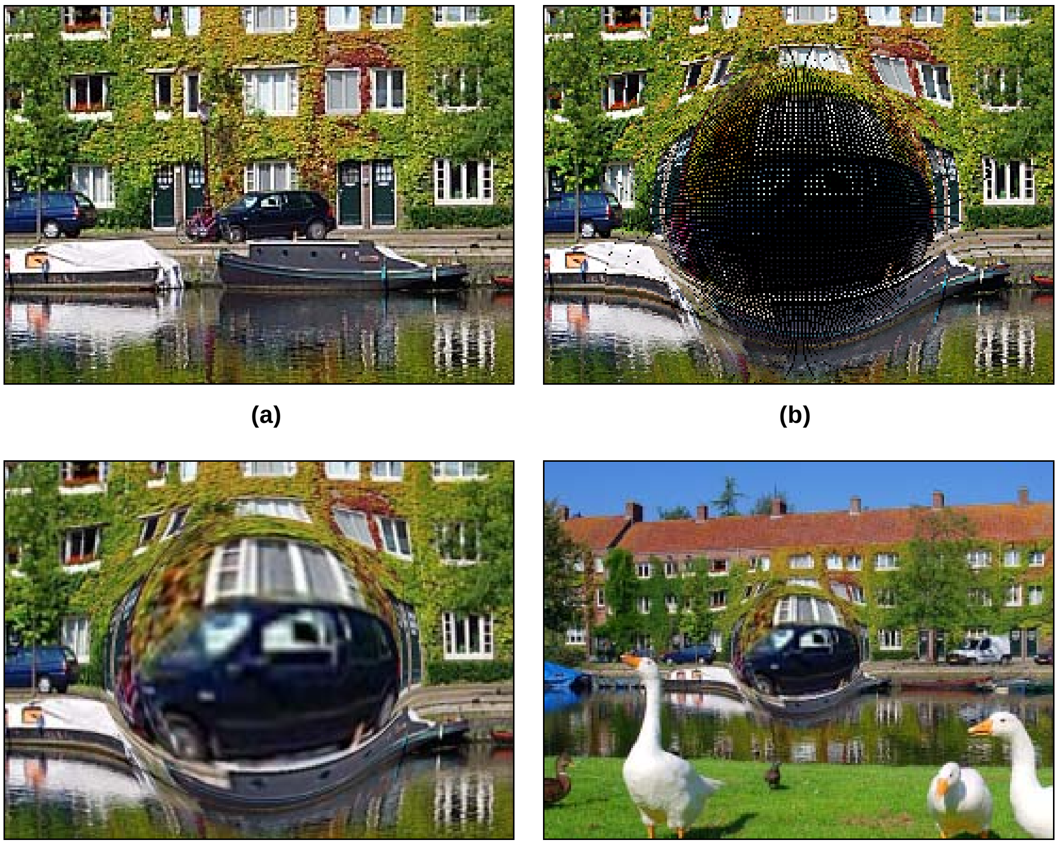

(a) (b) (c) (d)

Figure 3.8 Solving the problems of *raw* distortion: (a) a part of an undistorted image; (b) the raw distortion with gaps between pixels; and its adjustments (c) with the same image size, and (d) smaller size.

In the raw distortion image, there are gaps (un-filled spaces) between the pixels next to the focal point and that focal point, since the magnifying transformation moves every pixel a distance $(x_m, y_m) - (x_i, y_i)$ from it original position. Thus, we need to “remove” those gaps to get the well-seeing distortion view. To discard the gap inside a tetragon created by a four-adjacent-pixels transforming, we can scale the tetragonal region to a bigger size that covers the gap. Anyway, the functions for scaling an image region supported by programming libraries do normally work just for rectangles, not for arbitrary tetragons. To tackle the matter, one can build a scale for arbitrary shape, but it is still impractical, since the edges between the resulting scaled regions may also be not smooth, and they require further adjustments, i.e. using the anti-aliasing techniques for pixels color, to eliminate the noise.

The idea of updating the view implemented in my application gives a better solution for the problem, and it is quite simple. Based on the pixels density, i.e. the number of pixels inside a region, I suggest mapping a group of pixels in the raw distortion into one new pixel in the final image. There, the relevant pixel color is the average of all pixels in the raw region, and it remains the dominant color for that region. The result image becomes a "scaled" version of the *raw* distorted image, but in a smaller size. Also, with that method, in order to get the expected size for the previewer, we should first resize the image which is used for the distortion computation, as mentioned in step 1 and 2. The method works quite well and fast since we need just a small previewer for metadata exploration.

The magnification level and lens shape are controlled through some parameters. In EPS, magnification level depends on the height h_f in comparison to d_b. By increasing h_f, the view plane moves nearer to the RVP, and therefore raises the magnification. Figure 3.9 gives illustration to that concept. Following the Equation 2.5, the magnification factor *mag* can be defined as in equation 3.5. With this factor, we can easily get the density of raw distorted pixels to get the scale ratio for data preparation defined in step 1.

$$mag = \frac{d_b}{d_b - h_f} \tag{3.5}$$

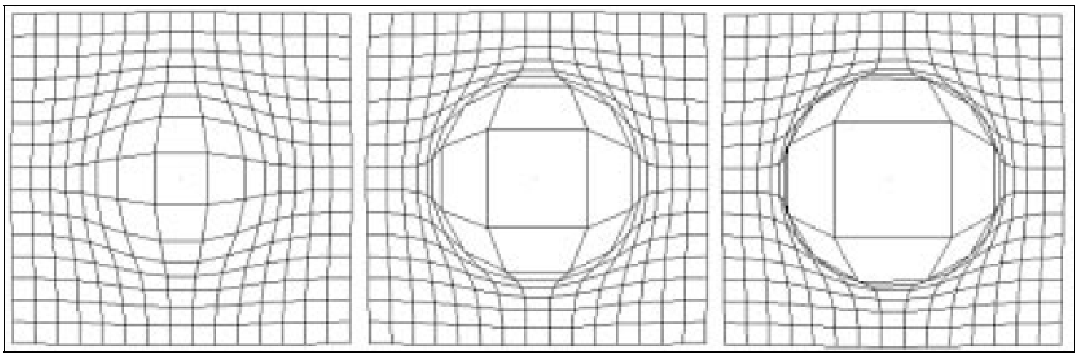


Figure 3.9 Effects for distortion view varying the height h_f [24].

The drop-off functions and the metrics used to compute the distance from a pixel to its focal point do also affect the shape of the distortion view. In section 2.2.2, we learnt that the Gaussian for drop-off function where the standard deviation σ is a predefined parameter produces best shape for a lens, thus it is chosen for my implementation. The distance d_p is another parameter, with Euclidean (L-2) is the metric used, since it produces the best sphere shape, in comparison to other metrics L-n (n≠2) [24, §4.2]. The size of the distortion view is also a matter for considering. In my implementation, I choose predefined constants for that, based on the sizes of the previewer and the watermarked region. Supposed that *rad* is the radius representative for such size, the computation of d_p for pixel (x_i, y_i) is formularized in equation 3.6.

$$d_p = \frac{L_2}{rad} = \frac{\sqrt{(x_i)^2 + (y_i)^2}}{rad} \tag{3.6}$$

For detail implementation, one can support relevant parameters in his tool so that the user can adjust the distortion view. In my Metadata Previewer, some of the mentioned options are given. The demonstrating results with predefined values for distortion parameters and the discussions for that will be deeper mentioned in the next chapter.

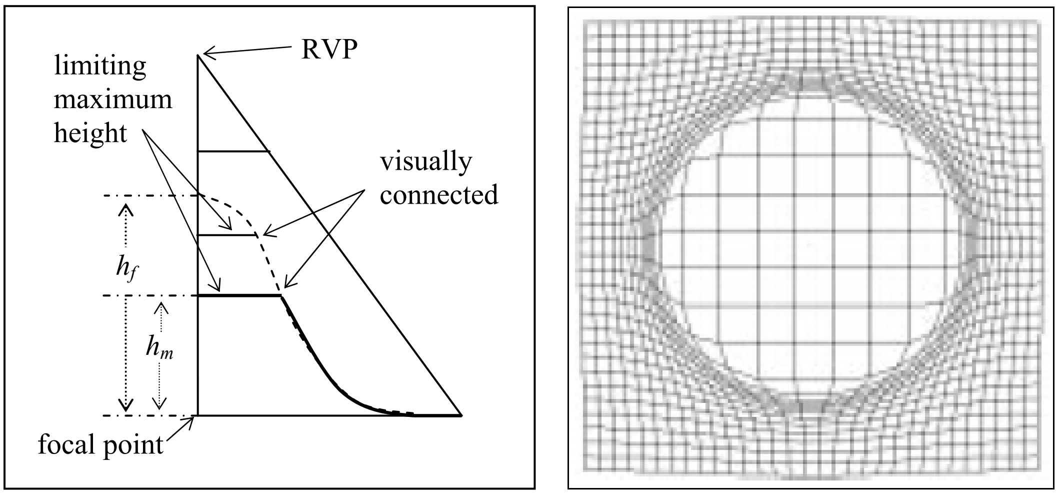


Figure 3.10 Diagram (a) and distortion view (b) for scaled-only focus region [24].

The final work mentioned in this section about distortion view creation is a matter for displaying the watermarked object. As we have seen, the lens that we got from the above computation is a sphere with the most magnification area at the focal point, and the context is much more collapsed as it is further from the focus. Thus, it does not give a really good view for the object in the case that the object is determined by a pre-marked contour as discussed in §3.2.3, since the contour may disappear if it is quite far from the central point. To avoid the limitation, a scaled-only focus region is used to flatten that view area [24, §3.5.1]. For specific tactics, the center of the 3D Gaussian curve can be projected up to the height h_f, which may be higher than h_m, and we can set the height h_m for all shifting pixels inside the scaled-only area to get the expected result, as presented in figure 3.10. This technique is also used, together with the normal distortion, in my Metadata Previewer tool.

3.4 Presenting the Abstract Metadata

As mentioned in the above introductory section, the previewer cursor consists of the distortion view for the watermarks objects determination, and the *information bars* for the associated metadata presentation – see figure 3.3d. The bars are defined as the *information bars* since they are suggested to be designed as the small *toolbars* – for saving visual spaces, and contain *information* of the metadata embedded in the exploring watermark object as well as their searching feedback. Actually, the embedded metadata can be extracted and displayed in different visual forms, as shortly discussed in section 3.1, in their original formats or in abstract forms, depending on the specific requirements and characteristics of the applications for complex information space exploration. The requirements for the visual interface design of an interactive navigating widget like the Metadata Previewer are, according to my studying, *well-looked*, *informative*, *and easy-to-use*, as fundamentally presented in section 2.3. In this *concepts* chapter, I would like to continue with the detail discussions concerning these main issues:

i. *What should be included in the information bars?* A useful previewer is the one that can show the most essential data. But that saying seems vague, since there would be no general answer for what the essential data really are. The data are normally expected to be retrieved according to the request of the users. In the case of the Metadata Previewer, some common properties of the associated metadata files are suggested for the *abstract metadata bar* and a chart bar for the expected searching information is proposed, namely as the *intersection bar*. The detail explanation about those bars will be presented later.

ii. *How the bars should look like?* This is the matter of visual decoration. To show the visual information in a small area of the previewer that dynamically navigate over the displaying canvas, the abstract objects like *icons* and *small texts* are considered, as discussed in the previous chapter. The questions come now are in which size and template, and at which position on the previewer are the possible and the most suitable selections for the design of the visual items? They are also the complex tasks to be discussed.

The toolbars for metadata presentation, as mentioned in (i), will contain the *abstract metadata bar*, and the *intersection bar*. **The abstract metadata bar**, as its name, is supposed to contain the information about the embedded metadata of the exploring watermark object. In the previous discussions for the concepts about the illustration watermarks images, we have learned that the embedded information does reflect the logical meaning of the objects they annotate, which are in any data formats. Thus, a well-designed metadata previewer is supposed to be able to extract and show the relevant information - for instance the detail map of a selected country, its image for coat of arms, the text information about the history, and other watermarked information in the metadata bar - embedded in the region indicating the cities, countries. The conventional way for us to do that is creating the appropriate displaying windows which require a large visualizing space. But they are not suitable to the case of the Metadata Previewer, since the previewer proposed in my work is supposed to be a small window that dynamically navigates over the watermarked image; and in addition to that, most of the space of the previewer is reserved for the presentation of the distortion view.

Consequently, here comes my solution. The “abstract” information specifying the metadata and the specific visual objects of *icons* and *short texts* are the keys for the visual design. The concepts about those types of visual objects for references can be found in the previous chapter, section 2.3.3. Since all data can be stored in files, we can get some abstract information from their file properties. Loading a metadata file, it would be necessary and feasible to show these kinds of information:

- **Data size:** This is the amount of data that was watermarked into the illustration object. It indicates the data capacity of the watermarked object. Depending on capacity, the data can be completely embedded or just its directive information with hyperlink to external filename is added into the image area.

- **File name:** In both cases of directly embedded or external hyperlinked, I suppose that the metadata are always saved into file. For that, filename with predefined icon is suggested to be included in the bar. In my implementation, icons for text document, image, audio, video file, and another one for undetermined file type are used.

- **Abstraction data:** This data is a conceptual one, and it would be difficult to idealize them as the most informative and extractable data. The predefined information logically depends on the application requirements. For a text document, we should look for some central words. For an image, we may show the histogram, number of colors, or image dimensions. For an audio or video file, we can think about the time length, or its streaming bit rate. In any case, the information data are extracted just following some supported tools.

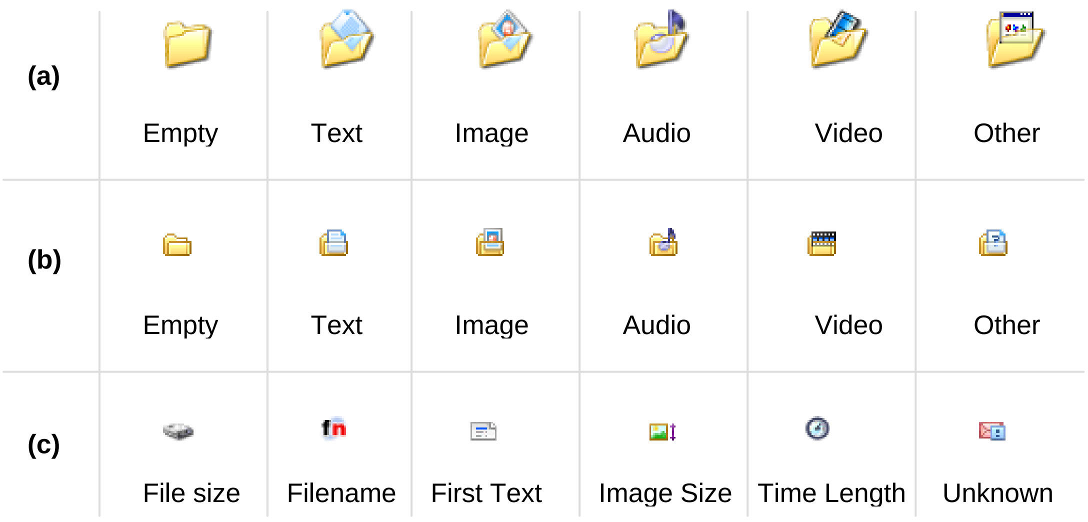


Figure 3.11 Icons used in the abstract metadata bar of the MP: (a) big and (b) small icons for file types, as of MS Windows XP®; and (c) some data properties.

The metadata files, as mentioned, would be *text*, *image*, *audio*, *video*, or *undetermined* formats, and the data properties, as I have just listed, are *file size*, *filename*, and *first text*, *image size*, *time length*, or *unknown* property for abstract metadata. Since they are to be presented on the metadata bar for abstract information, a set of predefined template for them are suggested. In figure 3.11, a list of icons used is presented. Icons are the most suitable visual objects representing those abstract items, because they can symbolically express the items and also occupy the least visual space. Getting the watermarked data, the icons together with short texts are presented. Texts are chosen for the contents of the visual items since they are easy to be extracted and describable for the metadata.

The question is how we should place the visual icons and the texts in the metadata bar. In section 2.3.2, pull-down menu, toolbars with just icons, piemenus and tab views are some methods can be used for multiple visual items presentation.

Based on those models, I suggest creating some templates as presented in figure 3.12. A list of abstract metadata items with icons and texts like a normal menu can be placed above or under the distorted view (a); around the lens (b) like a piemenu; or just a tab views with representative icons as shown in (c). For visual perception and data information criteria, each template has its own advances and limitations. For instance, with the menu-based bar (a), we need a large space for the multiple items presenting, which is not so relevant in my MP since the previewer is expected to be as small as possible. The advantage of this presentation model is that all the necessary abstract data are displayed. The similar way, but with a nicer "boder" – though it is not easy to watch – is the design of items around distortion lens like a piemenu (b). Here, text with smallest font size (which are perceivable to the users) and icons in smallest size are applied. In my implementation, `Helvetica` or `SansSerif` fonts with size=7 `pointSizes` (in Trolltech Qt®), and icons with height less than 16pt – figure 3.11c – are the best matches.

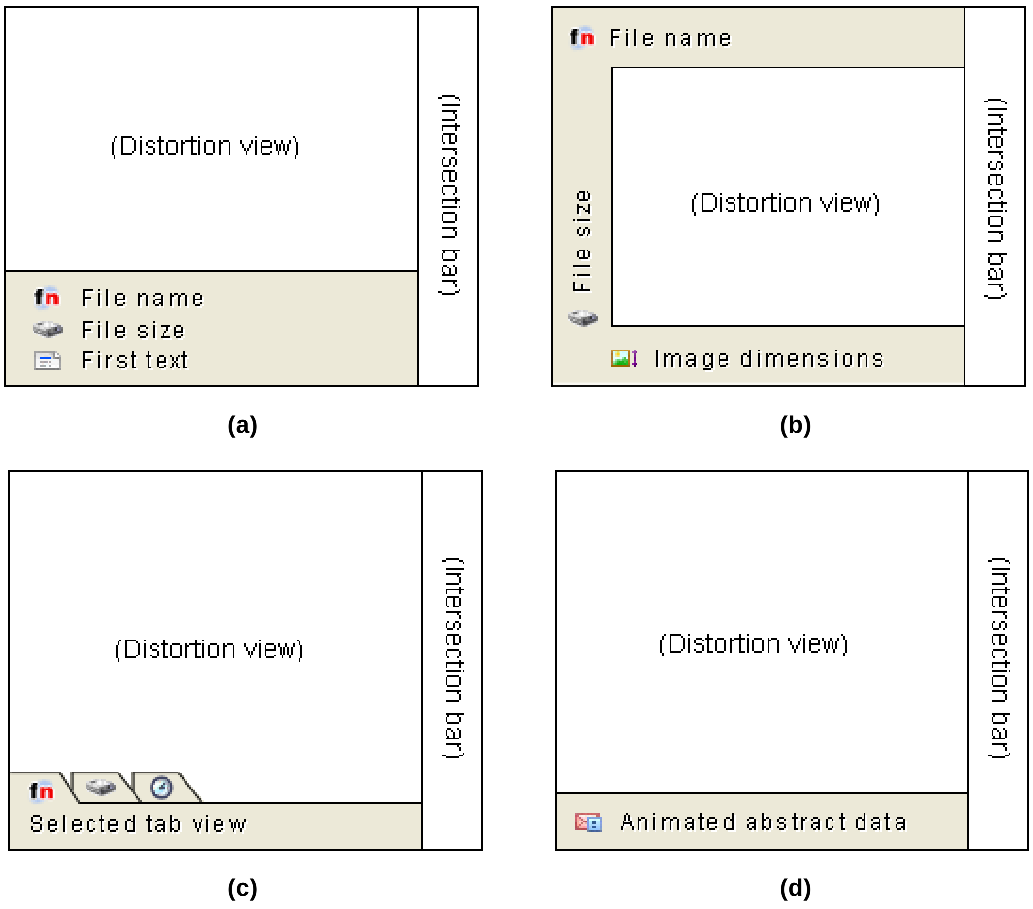


Figure 3.12 Abstract visual items placements - (a) like a normal pull-down menu, (b) in form of piemenu, or (c) tab views. The illustration for in-turn animation is presented in (d).

With the concepts of display rate - see section 2.3.5, we can save the visual space for metadata presentation by alternately displaying the abstract items in the toolbar. They are shown like animation, with the display rate of a few seconds per each "frame" containing the icon and short text, as illustrated in figure 3.12d. This last method adapts all requirements of *well-looked*, *informative*, *and easy-to-use* for GUI design, and therefore is implemented in my MP. In the next chapter, we will see the demonstration results of both normal and animated metadata bars in figure 4.9.

The second information bar that represents the data searching feedback is **the intersection bar**. For the purpose of data exploration, not only the metadata properties are displayed, but also their *meanings* would be in consideration. The metadata previewer is expected to show the likeliness of a looking-for metadata and the exploring one. Giving some expected information, for instance some text keywords, file size, or image color histogram, whenever the previewer cursor is moved over the watermarked object, the system compares the two, and the "intersection" between them is displayed in the form of visualizing chart bar. Here, chart bar is referred to as a useful kind of quantitative visualization – see section 2.3.4. The task for displaying such data is quite simple; with column bar showing the percentage of similarity is suggested. However, the computation of data similarity is a complex task, since we need to analyze and set up specific metrics for them. That work is out of scope of the thesis work, thus it would be suggested for future directions.

The simplest way is creating a solid chart column as presented in figure 3.13a, with the gray box indicates the maximum data likelihood (100%), and the filled-in black region is the amount of intersection. Since the visual space is so tiny, it is irrelevant to add some more information, e.g. the numerical label of 25%, 50%, etc., next to the bar. The better way for showing such reference can be adding stick lines into the bar, as shown in figure 3.13b, or a number indicating the intersection level inside the bar, as shown in figure 3.13c. Colorizing the bar is also a necessary task for a well-looked GUI application, as illustrated in figure 3.13d.

The methods for creating and displaying the information bars that we have just discussed may give a good solution to the problem of metadata exploration. However, it is not the only approach. There would be other ways that can also well perform the task, especially in the case of multiple metadata exploration – see chapter 5. For some applications, the requirement for keeping the original presentation of the image is not necessary, thus we can highlight it on application canvas. In that case, the previewer can be completely used for the metadata presentation. Labeling techniques or cascading previewers can be employed to tackle such kind of multiple level illustrations watermarking system.

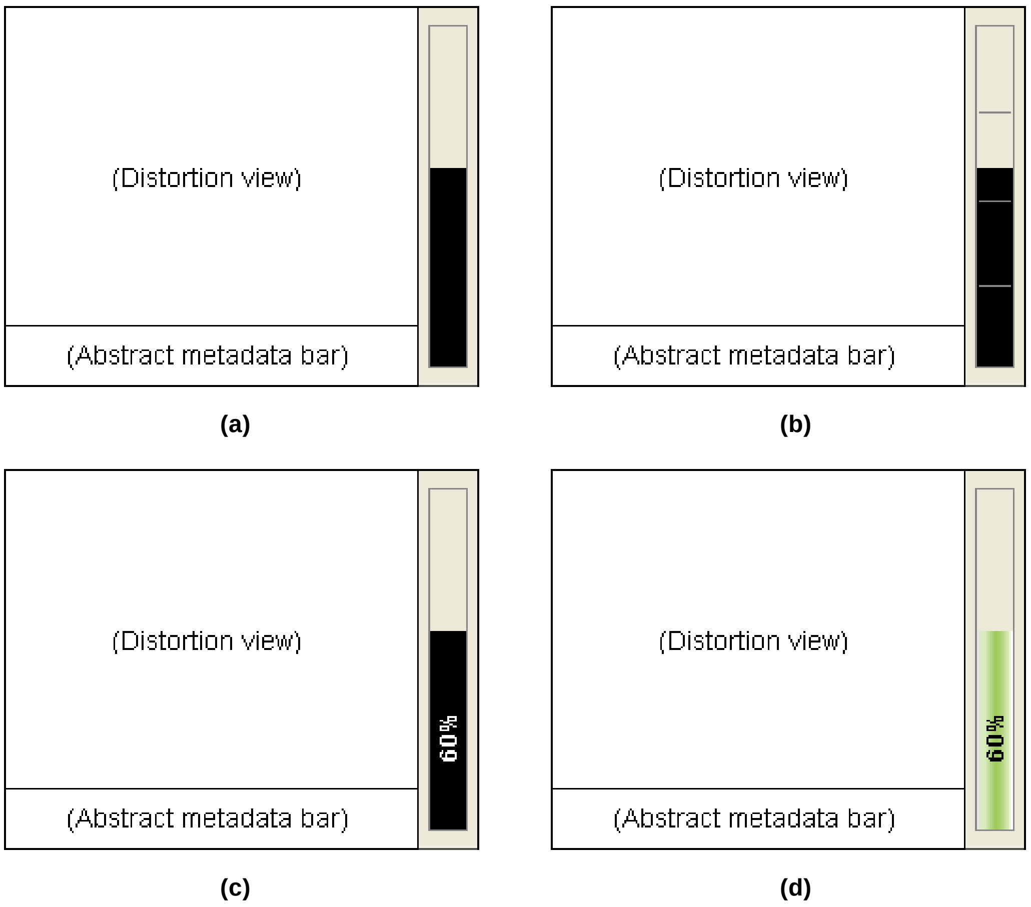


Figure 3.13 Some intersection bar designs - (a) simple solid color, (b) adding some more sticks for percentage reference, (c) adding numerical label, and (d) colorizing the column.

3.5 Summary

In this chapter, the two main problems for the thesis have been outlined and discussed. Distortion views and GUI design techniques are exploited for the task of generating and representing the Metadata Previewer. Depending on the position of the mouse cursor over the explored image, the selected watermarked object - if existing - is determined and its visual region is distorted and highlighted. The related watermarked data, known as the illustration metadata, are then extracted in abstraction formats and displayed. Due to the specific data issues, the work for the distortion view creation and presentation is fully applied, while the task for metadata presenting just correlates with GUI design problems without considering about the way the abstract data are extracted.

Markers in form of small dots are placed on the previewer to drive the viewer's attention to the watermark objects, and the selected watermarked region is determined by the highlighted contour. Average moments of the object pixels and its skeleton pixels are used for the definition of the markers' positions, and contrast colors are suggested for the contour high-light, based on the input metadata with a predefined data structure. The elastic presentation space proposed by Carpendale is employed for the task of creating the distortion view, with normal bell shape and the scaled-only magnification area are the two options for the previewer demonstration. The Gaussian drop-off function, Euclidean distance metric, and size for object region as options are the parameters considered for the implementation of the Metadata Previewer.

As an example for exploration application, the previewer is proposed to show the metadata in forms of file properties, as of data size, annotation filename, and their abstract information of central words for text file, view dimensions for image, or time length for audio and video. They are together with the file format (text, image, audio, video, or undetermined format) presented in the "abstract metadata bar", and the expected similarity for the searching data is displayed in form of chart bar – as quantitative visualization – in the "intersection bar". Since the methods mentioned here are just the proposals of my work for a specific tool, it gives the chance for us to discuss and suggest the other methods for generating and displaying the embedded metadata of the augmented images as well. However, they are the topics for the next chapters about results evaluation, future directions and conclusion, and we will now come to the next chapter about implementation issues and demonstration results.

Chapter 4

Implementation

In chapter 3, the methods for generating and displaying the metadata previewer have been theoretically studied and discussed. Distortion view based on EPS model and the metadata presentation using toolbar are the two subjects in consideration. Following the investigation, the *Metadata Previewer* project is developed and reported in this chapter. The structure of the chapter is as following. In section 4.1, the overview to the work and the programming environment are presented, and the detail implementation issues are described in section 4.2. Finally, the chapter ends with some screenshots for results demonstration in section 4.3.

4.1 Overview

The first point of this overview section is the reference to the programming environment. As my expectation, the implementation code is designed to be independent from any platform, portable and flexible for future extension or update. Consequently, it is written in C++ using Trolltech's Qt toolkit and the QuaZIP library for ZIP archive accessing. In detail, they are shortly narrated with the following software packages:

- **Trolltech's Qt toolkit[1]:** The C++ toolkit for cross-platform GUI application development that provides single-source portability across Microsoft Windows, Mac OS X, Linux, embedded Linux, and all major commercial Unix variants, developed by Trolltech. The main benefits for the project built with Qt are the support of a great amount of libraries that

1 http://www.trolltech.com/products/qt/

do not rely on the operating system, and the use of signals and slots mechanism.

Based on them, the source written for this project is quite short, just about 1200 lines of code not counting the incluced C++ libraries. The *Qt Open Source Edition* under GNU General Public Lisence - version 4.1.0, for MS Windows - is used, which is compiled through MinGW. For the whole project, different class libraries are employed, including:

`QtCore`: The module contains core non-GUI functionality, which is the basis for a Qt-based application. In this application, `QTimer`, `QFile`, `QList`, `QSize`, and some others are the classes belong to `QtCore` module.

`QtGui`: This is the biggest classes module supported in Qt, extends QtCore with GUI functionality. Different with the other toolkits which use the toolkit of the target platform for their implementation, for instance wxWidgets[2] or SWT[3], Qt uses its own paint engine and controls. It emulates the look of the different platforms it runs on, and this made the porting work easier since just few classes in Qt depended on the target platform. In my implementation, the classes of this module make the project code shorter, where `QImage`, `QMenu`, `QLabel` are some instances of visual widgets classes and `QPainter` is the main engine for interface drawing.

`QtXml`: Different with the above modules, which are found in normal Qt applications, QtXml is an add-on module just used for the special purpose of parsing XML documents. In my project, it is included in the task of loading the central XML file of the illustration watermarking container.

- **Minimalist GNU for Windows[4]:** The runtime headers set used in building a compiler system, based on the GNU GCC and binutils projects. Here, MinGW is employed as the C/C++ compilers for compiling and linking the Qt code in Win32 platforms.

[2] wxWidgets – Cross-platform GUI library, http://www.wxwidgets.org/

[3] SWT – Standard Widget Toolkit for Java, developed by Eclipse, http://www.eclipse.org/swt/

[4] MinGW - http://www.mingw.org/

- **QuaZIP for ZIP/UNZIP package[5]:** A simple C++ wrapper over the Gilles Vollant's Minizip[6] that can be integrated into Qt project, written by S. A. Tachenov. Though it is not a speed-up version, the code can be freely applied in my project to extract the data files in a zip IWM container.

The second point of the section refers to the workflow of the implementation tasks. Figure 4.1 shows the pipeline for the Metadata Previewer (MP) creating and displaying. In general, that pipeline contains three main parts. In the left column, the necessary data files are loaded as the system opens a new illustration watermarking image. On the right charts, the rest two parts are presented: the processing for view creation and the displaying of resulted Metadata Previewer.

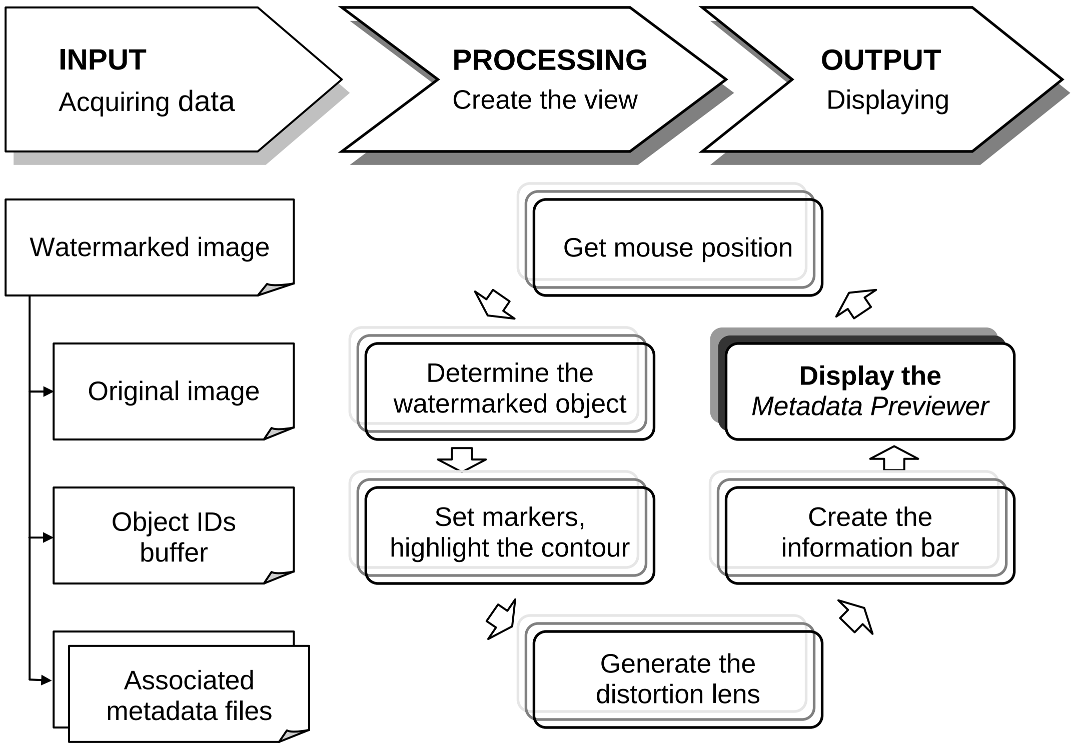

INPUT
Acquiring data
PROCESSING
Create the view
OUTPUT
Displaying
Watermarked image
Original image
Object IDs buffer
Associated metadata files
Get mouse position
Determine the watermarked object
Set markers, highlight the contour
Generate the distortion lens
Create the information bar
Display the Metadata Previewer

Figure 4.1 The pipeline for *Metadata Previewer* creation and displaying.

The input file is a zip archive that contains all necessary data files that have been mentioned in § 3.2.1. The system opens the file and unzips it using the QuaZIP library. For describing those data, a central XML document is included

[5] QuaZIP - Qt/C++ wrapper for ZIP/UNZIP package - http://quazip.sourceforge.net/

[6] Minizip: Zip and UnZip additionnal library - http://www.winimage.com/zLibDll/minizip.html

in the package. The system parses the XML and loads the data files to memory for future processing. Detail XML data structure for that description will be presented in section 4.2.1.

The main programming task is the creation and displaying of the previewer window. Detail algorithms are presented in sections 4.2.2, 4.2.3, and 4.2.4, which follow the theoretical discussions in chapter 3, section 2 to 4. In each section, beside the pseudo-code for the detail algorithm, the sequence diagram for the implementing task is sometime given and explained, if necessary. Generally, the project is slightly managed and UML[7]-documented. Here, we will get the overall view to the implemented classes through the collaboration diagram shown in figure 4.2 with the following classes:

- **`MPFrm`:** The main form of *Metadata Previewer* application. This class controls the whole GUI for MP with menu, toolbar, canvas client area, and the previewer window. All GUI objects are Qt instances, except the canvas and the previewer. The functions and slots defined in this class are for event handling of file manipulating and distortion view options.

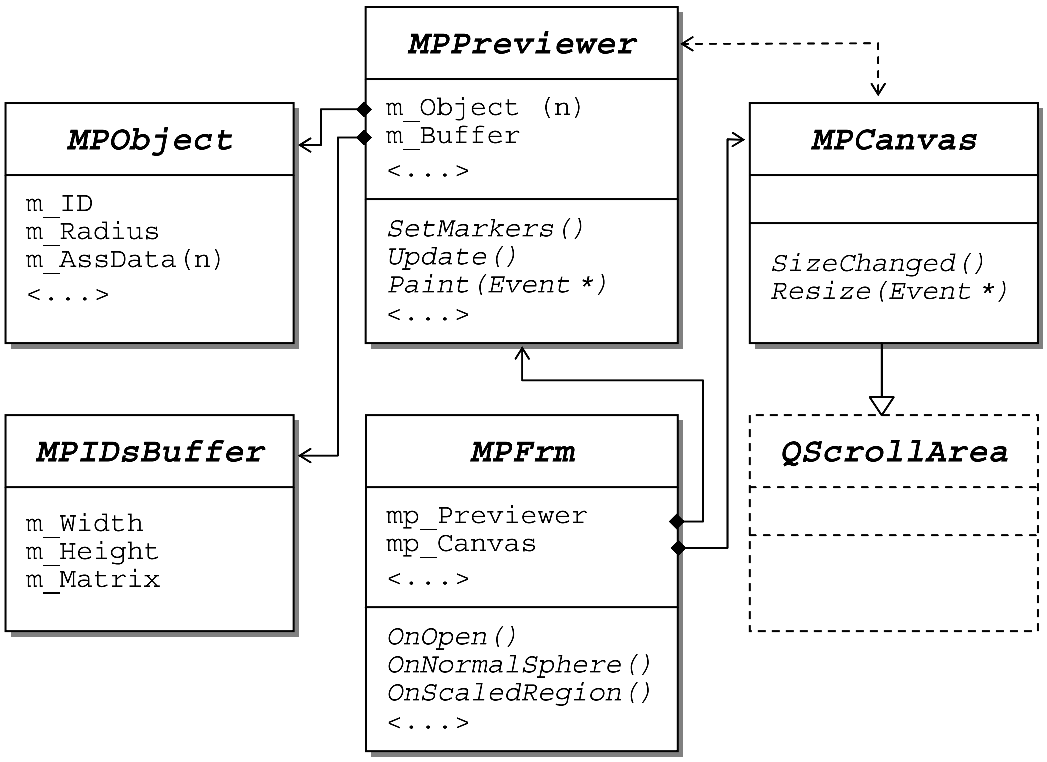


Figure 4.2 Overview diagram for implemented classes of MP project.

[7] Unified Modeling Language - http://www.uml.org/

- **`MPCanvas:`** The client area of MP is a scroll window `QScrollArea`. Since we need to adjust the previewer size according to client size, as mentioned later, the derived class is defined with *`SizeChanged`* slot for the task.

- **`MPPreviewer:`** This is the most important GUI class in my project. The class contains list of objects data, object IDs buffer, and all necessary data for the painting task. Beside the *processing* functions for previewer window generating, events for mouse movement, timer, and painting are the other methods in consideration.

- **`MPObject:`** Single watermarked object is described through this class, where center point, object's radius, ID, and the associated metadata are maintained.

- **`MPIDsBuffer:`** Together with `MPObject`, `MPIDsBuffer` is another non-GUI data class used in `MPPreviewer`.

4.2 Implementation Issues

The above section has generally introduced the whole programming task for the MP, with the implementing workflow and its overall classes. Detail to implementation issues, for instance the definition and management of application data structure, the detail programming algorithms, and the illustrated sequence diagrams are referred to in this section. The subsections are presented following the tasks order mentioned in figure 4.1.

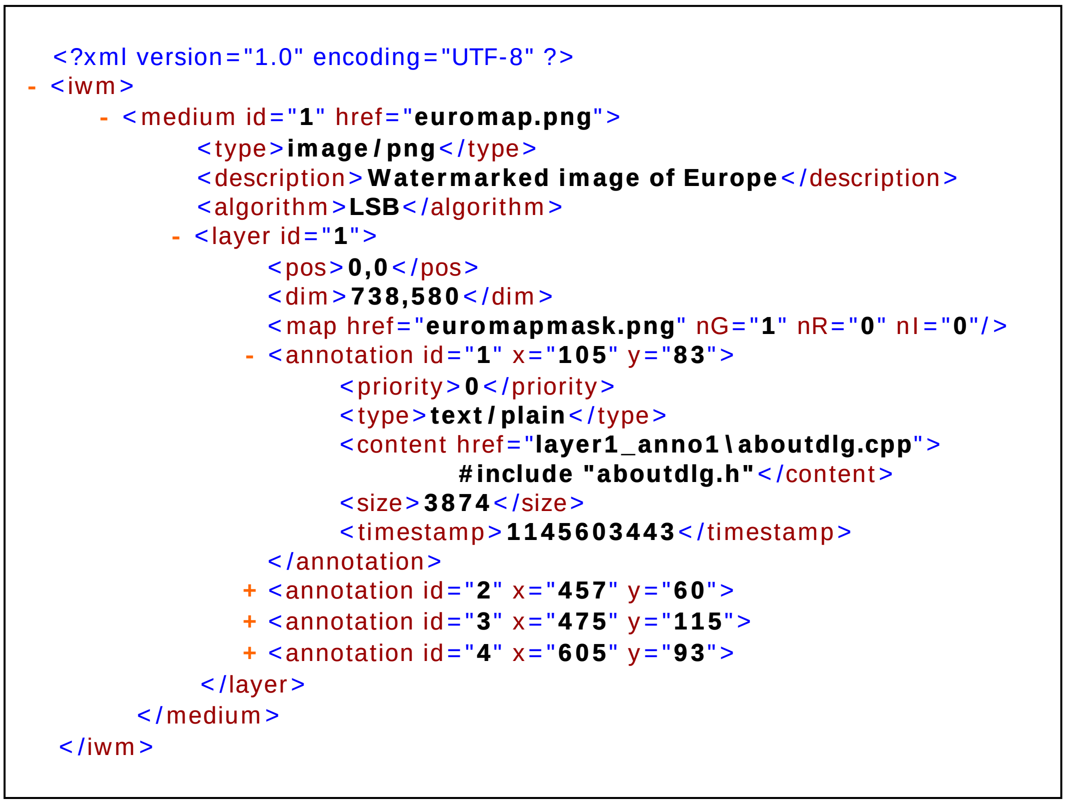
4.2.1 Data Structure

As presented, the MP tool loads the watermarked image in the form of a zip archive and produces the necessary data files for future processing. The data files are described through a central XML document. In figure 4.3, an example for such XML structure is given, which explained as below:

- **Medium:** The un-watermarked image, or the "original image" mentioned in §3.2.1. There can be more than one such image in an IWM container. For demonstration, the number of mediums in my application is only one. File name, file type, medium description and the watermarked algorithm are some children elements, which can be used separately depending on the application purposes. In MP, just `href` for filename is used.

- **Layer:** For each medium there should be some layers with their own ID. The layer can be known as a mask image buffer with its position and dimensions determined by the medium image. The object IDs buffer is a mergence of all such layers.

- **Annotation:** In a layer, one or many annotation elements are constructed. This annotation denotes the relevant watermarked object in the layer with information about the object ID and its associated metadata files.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <iwm>
    - <medium id="1" href="euromap.png">
          <type>image/png</type>
          <description>Watermarked image of Europe</description>
          <algorithm>LSB</algorithm>
        - <layer id="1">
              <pos>0,0</pos>
              <dim>738,580</dim>
              <map href="euromapmask.png" nG="1" nR="0" nI="0"/>
            - <annotation id="1" x="105" y="83">
                  <priority>0</priority>
                  <type>text/plain</type>
                  <content href="layer1_anno1\aboutdlg.cpp">
                        #include "aboutdlg.h"</content>
                  <size>3874</size>
                  <timestamp>1145603443</timestamp>
              </annotation>
            + <annotation id="2" x="457" y="60">
            + <annotation id="3" x="475" y="115">
            + <annotation id="4" x="605" y="93">
          </layer>
      </medium>
  </iwm>
```

Figure 4.3 XML document describing the data files for an IWM container.

By using XML data structure, the task for files management becomes more flexible and exchangeable. The data file can be easier managed and updated in the IWM authoring tool, delivered to different applications, and selectively used in specific softwares. In my application, some data elements in the above XML structure are not in used. Beside, since the data are kept in files which are packed into a zip archive, the hyperlinks to the files have the direct location names. Web-link and database link are some alternative solutions, as presented in the previous chapter.

4.2.2 Objects Marking and Highlighting

The watermarked areas in the augmented image are determined through their region's central points, and the selected object is highlighted by the contrast color contour, as presented in the previous chapter, section 3.2. Here we come to the implementation tasks. In the XML structure shown in figure 4.3, those marker points are manually specified (by the authorizing tool), for instance the marker of the annotation object ID=1 is positioned at point (105, 83). Whenever a new XML annotation element is parsed, the marker is created, together with a new MPObject instance in the application's memory. If such central point has not been specified yet, the system automatically computes it using the formula 3.1 or 3.2. Of course, the marker can also be re-calculated later through the application options. The sequence diagram for such implementation is presented in figure 4.4.

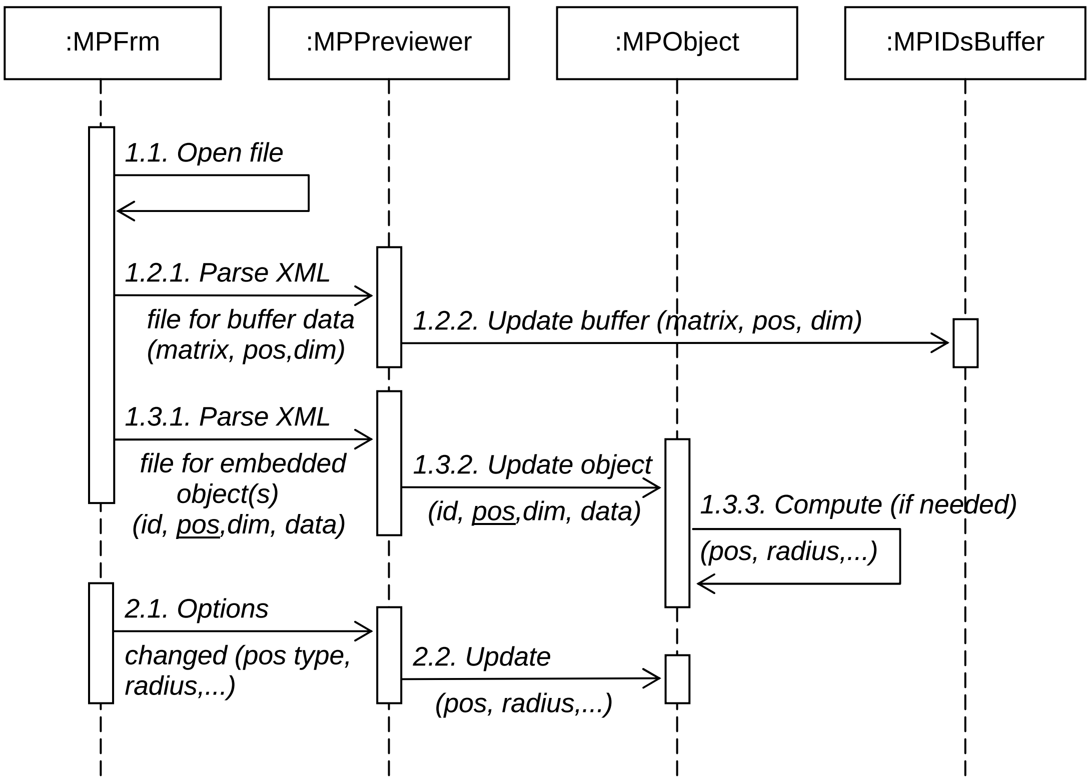

:MPFrm
:MPPreviewer
:MPObject
:MPIDsBuffer
1.1. Open file
1.2.1. Parse XML file for buffer data (matrix, pos,dim)
1.2.2. Update buffer (matrix, pos, dim)
1.3.1. Parse XML file for embedded object(s) (id, pos,dim, data)
1.3.2. Update object (id, pos,dim, data)
1.3.3. Compute (if needed) (pos, radius,...)
2.1. Options changed (pos type, radius,...)
2.2. Update (pos, radius,...)

Figure 4.4 Sequence diagram for updating the application's data.

The mask buffer for the watermarked objects is also created according to the input data, as we have just seen in the diagram. Having such data, whenever a watermarked object is selected, its contour is computed for the highlighting task. The working process is illustrated in figure 4.7 for diagram of generating and displaying the distortion view. The pseudo-code algorithm for such contour creation is presented in figure 4.5.

```
for r from 1 to CONTOUR_SIZE do
   specify the color color for the contour pixel,
   e.g. BLACK and CYAN in my application
   determine the contour pixel and set color for it, by
      for y from top to bottom of the object area do
         for x from left to right of the object area do
            if buffer pixel MPIDsBuffer[x][y] != ID and at
            least one of its surrounding pixel is ID, where
            ID is the object ID then
               mark it as the contour pixel
            fi
         od
      od
   set color for all contour pixel in the image, and assign
   them the value ID for the next loop
od
```

Figure 4.5 Pseudo-code algorithm for highlighting the selected object contour.

4.2.3 Generating the Distortion View

The methods for distortion view generation and displaying have been particularly analyzed and presented in section 3.3, where EPS model developed by Marianne Carpendale is used for the lens creation, and the bell-shaped with Gaussian drop off function and the scaled-only focus region are the two lens-manipulating options. Using the basic techniques from EPS, we did know that it leads to the matter of showing gaps inside the distorted image (see §3.3.2), and the tactics for lens adjustment were suggested. Here, simplified algorithm for such lens creation is presented, as in figure 4.6.

Some remarks should be noticed in the above algorithm:

- **Markers**: The markers are assigned at the central points of the watermarked objects. However, in the case of bell shape distorting, the marker is quite large and covers most of the view area. Thus, it is necessary to give an option for selMarker (marker for the selected region) - see points `(1)`, `(3)` - to be displayed or not.

```
void LensCreation(bool hasObj, bool selMarker,int lensType){
   if (!hasObj){ // no object is selected for distorting
       (1) set markers for all object;
       (2) create the scaled view for the previewer;
   }
   else {
       (3) set markers depending on selMarker;
       (4) compute raw distorted image size and create the raw
            image;
       (5) highlight the selected object contour, using
          algorithm in Listing 4.2;
       switch(lensType){
             case BELL_NORMAL:
                 (6) for each point (x_i, y_i) in the object
                      matrix pixels
                    (6.1) compute its distance to the central
                          point (x_c, x_c);
                    (6.2) compute the distorted point (x_m, y_m);
                    (6.3) update the raw distorted image;
                break;
             case SCALED_ONLY:
                 (7) compute the height h_m for scaled region;
                 (8) for each point (x_i, y_i) in the object
                      matrix pixels
                    (8.1) compute its distance to the central
                          point (x_c, x_c);
                    (8.2) compute the distorted point (x_m, y_m)
                          according to its position (distance)
                          from the central point and h_m;
                    (8.3) update the raw distorted image;
                break;
       }
       (9)  create the expected final image;
       (10) scale the raw image by scanning each block of raw
            pixels;
            (10.1) get its average color;
            (10.2) assign color value to the relevant pixel in
                   the expected previewer;
   }
}
```

Figure 4.6 Simplified algorithm for distortion lens creating.

- **Scaling**: The scaling task - `(2)`, `(10)` - to get the expected previewer is always done by computing the average color for a block of pixels, instead of using Qt function, even for non-distorted scaling `(2)`, since it produces the better result.

- **Lens type options**: The code for each type is done separately - `(6)`, `(7)`, `(8)`, though they are implemented using the same formulae (see chapter 3), since we need to save the speed.

- **The previewer size**: The size of the displaying "window" is not a constant, though we expected it to be a constant by specifying a predefined value. Instead of that, it is relatively computed based on the object size in the original image - `(4)`, `(9)`.

By the way, what I have just mentioned as the previewer "window" is just the visualizing presentation of the distorted image and the information bar (see §4.2.4). That window is not the implemented `MPPreviewer` widget, for the Qt widget need to capture all mouse events on the client area, the `MPPreviewer` is assigned with the `MPCanvas` size. Complete working process for events manipulations are illustrated through the sequence diagram in figure 4.7.

4.2.4 Creating the Information Bar

The implementation for information bar presentation is the shortest part in my project. Suppose that all data needed for the intersection bar and the metadata bar displaying are available, what the MP has to do is just showing them. Thus, the code is directly included in the painting function of the `MPPreviewer`. In sequence diagram of figure 4.7, it is the *5.1 Paint / Refresh* function.

Following the concepts suggested in §3.4, the function consists of 3 parts:

- **Toolbars drawing**: As I have just mentioned in the last section, `MPPreviewer` is a widget with size equal to the `MPCanvas` size. It means that the window is transparent over the canvas. Thus, we should paint everything to show the *previewer*. The background area and its 3D borders are painted close to the distorted image to make the toolbars.

- **Intersection data drawing**: The data is drawn as a graphical chart. With the input value for percentage of similarity, relevant area in chart is computed, and a stretchable image is painted to that area to show the intersection data.

- **Selected metadata drawing**: Due to the limited space, the "abstract" metadata with small icon and short text are alternatively painted onto the bar. The task is accomplished whenever a new timer event occurs.

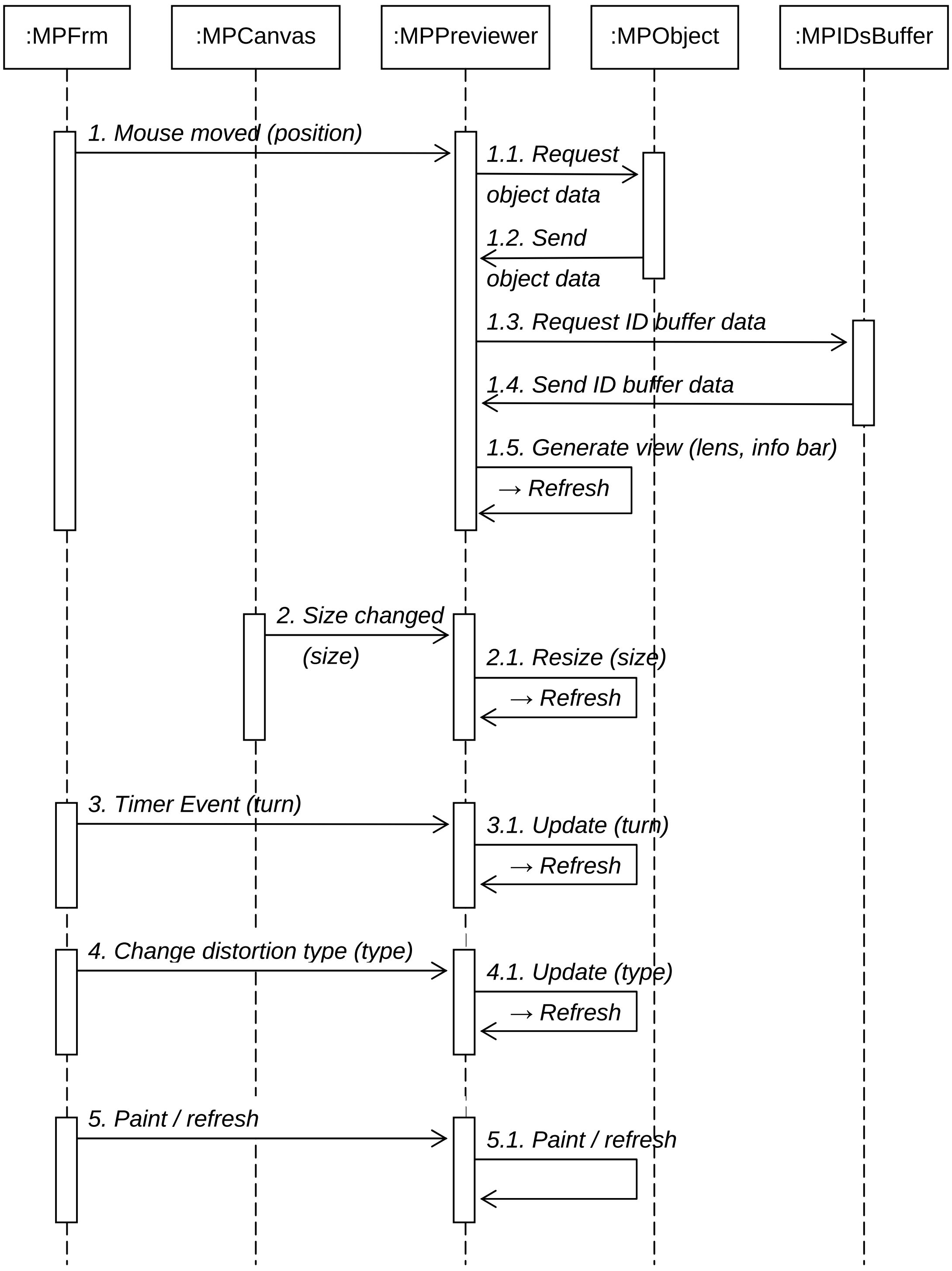

:MPFrm
:MPCanvas
:MPPreviewer
:MPObject
:MPIDsBuffer
1. Mouse moved (position)
1.1. Request object data
1.2. Send object data
1.3. Request ID buffer data
1.4. Send ID buffer data
1.5. Generate view (lens, info bar) → Refresh
2. Size changed (size)
2.1. Resize (size) → Refresh
3. Timer Event (turn)
3.1. Update (turn) → Refresh
4. Change distortion type (type)
4.1. Update (type) → Refresh
5. Paint / refresh
5.1. Paint / refresh

Figure 4.7 Sequence diagram for event manipulations of previewer window creation and displaying.

In figure 4.7, the handling events are not completely exact to what is implemented in the classes. For instance, the mouse move event and the refreshing (repaint) event are realized in `MPPreviewer`, but I sketch them as events from `MPFrm` since they are dispatched from there. Similarly, the *paint / refresh* event occurs whenever the window needs to be updated.

4.3 Results

In this section, a list of screenshots captured from my *Metadata Previewer* tool is demonstrated. Accompanying with them are their descriptions. Furthermore, some short discussions are also included to give a better view to the readers. Following the implementation process and the application options, the screenshot figures are presented like that:

- In figure 4.8, a full screenshot of the Metadata Previewer with notes to the visual elements is shown. This gives an overview to the tool so that we can realize the demonstrating objects shown in the next figures easier.
- In figure 4.9, we will see some different shapes of the previewer, according to distorted view size and information bars size and format.
- Figure 4.10 shows some views with different distortion lens parameters for the same selected object.
- Figure 4.11 displays the previewer with different views for the watermarked objects determination through options for markers and the highlighting contour.
- And finally, figure 4.12 presents the previewer over a selected watermarked area with options for lens shapes.

In all cases, the demonstrating results for the previewer are presented with distortion lens and information bars created, except the one shown in figure 4.9a for previewer navigating over the unembedded (not watermarked) area.

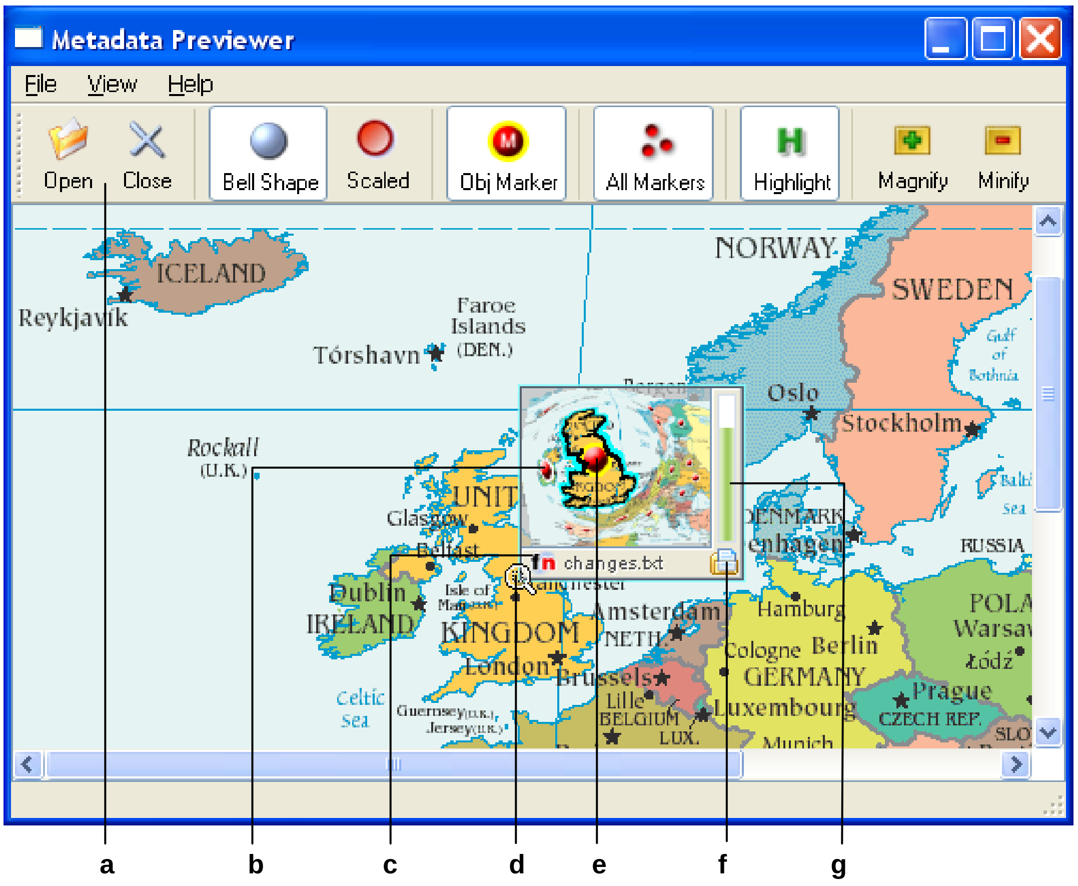


Figure 4.8 Screenshot for Metadata Previewer with notes:

a. Application toolbar with options for: open file, close file; change shape for the distortion lens (bell shape as the normal one, and the scaled-only focus region); hide/show the marker of selected object, all markers of all objects, or the highlight contour of the selected region; and magnify / minify the previewer. In this example, the normal distortion view with all markers and object high-light is chosen.

b. When distorting the selected object, the markers for other watermarked areas are also distorted, respectively. Looking at them, we do not only know the object place, but also can determine the distance for further navigation.

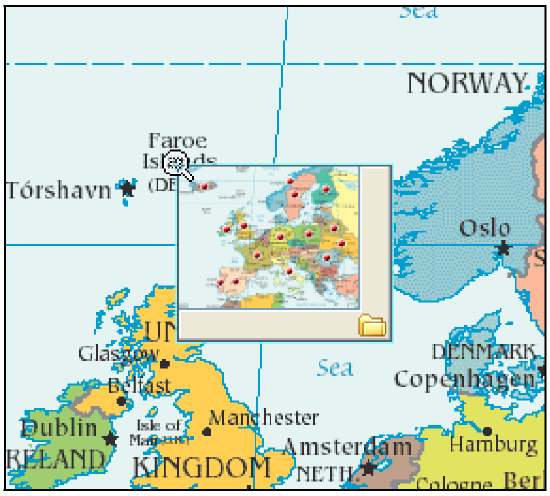
c. Small icon and text shown in the *abstract metadata bar*. It is animatedly changed in turn. Here, text content of a document file is presented.

d. The mouse cursor for position over the selected object. In this case, it's over the map of United Kingdom.

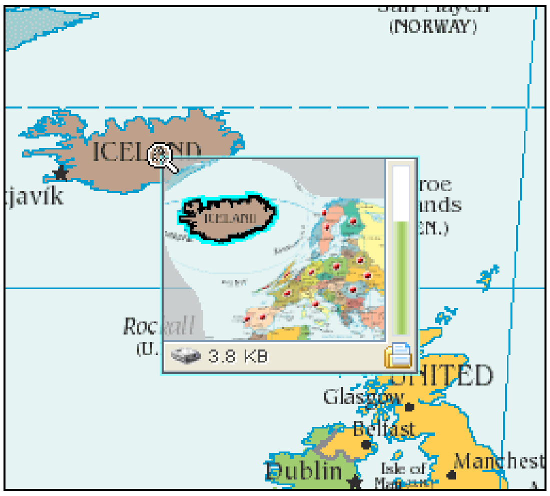
e. Distorted object shown with displaying options. In this image, object marker and highlight contour are included.

f. Icon for file type (not animated). The above icon indicates a text file.

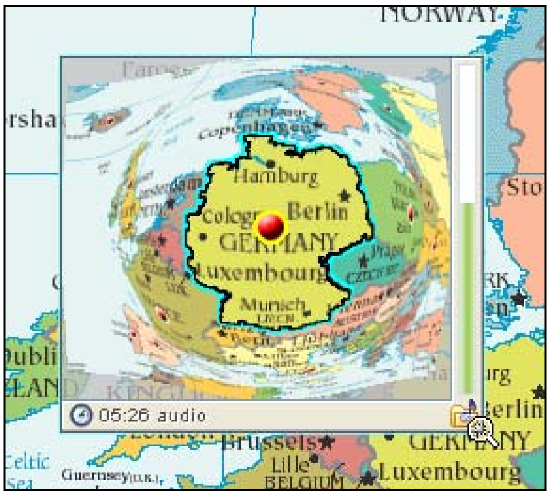
g. Intersection bar with amount of similarity to the expected searching data.



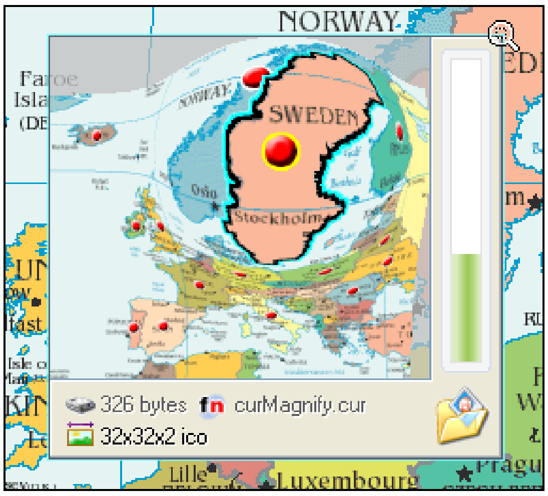
(a)



(b)



(c)



(d)

Figure 4.9 Different shapes of the previewer, according to its distorted view size and information bars size and format:

a. Previewer size ≈ 75 pixels, small-animated information bars, moving over non-watermarked area.

b. Previewer size ≈ 75 pixels, lens radius = 175 pixels, small-animated information bars, moving over watermarked area of Iceland.

c. Previewer size ≈ 150 pixels, lens radius = 350 pixels, small-animated information bars, moving over watermarked area of Germany.

d. Previewer size ≈ 150 pixels, lens radius = 300 pixels, big-full information bars, moving over watermarked area of Sweden.

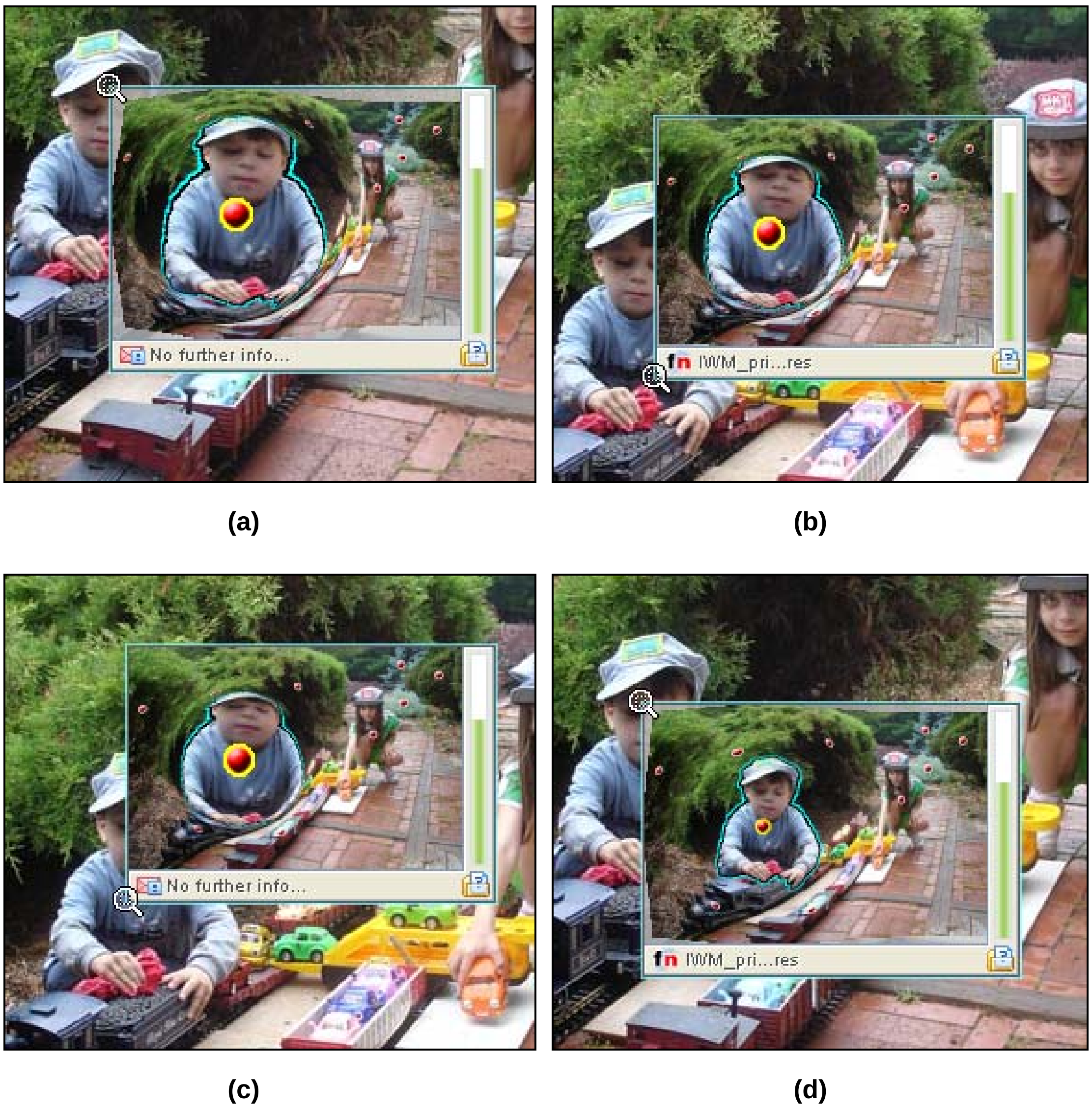


Figure 4.10 Previewers with different distortion lens parameters for the same selected object, and:

a. Lens radius *rad* = 250 pixels, Gaussian standard deviation σ = 0.5, distance from RVP to base plan d_b=100, maximum focus height h_f=75.

b. *rad* = **2 × object radius**, σ = 0.5, d_b=100, h_f=75.

c. *rad* = 250 pixels, $\boldsymbol{\sigma}$ **= 0.2**, d_b=100, h_f=75.

d. *rad* = 250 pixels, σ = 0.5, d_b=100, $\boldsymbol{h_f}$**=50**.

The previewer size is ≈ 120 pixels with small-animated information bars designed.

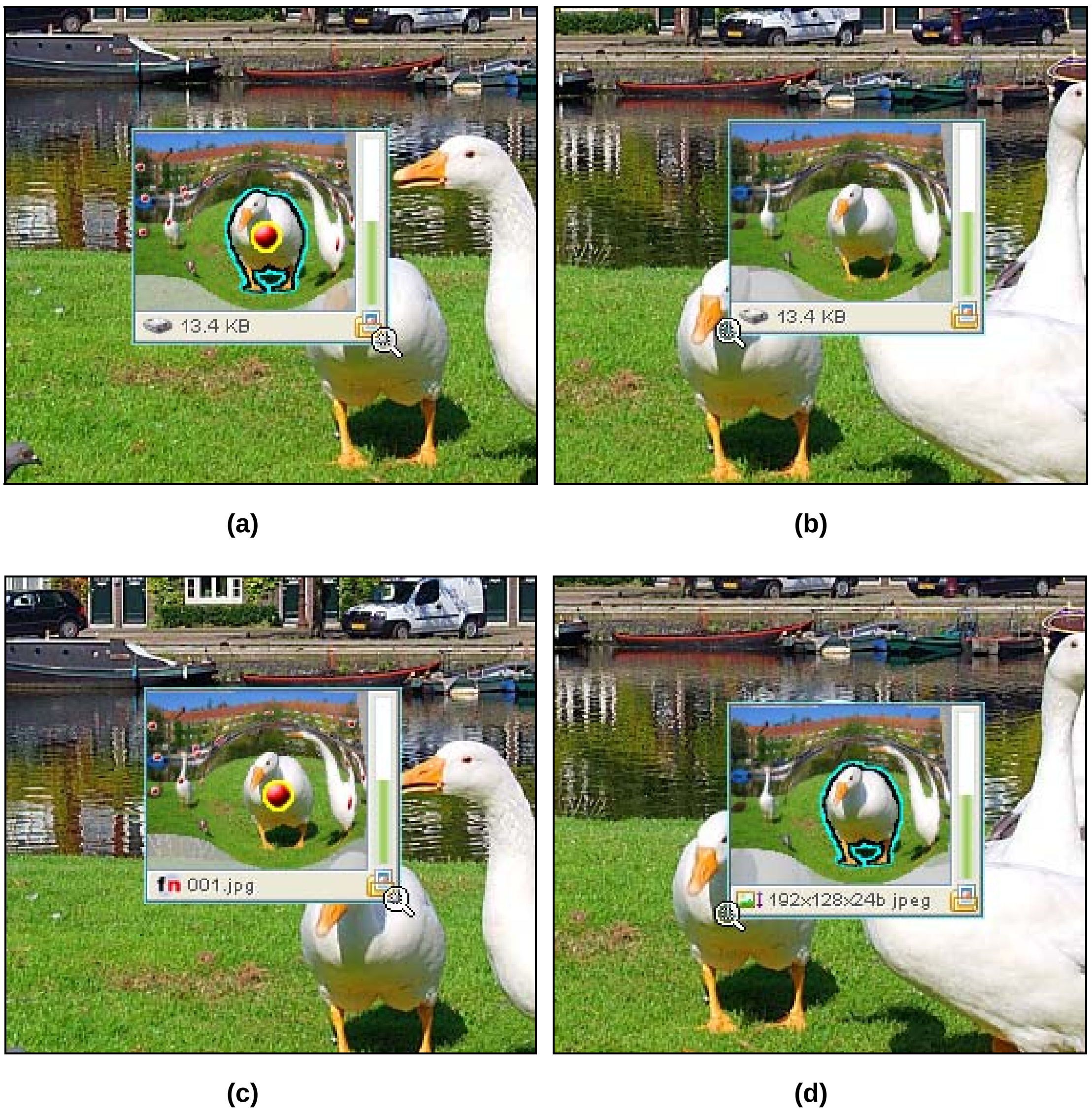


Figure 4.11 Previewers with different views for the watermarked objects determination through options for markers and the highlighting contour.

a. Including all markers for objects determination, highlight contour and marker for selected object.

b. Without including anything.

c. Just including the markers for objects navigation.

d. Just including the highlight contour for selected object determination.

The previewer size is ≈ 75 pixels with lens radius = 175 pixels and small-animated information bars designed.

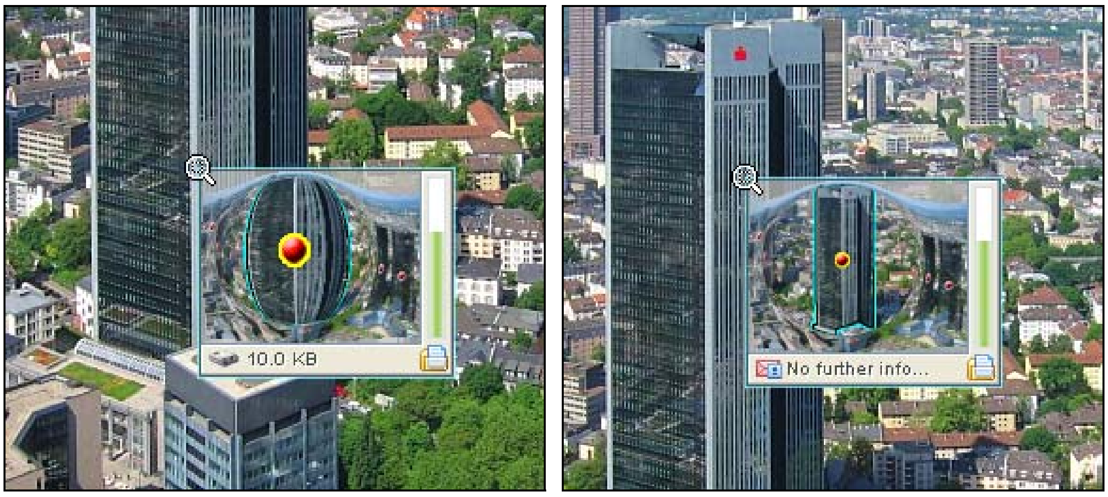


Figure 4.12 Previewers over a selected watermarked area with options for lens shapes:

a. Bell shape with Gaussian drop-off function and Euclidean distance for points to the focus computed.

b. The scaled-only focus region view with the similar parameters presented in (a).

The previewer size is ≈ 75 pixels with lens radius = 175 pixels and small-animated information bars designed.

Chapter 5

Evaluation and Future Directions

The results that were presented in the previous chapter have closed the main work for my thesis, and open some topics for evaluation and discussion in this chapter. In section 5.1, the general introduction to the open directions will be given following the general assessment to the thesis work; and some more details are then mentioned in the later sections. In section 5.2 we will find the topics for *metadata presentation revision*, and the discussions about the *previewer's navigation* will be mentioned in section 5.3. Finally, a general proposal to *multilevel illustration watermarking system* with the specificity to *metadata management* is presented in the last section before we come to the last chapter for concluding the thesis.

5.1 A General Assessment to the Work

Every work results with its own fruits, which are some time sweet, but some time disappointed to the workers. For that, a general discussion and evaluation to the research and its achievements would be useful to the researcher, especially for his further investigation. In this section, I would like to generally represent the work that has been done in my thesis, for both methods and implementation results, through my scale for success and limitation, which leads to some open future directions.

The common way for evaluating a work is measuring its correctness according to a predefined hypothesis or data set. However, that way seems not to be relevant to the problem of metadata presentation, since there would be no general criterion for a visualization technique measurement. The most suitable way to assess a presentation technique or framework should be done through the experience and feedback from the experts and users. Thus, user case study would be necessary in order to complete my project. Due to the limited time and the thesis' scope, I would not accomplish the evaluation, but let it be one of the open directions for the future investigation.

Consequently, in this section, I would like to talk about the work regarding three main points (i) what I should do next to accomplish the tool so that it can be released, (ii) which improvements can be applied to the current implementation, and for which parts to get a better application, and (iii) whether there are any new ideas for the work which may lead to new methods, suggestions for better solutions.

About the first point, there are some illustration parts that have not been completely implemented in the Metadata Previewer. In my current tool, they are manually defined for the demonstration results. Thus, we need to accomplish the three following problems:

- *The watermarked input data*: For a deliverable application, stable data structure for the watermarked images should be defined and implemented in the whole system. However, in my MP, it has not been done. For the demonstrating results, I just partly use IWM archive for the input data. Thus, developing or employing a library for such watermarked image structure is expected to be implemented. Where, libSmage [22] or IWM [10] data structures can be used for the overall illustration watermarking system.

- *Metadata for information bars displaying*: The main task of my project is designing the previewer and displaying it. Nevertheless, the code for creating such previewer based on the input data and the manipulation event from the user is also not completely done. I did implement the functions for creating the distortion lens, but not those for the task of generating the data for information bars displaying. Therefore, they should be implemented or inherited from the other libraries.

- *Marker positions*: The central moment of region's skeleton algorithm that was analyzed as the best solution for computing the central points of the watermarked objects has not been implemented yet. Thus, it would be considered for the future work.

Such processing data are mandatory for the application to be releasable, and they are the implementation issues. Beside, as mentioned in chapter 3 and 4, some theoretical methods for the improvement of the MP should also be applied to get a better visualizing previewer. Here, I would like to list them once again:

- *Selected object contour*: The object contour must be generated after we get the distorted image to get a better presentation. In the current tool, such creation is just done in the pre-processed image which leads to some exclusion of distorted pixels in the resulted image.

- *Specific data format for metadata presentation*: We have just mentioned about the implementation tactics for extracting such metadata. It means that we only tackle the question of "how to implement the task." However, that is not enough. In general, we should also answer the question for "what should we display on the bars?" For that, it would be nice to make some user study surveys with different kinds of data through a collection of presentation forms, and get the most suitable one for the specific application.

By the way, the crucial target of this chapter is presenting some open topics for the future investigations based on the current work. Since the techniques for visual interface design are widely investigated and applied, there would be no best solution to this problem among a variety of methods that can be used. Thus, after making discussion with some researchers, I just want to revise the matter through the following topics:

- **Logical or physical metadata presentation**: The concept of illustration watermarking and the method for metadata presentation using distortion technique have opened a general view to this research of metadata presentation. In detail, distortion view is used to specify the watermark object which is not in its original form, and "abstract" metadata are selected to be displayed according to the requirements of the users. In other words, they are really the visual presentation in a logical form, not the physical one.

  In section 5.2, we will learn about this topic where the concept of the *Cascading Previewers* is defined for multilevel metadata, as a general sample for the topic.

- **Navigation with the previewer**: A well-designed HCI application for metadata exploration should not only display the previewer for metadata information, but also support the interactive interface for data searching and navigation. *Temporal continuity* for distorted image presentation and *cursor assistance* are some open topics for tackling this matter, as presented in section 5.3.

- **Multilevel metadata**: The embedded data in the illustration watermarked do annotate the objects that they associated with. In my project, watermarked data is just simple metadata which can be easily manipulated through a file. However, in a general case, an illustration watermarking system is expected to be able to encode different metadata into an object, and the embedded data can again be data that include further links. That leads to a system for multilevel illustration watermarking. In our discussion, *cascading previewers* is the example for a multiple level metadata presentation.

5.2 Revision for Metadata Presentation

We will discuss about the matter for *symbolic presentation* of the watermarked metadata through an example about map exploration. Let's consider Google Earth[1], the current well-known application for exploring, searching and discovering our planet using multiple zooming and navigating levels. At different zooming steps, displayed data are specifically presented on the browser with different data formats. Such data are not physically zoomed or distorted from the previous or later step images, but geometrically loaded from the predefined database in Google servers.

Returning to our own problem, metadata are those that describe the watermarked objects. For that, with a map image, we can encode not only one metadata, but also a bunch of information with multiple metadata levels into an area inside the map. Suppose that we have a map of Europe as demonstrated in the previous chapter, but with multiple illustration watermarking levels. At the place of Germany, we suppose to watermark a detail map of the country, an image for its coat of arms, and a short writing about the politics. Similarly, the map of Germany is also embedded with information for the federal states, as the other illustration watermarking images, and so forth. Our task now is navigating over the country and exploring its associated metadata.

In this case, it should be better not to use only distortion lens for representing the selected area. Whenever the previewer is moved over the map of the country, the distortion view highlights the object and some abstract information is displayed as we have designed. Then, for detail information of the country, we lock the

[1] Google Earth – A 3D Interface to the Planet, http://earth.google.com

current navigating process by clicking on the image, pop up a new previewer and do further exploration. The new previewer might include the client area for metadata presenting and toolbars for further navigation. Choosing a tab for relevant information, we can get the necessary data, and continue to explore if the exploring image is another illustration watermarked image as well. The sequence of such hone-in windows for metadata exploration is defined as *cascading previewers*. In figure 5.1, we can find a sketch for the concept.

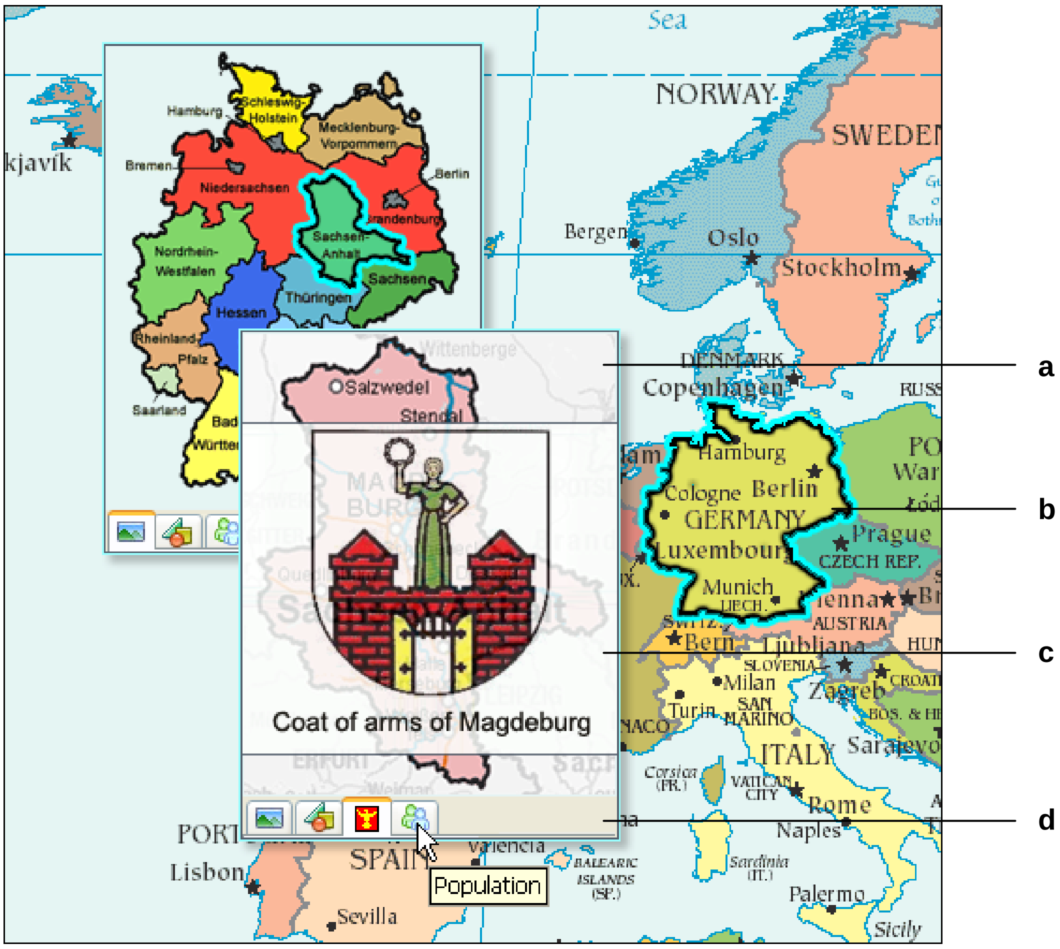


Figure 5.1 Cascading previewers: Exploring the coat of arms of Magdeburg.

a. The second level previewer showing the metadata for state Sachsen-Anhalt.

b. The selected watermarked object. In this example it is the map of Germany for the first level exploration.

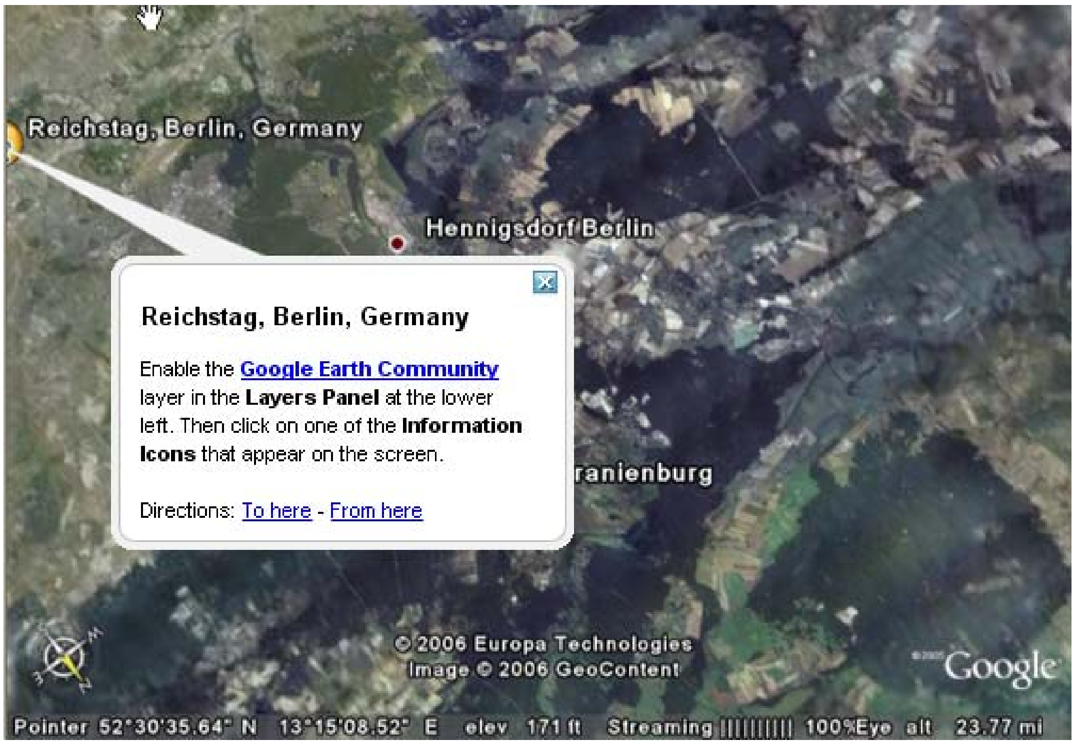
c. Metadata presentation according to the selection in previewer's information bar.

d. The toolbar with icons and tooltips for metadata displaying options.

The hierarchic structure for multilevel metadata presentation just mentioned adapts well the basic requirements for a usable illustration watermarking system. It does provide flexible navigating action, easy level selecting and exploring manipulation, i.e. we can switch to an exploration level at any time by clicking on the relevant previewer widget, and keep the working context and the presentation space visible and manageable all the time. Of course, this is not the only way for interface design and presentation for multiple level illustration watermarked image, since we can investigate other techniques for graphics, geometry, or illustrating items among the tremendous world of visual interface design theory.

5.3 Previewer's Navigation

In my current development, there is no assistance from the system for the user to navigate the previewer to a searching-metadata area, except the markers for objects determination. An interactive interface application should be better if giving some solutions to that matter. For instance, in Google Earth, whenever the planet is explored, not only the markers, tool tips, or highlight contours for places and areas are displayed, but also an arrow with explanation panel is included to inform the searching information and the direction to the object. Figure 5.2 is an image captured from Google Earth that illustrates the idea.



Reichstag, Berlin, Germany

Enable the Google Earth Community layer in the **Layers Panel** at the lower left. Then click on one of the **Information Icons** that appear on the screen.

Directions: To here - From here

Pointer 52°30'35.64" N 13°15'08.52" E elev 171 ft Streaming |||||||||| 100% Eye alt 23.77 mi

Figure 5.2 Google Earth: Explanation panel and arrow pointer for the direction to the searching place of *Reichstag, Berlin, Germany*.

Similarly, HCI methods can also be investigated and applied to my tool as the open topics for future work. Arrow pointer for direction to the searching metadata, temporary continuity for distortion lens creation, animated markers, and tool tips are some techniques that can be used to improve the *Metadata Previewer*. Different topics can be discussed for this problem, for instance:

- What shape of arrow pointer is suitable for the MP? And where is the best place to lay the arrow? For instance, we can design an opaque shape for the arrow pointing to the searching metadata, and that arrow is settled over the previewer widget.

- How should we inform the objects nearby the previewer when it is navigated over the image? Simulating the distortion lenses' sizes and shapes would be a nice solution to this matter. The idea is as close to an object for the MP as bigger size and magnification level for the watermarked object. Figure 5.3 shows a simple example for this tactic.

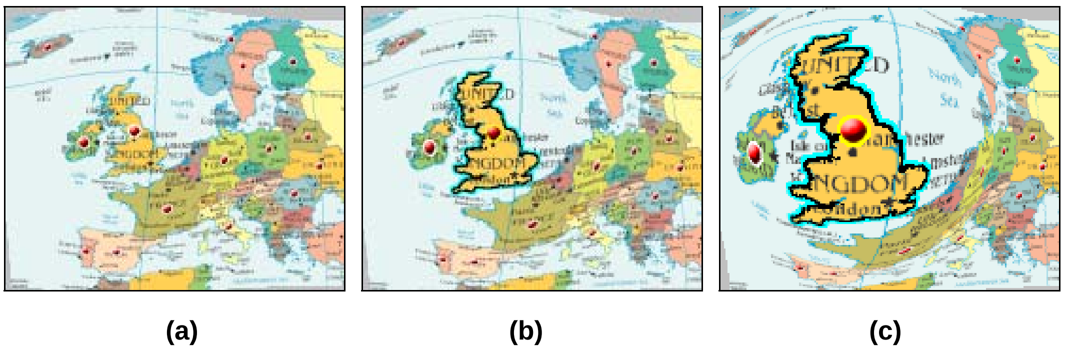


Figure 5.3 Interactively simulating the distortion lenses for object determination when MP is moved over the explored image. In (a) the cursor is outside the watermarked region of United Kingdom, but in a close distance. In (b) the cursor is at the border of the selected region, and (c) is at the region's center.

For faster searching to some expected metadata, beside the markers or labels for the objects, "intersection" bars or charts are also in consideration. A good interface with such assistance can help the user to save the time for checking all the objects to find the necessary metadata, as supported in my current MP tool.

5.4 Multilevel Metadata Management and MIWS

Data management is an important aspect in almost all computer applications, especially in this age when multimedia are increasingly growing in size and collections. Depending on the applications' purposes, different databases or data structures are defined and used for different applications. Thus, an illustration watermarking multimedia system does need a well-established data management scheme beside the tools for interactive graphics presentation and data manipulation. Considering that problem for my MP tool, the idea is really reasonable, since the watermarked images - in their own definition - are suggested to be managed and used similar to any normal image in PNG, JPEG or other formats. Therefore, IWM data could not just be seen as an archive file containing the descriptions. The watermarked images and the associated metadata should be managed in special ways so that they are unique and relational to each others in a collection or network. This makes the task for storing or later retrieving the multilevel metadata much more convenient and flexible.

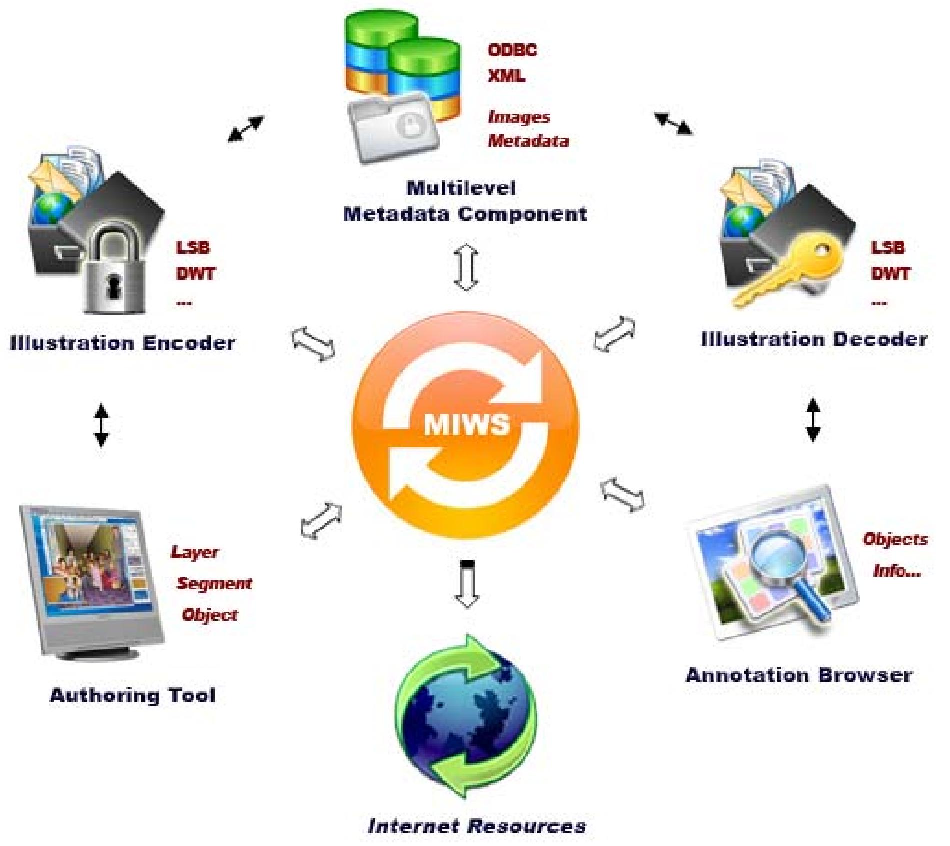


Figure 5.4 Overall architecture for Multilevel Illustration Watermarking System.

Based on the fundamental illustration watermarking model presented in chapter 2 (figure 2.3), I would like to outline a general architecture for Multilevel Illustration Watermarking System (MIWS), as shown in figure 5.4. The system can be known as a network with different application components. The *Cascading*

Previewers that we repeatedly mentioned in this chapter is the example of *Annotation Browser*, in which the user can hone in and explore the metadata information of the watermarked images. The *Illustration Decoder* is implemented as the assistant tool for extracting and retrieving the metadata that supply the previewers. And all data files are developed and managed through the *Multilevel Metadata Component* (MMC). Beside, the *Illustration Encoder* for the task of embedding the data and the *Authoring Tool* for objects and data files management are the rest two main components contributing to the complete system. They do link together with their own working functions to the kernel accessing component of MIWS. For further development, the whole system can also be connected to Internet through a self-defined interface for the task of communicating, publishing or retrieving (other) data resources.

The architecture for MIWS has closed my thesis work with a wide view and open tasks for future developments. Promising work with a complex integration of many fields of Information System, Computer Security, Software Development and Internet Programming is suggested, but Interactive Graphics is still the main subject in consideration.

Chapter 6

Conclusion

6.1 Summary

The goal of this thesis was to develop a tool which can be used to explore images which were augmented with metadata using illustration watermarking techniques. The basic task was to design a small widget which moves with the mouse cursor that shows the watermarked object and its annotation information whenever the cursor passes the watermarked region. The region in the augmented image is an individual segment and it can be highlighted and focused. A fisheye lens technique was used for such focusing, and markers and region contours were employed for the identification of the objects. As the watermarked object is identified, its annotation data is retrieved and displayed on the Metadata Previewer (MP) widget.

Based on the basic theories of distortion views for complex information spaces presentation, the EPS model developed by Sheelagh Carpendale was chosen for the generation of the fisheye lenses. That model was selected since it was a general framework for the creation and manipulation of the lens shapes for the distorted image which adapts well the requirements of the MP interface. The contribution of my work in this task was the adjusted algorithm for the distortion views concerning the expected size of the previewer. Besides emphasizing by distortion, the contour of the watermarked region which was focused was highlighted to indicate its extent. In addition, region markers were included on the previewer as the directives for the navigating process of the user when he or she explores the augmented images. Among some different techniques suggested for the specification of the markers, the central moment of the region's skeleton was the best method recommended in my work. For highlighting the object regions, the contours with contrast colors was proposed and implemented using the simplified region growing algorithm.

To represent the metadata associated with the watermarked region, the abstract visual information for those metadata was supposed to be extracted and displayed. Icons and short texts were suggested for the abstract data displayed on the *abstract metadata bar*, and quantitative visualization in form of chart bar was employed for the presentation of the searching data in the *intersection bar*. Exploiting the theories of GUI design in HCI, the central work in this task was the analysis and discussions for the interface templates that were considered to be advanced than the conventional ways in presenting complex information spaces.

The demonstration MP tool was implemented in C++ using Trolltech Qt® toolkit and QuaZIP library for ZIP archive accessing. It was compiled and linked with MinGW - the CGU C/C++ compiler for MS Windows®. With the analyzed concepts and the demonstrated results, the further discussions and evaluations to the work had also been performed, and the open directions for the future work were then presented.

6.2 Future Work

Despite of the mentioned achievements, there are still considerable amounts of work to be done to make complete the MP tool. In this section, a short summary to the future works is given following their presentation in the previous chapter:

- Integrating the previewer as a navigation tool to a complete illustration watermarking system, for instance the Smage or the IWM.
- Developing or employing the standard structures for the application data used in the system.
- Improving the visualization of the MP with advanced GUI and HCI techniques, for instance using methods of temporal continuity or cursor assistance for the navigation of the MP.
- Developing a more general system for multilevel metadata exploration, following the overall architecture of MIWS, where Cascading Previewers is the proposal for the annotation browser.

References

[1] A. Hanjalic, G.C. Langelaar, P.M.B. van Roosmalen, J. Biemond, R.L. Lagendijk. *Image and Video Databases: Restoration, Watermarking and Retrieval*. Elsevier Science Amsterdam (NL), 2000.

[2] Andries van Dam, "*Post-WIMP User Interfaces*," In *Communication of the ACM*, Vol. 40, Issue 2, pp.63-67, ACM Press, NY, USA, 1997.

[3] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 2nd Edition. Addison-Wesley, Reading, MA, 1992.

[4] Cart Gutwin, "Improving Focus Targeting in Interactive Fisheye Views," In *Proceedings of the ACM Conference on Human Factors in Computing Systems, CHI 2002*, pp. 267-274, Apr. 2002.

[5] Cart Gutwin, and Chris Fedak, "Interacting with Big Interfaces on Small Screens: a Comparison of Fisheye, Zoom, and Panning Techniques," In *Proceedings of Graphics Interface*, Canadian HCI Society, pp. 145-152, May 2004.

[6] Christine Strothotte and Thomas Strothotte. *Seeing Between the Pixels: Pictures in Interactive Systems*. Springer-Verlag Berlin, 1997.

[7] David Salomon. *Data Privacy and Security*. Springer-Verlag New York, Inc. 2003.

[8] Deborah A. Mitta, "A Fisheye Presentation Strategy: Aircraft Maintenance Data", In *Proceedings of the IFIP TC13 Third Interational Conference on HCI*, pp. 785-880, 1990.

[9] D. Schaffer, Z. Zuo, L. Bartram, J. Dill, S. Dubs, S. Greenberg, and M. Roseman, "Comparing Fisheye and Full-Zoom Techniques for Navigation of Hierarchically Clustered Networks," In *Proceedings of Graphics Interface '93*, pp. 87-96, Toronto, Canada, May 1993.

[10] Dinh Quyen Nguyen, Thomas Vogel, "Illustration Watermarking Toolkit," <http://hatchetfish.sourceforge.net/>, 2006.

[11] Edward R. Tufte. *The Visual Display of Quantitative Information*, 2nd Edition. Graphics Press, Cheshire, Connecticut, 2001.

[12] Emmanuel G. Noik, “A Space of Presentation Emphasis Techniques for Visualizing Graphs,” In *Proceedings of Graphics Interface ’94*, pp. 225-234, Canadian Information Processing Society, 1994.

[13] Fabien A. P. Petitcolas, “The Information Hiding Homepage Digital Watermarking & Steganography”, <http://www.petitcolas.net/fabien/steganography/>, 2006.

[14] George W. Furnas, “Generalized Fisheye Views,” In *Proceedings of HCI’86*, pp. 16-23, ACM SIGCHI, Apr. 1986.

[15] H. Sonnet, T. Isenberg, J. Dittmann, T. Strothotte, “Illustration Watermarks for Vector Graphics,” In *11th Pacific Conference on Computer Graphics and Applications (PG'03)*, pp. 73-82, Canmore, Canada, Oct. 2003.

[16] İsmail Avcibaş, *Image Quality Statistics and Their Use in Steganalysis and Compression*, PhD Thesis, Boğaziçi University, Turkey, 2001.

[17] J. G. Hollands, T. T. Carey, M. L. Matthews, and C. A. McCann, “Presenting a Graphical Network: A Comparison of Performance Using Fisheye and Scrolling Views”, In *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*, pp. 313-320, MA, USA, 1989.

[18] Jeremy Reimer, “A History of the GUI,” <http://arstechnica.com/articles/paedia/gui.ars>, Jun. 2006.

[19] J. D. Mackinlay, G. G. Robertson, and S. K. Card, “The Perspective Wall: Detail and Context Smoothly Intergrated,” In *Proceedings of CHI’91*, ACM, New York, pp. 173-179, 1991.

[20] Kevin Mullet and Darrell Sano. *Designing Visual Interfaces: Communication Oriented Techniques*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[21] L. Bartram, A. Ho, J. Dill, and F. Henigman, “The Continuous Zoom: A Constrained Fisheye Technique for Viewing and Navigating Large Information Space,” In *Proceedings of ACM Symposium on User Interface Software and Technology*, pp. 207-215, 1995.

[22] Lothar Schlesier and Henry Sonnet, “Smage: Smart Images,” <http://www.smage.de/>, 2006.

[23] M. Sarkar and M. H. Brown, “Graphical Fisheye Views of Graphs,” In *Proceedings of CHI’92*, *Human Factors in Computing Systems*, pp. 83-91, 1992.

[24] M. S. T. Carpendale, *A Framework for Elastic Presentation Space*, PhD thesis, Simon Fraser University, Canada, 1999.

[25] Marcin Wichary, “Icons Components,” in *Graphical User Interface Gallery*, <http://www.guidebookgallery.org/icons/components/>, 2006.

[26] Martin Kutter, “Digital Watermarking World,” <http://www.watermarkingworld.org/>, 2006.

[27] Mehmet Kivanç Mihçak, *Information Hiding Codes and Their Application to Images and Audio*, PhD Thesis, University of Illinois at Urbana-Champaign, USA, 2002.

[28] N. Kadmon and E. Shlomi, “A Polyfocal Projection for Statistical Surfaces,” In *Cartographic Journal*, 15(1): pp. 36-41, 1978.

[29] Neil F. Johnson, “Steganography & Digital Watermarking,” <http://www.jjtc.com/Steganography/>, 2006.

[30] P. Moulin and R. Koetter, “Data-Hiding Codes,” (tutorial paper), *Proceedings IEEE*, Vol. 93, No. 12, pp. 2083-2127, Dec. 2005.

[31] P. Moulin and R. Koetter, “Data Hiding - Theory and Algorithms,” *Lecture Notes*, Institute for Mathematical Sciences, Singapore, Dec. 2003.

[32] Robert Spence and Mark Apperley, “Data Base Navigation: An Office Environment for the Professional,” In *Behaviour and Information Technology*, 1(1) pp. 43–54, 1982.

[33] Stefan Katzenbeisser and Fabien A. P. Petitcolas. *Information Hiding: Techniques for Steganography and Digital Watermarking*. Artech House, Norwood, MA, 2000.

[34] T. Götzelmann, K. Ali, K. Hartmann, and Th. Strothotte, “Form Follows Function: Aesthetic Interactive Labels,” In *Computational Aesthetics in Graphics, Visualization and Imaging 2005*, pp. 193-200, Girona, Spain, May. 2005.

[35] Thomas Strothotte. *Computational Visualization: Graphics, Abstraction, and Interactivity*. Springer-Verlag Berlin Heidelberg, 1998.

[36] Thomas Vogel, “Illustration Watermarking: A Novel concept for Dissemination of E-Learning,” In *4th IASTED International Conference on Web-based Education*, pp. 371 – 375, Grindelwald, Switzerland, Feb. 2005.

[37] T. Vogel, D. Q. Nguyen, J. Dittmann, “BlobContours: Adapting Blobworld for Supervised Color- and Texture-Based Image Segmentation,” In *Proc. SPIE - Multimedia Content Analysis, Management, and Retrieval 2006*, Vol. 6073, pp. 158-169, San Jose, CA (USA), Jan. 2006.

[38] T. Vogel, J. Dittmann, “Illustration Watermarking: An Object Based Approach for Digital Images,” In *Proc. SPIE - Security, Steganography, and Watermarking of Multimedia Contents VII*, Vol. 5681, pp. 578-589, San Jose, CA (USA), Mar. 2005.

[39] Yanmei Fang, Ning Bi, Huang, D., Jiwu Huang, “The M-band Wavelets in Image Watermarking,” *IEEE International Conference on Image Processing 2005 (ICIP 2005)*, Vol. 1, pp. I - 245-248, Genova, Italy, Sept. 2005.

[40] Y. K. Leung and M. D. Apperley, “A Review and Taxonomy of Distortion-Oriented Presentation Techniques,” In *ACM Transactions on Computer-Human Interaction (TOCHI)*, Vol. 1, No. 2, pp. 126-160, Jun. 1994.

Authorship Declaration

I, Nguyen Dinh Quyen, hereby declare and confirm that this thesis is entirely the result of my own work except where otherwise indicated.

I have acknowledged on page *vii* the supervision and the guidance that I have received.

Magdeburg, July 28, 2006

Nguyen Dinh Quyen