A Content-Aware Design Approach to Multiscale Navigation

Cyprien Pindat

École Doctorale d'Informatique de l'Université Paris-Sud
Laboratoire de recherche en Informatique

THÈSE DE DOCTORAT

discipline : Informatique

soutenue le 20 Décembre 2013

par

Cyprien Pindat

A Content-Aware Design Approach to Multiscale Navigation

THÈSE DIRIGÉE PAR :

Pr. PUECH Claude	*Université Paris-Sud XI & INRIA Chile*
Dr. PIETRIGA Emmanuel	*INRIA & INRIA Chile*
Dr. CHAPUIS Olivier	*LRI & CNRS*

RAPPORTEURS :

Pr. CASIEZ Gery	*Université Lille 1, France*
Dr. PARIS Sylvain	*Adobe Research, Cambridge*

EXAMINATEURS :

Pr. FROIDEVAUX Christine	*Université Paris-Sud XI*
Dr. BOUBEKEUR Tamy	*Telecom ParisTech, France*
Dr. FEKETE Jean-Daniel	*INRIA*

To ... oT

Remerciement

Une thèse, trois ans, c'est long. C'est long parce que trois ans c'est court pour satisfaire l'exigence de nos aînés. Ces trois années de thèse auront donc été pour moi, à la fois les plus longues, les plus excitantes et les plus épanouissantes de mon cursus universitaire. Et ce marathon, j'en dois la réussite à de nombreux soutiens sur lesquels j'ai pu compter. Je consacre ces quelques lignes afin de leur rendre hommage et de leur adresser mes plus chaleureux remerciements.

L'Interaction Homme-Machine est partout. Au coeur de l'innovation, elle est omniprésente dans nos téléphones, ordinateurs et autres machineries digitales, et pourtant, paradoxalement, elle a été étrangement absente des amphithéâtres que j'ai fréquentés durant ma formation. Ma découverte de l'IHM je la dois à Frank Nielsen, Jean-Daniel Fekete, Eric Lecolinet et Yves Guiard sans qui je n'aurais probablement jamais entrepris cette thèse.

Je dois ma thèse à Claude Puech. Un grand merci pour avoir diriger ma thèse, pour y avoir apporté la 3ème dimension, et également pour m'avoir fait traverser l'océan pour rejoindre les terres du *ceviche*, du *Carmenere* et des montagnes. Merci à Emmanuel Pietriga pour m'avoir guider tout au long de ma thèse, mais aussi pour l'ascension des sommets surplombants Santiago. Je jette sur ma thèse le même regard heureux et fier que je portais sur ces montagnes que nous venions de grimper pour enfin embrasser la vue d'une Cordillère des Andes majestueuse et imposante. Alors oui, la Leonera nous aura résisté, mais je reviendrai et je la *pietriguerai*. Merci à Olivier Chapuis pour ton soutient inconditionnel, ton encadrement de contact, et pour les discussions épistémologiques sur le grand n'importe quoi et la toute beauté des statistiques.

Merci à Wendy Mackay et Michel Beaudouin-Lafon pour m'avoir accueilli au seins de votre équipe et d'avoir partager votre passion pour la recherche et l'IHM. Merci aux membres de mon jury Tamy Boubekeur, Christine Froidevaux, Sylvain Paris et Gery Casiez pour le temps et le travail qu'ils ont consacré à la relecture et l'évaluation de mes travaux. Merci également aux intermittents de la recherche : post-doc, doctorant, stagiaires et autres ingénieurs, dont j'ai eu la chance de croiser la route et de partager quelques cafés. Mention spéciale à Jeremy Garcia, David Bonnet et Clément Pillias qui ont été là depuis le début.

Un grand merci à mes amis pour jamais, mais vraiment jamais n'avoir refusé de claquer le carton autour de quelques pintes dans des rades miteux de notre chère capitale. Merci les amis, merci la famille. Un petit merci de rien du tout à Manuella (parce que toi il te fallait bien un merci particulier), sans qui ... bah je sais pas trop en fait.

Abstract

Computer screens are very small compared to the size of large information spaces that arise in many domains. The visualization of such datasets requires multiscale navigation capabilities, enabling users to switch between zoomed-in detailed views and zoomed-out contextual views of the data. Designing interfaces that allow users to quickly identify objects of interest, get detailed views of those objects, relate them and put them in a broader spatial context, raise challenging issues. Multi-scale interfaces have been the focus of much research effort over the last twenty years.

There are several design approaches to address multiscale navigation issues. In this thesis, we review and categorize these approaches according to their level of content awareness. We identify two main approaches : content-driven, which optimizes interfaces for navigation in specific content ; and content-agnostic, that applies to any type of data. We introduce the content-aware design approach, which dynamically adapts the interface to the content. The latter design approach can be used to design multiscale navigation techniques both in 2D or 3D spaces. We introduce Arealens and Pathlens, two content-aware fisheye lenses that dynamically adapt their shape to the underlying content to better preserve the visual aspect of objects of interest. We describe the techniques and their implementation, and report on a controlled experiment that evaluates the usability of Arealens compared to regular fisheye lenses, showing clear performance improvements with the new technique for a multiscale visual search task. We introduce a new distortion-oriented presentation library enabling the design of fisheye lenses featuring several foci of arbitrary shapes. Then, we introduce Gimlens, a multi-view detail-in-context visualization technique that enables users to navigate complex 3D models by drilling holes into their outer layers to reveal objects that are buried into the scene. Gimlens adapts to the geometry of objects of interest so as to better manage visual occlusion problems, selection mechanism and coordination of lenses.

Keywords : graphical user interface, multi-scale visualization, focus+context, detail-in-context, content-aware, lenses, design approach, presentation library

Résumé

Les écrans d'ordinateurs sont de très petite taille comparés à celles des jeux de donnés dans de nombreux domaines. Pour pallier au problème de visualisation de grandes quantités de données, les interfaces de navigation multi-échelles rendent possible l'exploration interactive des données, en facilitant la transition entre vues zoomées et dé-zoomées afin de permettre à l'utilisateur d'examiner les informations en détail ou de pouvoir les interpréter dans un contexte plus global. L'étude de ces interfaces est un domaine important de la recherche en Interaction Homme-Machine, et de nombreuses techniques de navigation ont été proposées lors des vingt dernières années.

Nous proposons une nouvelle approche de conception pour les interfaces de navigation multi-échelles dîtes *content-aware*. Cette approche est basée sur l'adaptation dynamique d'éléments de l'interface au contenu de la scène que l'utilisateur est en train de visualiser, permettant ainsi de proposer des représentations plus pertinentes. Nous présentons trois nouvelles techniques de navigation basées sur cette approche de conception, qui montrent comment appliquer celle-ci pour traiter différents problèmes de navigation, aussi bien en 2D qu'en 3D. Nous présentons dans un premier temps Arealens et Pathlens, deux lentilles de grossissement 2D dont la forme va s'adapter à la géométrie des objets d'intérêts afin de proposer une meilleur intégration de la vue zoomée dans son contexte environnant. Une expérience de laboratoire contrôlée de Arealens met en évidence un gain de performance de ce type de lentille par rapport aux lentilles de grossissement classiques pour une tâche de recherche visuelle. Nous introduisons ensuite Gimlens, une lentille de grossissement 3D permettant l'exploration de modèles complexes. Gimlens permet d'identifier rapidement les objets d'intérêt de la scène, d'afficher des vues détaillées de ces objets, de parcourir des orbites autour de ceux-ci et de les mettre en perspective dans leur contexte environnant. L'utilisateur peut également combiner les lentilles pour afficher simultanément différentes vues complémentaires de la scène.

Mots clés : interface graphique, visualisation multi-échelles, focus+context, détail-in-context, sensible au contenu, lentille, méthode de conception, librairie de présentation,

Table des matières

Table des figures

Introduction

The digital universe keeps expanding faster, between 2001 and 2006 its size increased by a factor of 10 and it is foreseen to keep expanding by a factor of 10 every 5 years [53]. Computer screens are our windows to the digital universe, but yet they remain very small compared to the size of information spaces we need to understand. Large datasets arise in many domains. Geographical Information Systems produce world maps at street level, such as OpenStreetMap [120] that would require $18 \cdot 10^{15}$ pixels to rasterize. Astronomical space telescope Spitzer took in 2008 an infrared picture of the inner part of our Galaxy, made of thousands of frames stitched together to produce a 4.7 billion pixel image. Artists stitch together several thousands of pictures taken by conventional SLR cameras to create gigapixels images, such as the 272 gigapixels panorama of Shanghai [121]. Various industries involve in their processes capture, generation or real-time analysis of huge amounts of data. Financial firms, retailers or content providers generate huge visualizations to understand trends, find patterns and decide business strategy. Manufacturing industries involve CAD modeling throughout the production pipeline, resulting in large 3D models.

Gaining insight into such datasets, seeking for a particular piece of information, or only browsing with no particular purposes in mind but exploration requires multiscale navigation capabilities. Shneiderman developed a general mantra to guide the design of interfaces supporting visual information seeking : "Overview first, zoom and filter, then details-on-demand" [108]. In fact, in some situations, users are very likely to zoom-in first and reach for details to see *what it looks like* then zoom-out to understand *where they were* and then zoom-in again to change location and compare against another part. However the core idea is that multiscale navigation systems, to support the exploration of large information spaces, must allow users to view the dataset at multiple scales.

Designing multiscale navigation techniques typically involves one of three main

interface schemes or combination thereof [35] : overview+detail, which uses spatial separation between focus and context views, zooming which uses temporal separation, and focus+context which blends seamlessly the focus view into its surrounding context. Typical implementations of the latter are magnification lenses and their derivatives.

Magnification lenses usually feature a large, possibly full screen view of the data at low scale, providing users with a global overview. Users can instantiate lenses to get detailed views of the datasets through a movable display region, that they can freely move onto the context view. Using lenses to successively magnify regions allows for showing details surrounded by their context and thus reduce visual discontinuity between the views. A feature that reduces mechanical and cognitive overhead for users shifting their attention successively between details and context, as reported by Grudin [55].

Multiscale Navigation Issues

Designing magnification lenses that allow users to successfully navigate large information spaces involves the three following challenges, that we try to address in this thesis.

Detail in Context Integration

Plumlee and Ware identified *linking* [94] as one of the main challenges for designing magnification lenses. It refers to the cognitive process of understanding the relationship between the focus and the context views. For users to fully benefit from two views at different scales, they must understand the zoom ratio, translation and tilt involved in the magnification. Many techniques emerged in the literature for linking multiscale 2D views such as distortion that integrates smoothly the focus view into the context view ; view proxy ; line tethers [134] ; or other visual artifacts that help linking.

How to integrate magnified regions into a context view has been the focus of a lot of research effort. See [35] for a review of techniques and comparative experimental evaluations of them. Though many techniques improve navigation performance, many issues remain to be addressed.

Distortion was introduced to provide seamless focus-in-context integration. While it guarantees visual continuity, it also causes problems of interpretation[32, 28], focus targeting [3, 56, 93] and virtual navigation. Indeed, most techniques are based on lens shapes defined statically by distance functions obtained through

L(P)-metrics [29]. They often fail to provide relevant magnifications of object(s) of interest whose distorted representation prevents users from recognizing them.

View proxies, directional indicators and tethers were introduced to integrate a focus view into a context view. While they support linking in sparse scenes – Plumlee implemented them for navigation in ocean data [95], which are sparse and flat – they fail to ease linking when the object under focus is invisible in the context view. Which is frequent when navigating complex 3D scenes that feature numerous parts clustered closely together, generating a lot of visual occlusion. Such environments require new techniques for linking focus and context views.

Interaction

As suggested by Mackinlay [82], navigating large information spaces requires more than static magnification lenses. Interfaces must let users interactively readjust the view parameters. Indeed, identification of regions of interests, getting details of these regions, relating them, and putting them in a broader spatial context, requires tedious reconfiguration of the lens viewport. Introduced by Ware and Lewis, the **DragMag** [134] provides two strategies of interaction to control position and zoom level of the magnified view. Either users interact "through" the lens, which thus behaves as a zooming window : dragging adjusts the position of lens' viewport and a zoom slider controls the scale of the magnification ; or users interact "through" the context view directly pointing at regions of interest to adjust the lens' focus accordingly. Combining two such strategies is actually a benefit of magnification lenses. Focus interaction allows to inspect the surroundings of a region of interest, and context interaction allows to quickly switch from one region to another.

Highly occluded environments caused by a high density of numerous overlapping objects clustered closely together, and possibly including one into each other are notoriously difficult challenges to address for implementing such interaction models for complex 3D scenes.

Combining multiple lenses

General exploration often involves pattern matching when trying to understand the general structure of a dataset. Plumlee and Ware showed that users could significantly benefit from multi-view interfaces for comparison tasks, as those help decrease visual memory load [96]. Another advantage of magnification lenses is that they allow for multiple lenses on the same display, letting users inspect several parts of the datasets in parallel, and thus favor comparison tasks. Multi-view

interfaces are of special interest as well for large display environments such as the one described in [89], or CAVEs [41], where multiple collocated users could collaborate navigating to different parts of the information space. However multi-view interfaces can be cumbersome, and generate overhead for users as they increase complexity of the interface [131]. The design of such techniques must provide the right mechanisms to combine, coordinate and manipulate multiple lenses to support navigation.

Approach of the Thesis

We identify two categories of design approaches for multiscale navigation. The first, that we term *content-agnostic* design approach, only depends on the dimensionality of the representation (1D, 3D or 3D) and provides generic navigation techniques suited for any type of data. Pan&Zoom, fisheye lenses or standard overview+detail (see [36]) are examples of techniques that follow this approach. The second, that we term *content-driven* design approach, assumes some geometrical characteristics of the representation (for instance a layout, or particular shapes of the information objects) specific to a dataset, to provides navigation techniques that help users display relevant views of the datasets. The Perspective Wall [82] and the Document Lens [99] are notable examples of this design approach.

In this thesis we adopt a third design approach : the *content-aware* design approach. It dynamically adapts the navigation technique to the representation being visualized to improve navigation. The approach is conceptually situated between the two aforementioned approaches, while the *content-driven* design approach relies on global characteristics of the representation to provide static optimization of the navigation technique, the *content-aware* approach relies on local characteristics to locally optimize navigation, which tends to result in navigation techniques that apply to a larger range of data and representation.

Contribution of this Thesis

The main contributions of this thesis are :

1. **We introduce a new approach for the design of multiscale navigation.** We propose a new design approach enabling the design of navigation techniques that dynamically adapt depending on the content of the scene, allowing local optimization of the presentation or interaction mechanisms.

2. **We present two adaptive distortion-based magnification techniques.** Distortion allows to seamlessly integrate focus into context. However it hinders visualization as objects of interest can be distorted. We introduce two techniques that provide distortion-based magnification lenses whose shapes will adapt so as to preserve objects' visual appearance as much as possible.
3. **We propose a library that enables the interactive rendering of distortion-oriented presentations with possibly multiple foci of arbitrary shapes that adapt dynamically to the object they magnify.** The framework was designed to support parallel rendering on the GPU to achieve interactive rendering of complex lenses. The framework furthermore introduces fine controls over the distribution of the distortion, allowing high expressiveness and rendering of a wide range of lenses. Implementation of the framework allowed us to implement fully functional prototypes of the two aforementioned adaptive distortion-based magnification techniques.
4. **We introduce magnification lenses for navigation in complex 3D models.** The technique introduces *cone-cut*, a new content-aware 3D occlusion management technique that supports 3D linking in dense scenes and that supports *drilling* as well, which allows quick selection of objects of interest in dense scenes. The technique supports parallel exploration as well by coupling and handling conflicts between several lenses. The technique was successfully implemented and tested against several types of 3D models.

Outline of the Dissertation

The dissertation is organized as follows :

In the first chapter we will give a general introduction to multiscale navigation and present content-aware approaches for the design of multiscale navigation techniques. This chapter will serve two purposes : introduce terminology used throughout the dissertation, and better situate our work in the wider scope of visualization. In the second chapter we will present Pathlens, a fisheye lens whose shape adapts to the content of the scene to optimize distortion. We will highlight how content-aware design approaches allow to improve over state of the art fisheye lenses. In the third chapter we will present Arealens,a fisheye lens featuring multiple foci enabling users to navigate in object fields by expanding objects of interest piecewise. Again, content-awareness allows for better occlusion management and more relevant magnification standard techniques. In the

fourth chapter we will present the overall framework used to build both Arealens and Pathlens techniques, which is an image warping technique allowing to render distortion-oriented presentations that feature multiple arbitrary-shaped lenses. In the fifth chapter we will present Gimlenses which are used to navigate complex 3D scenes that couple an automatic 3D occlusion management technique that reveals current object of interest in the scene with a navigation technique allowing users to drill down into the scene reaching for objects buried possibly deep into the model. In the sixth chapter we will discuss the general content-aware approach, its implications for design, present a set of principles drawn from our experience implementing those techniques . We will also raise issues to be addressed in future work to improve integration in end-user systems.

Chapitre 1

Multiscale Navigation in Large Datasets

This chapter provides a high-level introduction to multiscale navigation and situates it in the more general context of interactive visualization. Concepts used for the rest of this Dissertation will be introduced and we will present the approach of this thesis for addressing issues related to multiscale navigation.

1.1 What is Multiscale Navigation ?

Multiscale navigation enables users to explore large information spaces. Helping users make sense of data is the very purpose of visualization. However, the growing size of datasets leads to the inability to convey all the information with a single static illustration. The interface is then tied to display only a subset of the data and requires multiscale navigation capabilities to let viewers interactively reconfigure what portion of the representation is being examined. We define multiscale navigation as navigation within very large information spaces where users need to view datasets at different magnification, or scales.

Navigation is a concept that originates from the physical world. It was then ported to electronic worlds, when monitor screens allowing to visually explore virtual information stored into computers' hard-drives, were introduced. Some concepts from navigating the physical world, such as landmarks and routes were therefore evaluated, and sometimes applied, in these new spaces [42]. Although, navigation in virtual worlds is ruled by completely different laws than navigation in physical world, Spence noticed that the questions asked are identical in both worlds : *Where am I ? Where can I go ? How do I get there ? What lies*

beyond ? [111]. In virtual worlds, locomotion, object visibility and shapes are not ruled by the laws of physics and can be altered for the sake of navigation and visibility. If this new freedom provides new opportunities to solve navigation issues[67], Spence acknowledged that, however, they " undoubtedly posed challenges for the interaction designer."[111]

We will give a concrete example of multiscale navigation taken from an every day life scenario, we will give also more examples of large datasets we are struggling to navigate in the remainder of this section.

1.1.1 What is Visualization ?

We define visualization as the use of computer-supported, interactive, visual representation of data to support its understanding. Pictures are powerful vectors for information, it has been a mechanism of communication long before the formalization of writing. A single picture can convey a lot more information, is quicker to interpret compared to a equivalent page of words. Among our perceptual senses, we rely heavily on our sight for everyday tasks and our visual processing pipeline is tremendously efficient [133]. This is partly due to the fact that human beings process images in parallel and can handle lots of information, as opposed to hearing which is processed sequentially, word after word, which leaves us unable to follow two conversations at the same time. Visual representations hence offer a much higher bandwidth for communication than other representations (sounds, smells etc...) and is a good choice for conveying great amounts of information.

The goal of visualization is to convey information from datasets to the users through all types of communication technology. On the one side we have users with their efficient pattern recognition capabilities coupled with their expectations on the data and decision making capabilities. On the other side we have computers with information stored on hard drives, computing power and display technology. Visualization lies at the interface between the two worlds, trying to take the best of all the resources available to help users in their understanding of datasets.

The design of visualization interfaces is challenging and builds upon visual perception, cognition, and requires a good understanding of the various technologies available for implementing interactive programs.

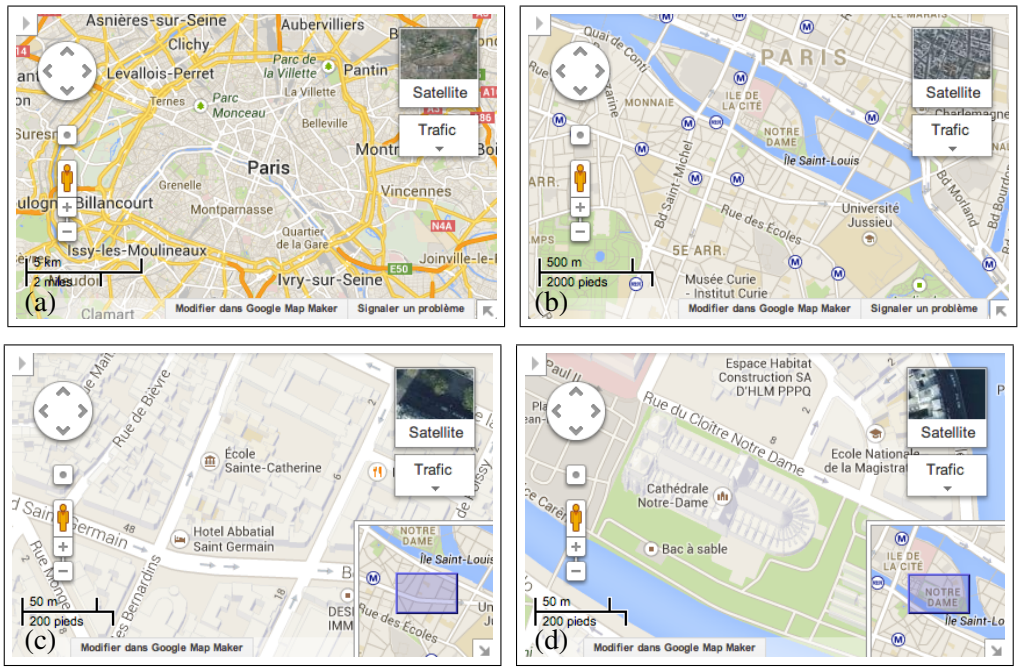


FIGURE 1.1: Illustration of the Context Loss Problem that arise during navigation with the Google Maps interface. A user is searching for *Notre Dame de Paris*. (a) large overview allows to spot the *Île Saint-Louis*. (b) zooming shows more detailed representations. (c) however zooming too much shows less contextual information and users are likely to get lost. A small overview provides more context and reveals that the target is across the river. (d) *Notre Dame de Paris*.

1.1.2 An Everyday Life Example

We use multiscale navigation interfaces on a day to day basis. Geographic Information Systems (GIS) providing representations of the whole world, from a rough overview down to every single streets, are an example of such multiscale datasets whose understanding requires multiscale navigation capabilities. Google Maps, Microsoft Bing Maps and their community-based open source equivalent OpenStreetMap, provide web platforms giving access to such datasets far too big to fit on a screen – rasterization of the OpenStreetMap would require at the highest level of detail an 18 peta-pixels bitmap.

Since their introduction in the beginning of the early 21^{th} Century, web mapping services gained tremendous popularity and they soon incorporated other types of representations, such as satellite views, or the well-known Google *Street View* allowing users to walk through streets of a city thousands miles away. Users rapidly took the habit to use such services.

To illustrate challenges awaiting designers of interfaces for navigating such datasets, we consider the simple following scenario. A person from the north of

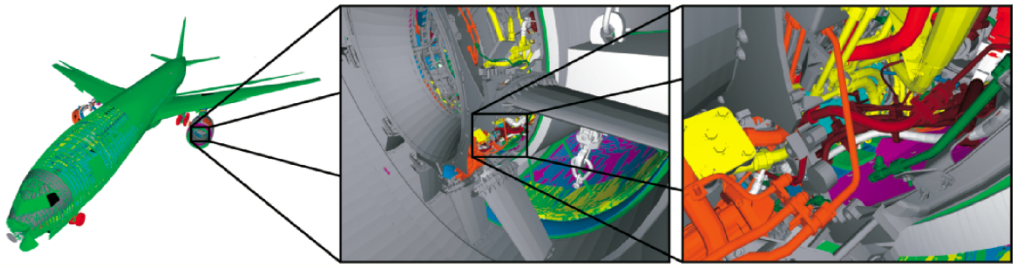


FIGURE 1.2: The Boeing 777 airliner was modeled in detail. The resulting dataset – which includes countless cables, screws, pipes, valves, nuts and bolts – is organized into more than 13,000 files. The raw Polygon representation, made of 350 Millions of triangles, is almost 12 Gbytes. Such datasets are notoriously difficult to navigate and require multiscale capabilities to let users drill into the model to view parts buried deep into the model. Taken from [44].

France travels south to visit the cathedral *Notre Dame de Paris*. On the way she must travel through *Amiens* which stands between *Lille* and *Paris*. Crossing *Amiens* is challenging because or the many turns needed in small streets, and requires a detailed map, but she must not lose sight of the overall purpose of the journey, which is to reach Paris and its Cathedral. For this reason the driver is usually to be found switching between two paper based maps, one with detail representation of the streets and the other one providing an overview.

An interface supporting navigation in GIS must provide access to both overview and detailed views of the map. Figure 1.1 presents the Google Maps interface. The multiple scales of the dataset are accessible by zooming the view (1.1 (a) and (b)). But as users keep zooming, less geographical cues (the river, subway stations, streets names etc...) necessary to know where we are into the data, are visible to users who eventually get lost in the vicinity of *Notre Dame* (Figure 1.1 (c)). This problem is known as the *Context loss* problem.

Google Maps provides a feature to address the *Context loos* problem : clicking in the arrow at the bottom right corner pops-up a small window that shows an overview of the data, bringing back some context information, and allowing users to locate their final destination and adjust their position accordingly. This interface scheme has been identified as one of the main multiscale interface scheme and was termed *Overview+Detail*[35].

1.1.3 Motivations

The amount of data produced worldwide keeps increasing at an ever faster rate [53]. Acquired or generated, both are stored in huge databases and advances in many

domains depend on our ability to make sense of all those datasets.

Many scientific disciplines depend heavily on observations of the world. Building scientific knowledge heavily depends on what we are able to observe. Improvements to our ability to capture the world surrounding us leads to larger and larger datasets. Recently, the Spitzer Space Telescope took a 4.7 billion-pixels infra-red picture showing the inner part of our Galaxy. It allows astrophysicists to observe unprecedented pictures of our galaxy. Visualization of these images is of main importance for confirmation of theories or building new ones and scientists need tools to inspect from a large overview of the image down to each pixels.

Many manufacturing companies such as the aerospace, automotive or electronic industries, rely heavily on modeling of their product throughout the production chain. They allow to simulate behaviors of the system, explore different design at lower cost. The Boeing 777 airliner was entirely modeled using CAD software, resulting in a complex 3D model (Figure 1.2) featuring more than 350 millions polygons [44]. Evolution of technology often moves toward smaller electronic components assembled in very dense environment to match industry standards resulting in more complicated datasets.

The goal of this Thesis is to design interfaces that better help users to make sense of large datasets. This thesis introduces a new design approach to design multiscale navigation, we then apply this approach to design several techniques addressing several issues of multiscale navigation. The techniques are demonstrated on various datasets that we selected to cover a wide range of data types. We use maps taken from OpenStreetMap or Google Maps, two web mapping services or satellite views of the earth. We use diagrams and WIMP interfaces representations. We use 3D CAD models such as a car engine or a plant model.

On the following Sections, we will first introduce a simple visualization pipeline to better understand where does fit such interface into a standard visualization platform. Then we will present the main interface schemes that support navigation in large datasets. Finally we will introduce the original approach of this Thesis along with two others design approaches that we identified from reviewing the literature related to multiscale navigation.

1.2 The Multiscale Navigation Process

Here we briefly introduce a high-level multiscale navigation pipeline. We present all the stages involved in the design of a software aiming at the exploration of

large information spaces. The first stage is to find a mapping between the data and a graphical representation. The second stage is to display the visual representation on the screen. The third and last stage is designing the interaction that let users select what subset of the information space they want to display.

1.2.1 Representation

Carpendale defines the *representation* as "the act of creating an image that corresponds to the information" [29]. This visual representation of the data is meant to convey relevant information to be displayed on a visual display. The image can be of several types : pixel-based or vector-based for 2D visualization, voxel-based (volumetric data) or meshes for 3D visualization. Representation starts with an analysis of the data available to extract information needed by users, and then map data to visual signs to convey information graphically. There are many ways in which data can be represented : color, line thickness, filling pattern and spatial arrangement (layout) among others. Readers should refer to the following readings [133, 54, 132, 110] for in-depth presentation of the field. In the scope of this thesis we rather focus on the type of data to be displayed.

Ware [133] suggests that there are two fundamental forms of data : entities and relationships. Entities are the objects we wish to visualize and relationships define structures and patterns that relate entities to one another.

Entities

Entities are the objects we want to visualize. There can be many kinds of entities. Often, visualization techniques are classified according to the type of entities they were designed to display. In the case of a geographical map, entities could be roads, motor ways, monuments, gas stations, water pond or any feature that a map can display. Users, depending on their need, might be interested by a subset of them only. Hence, a good visualization should let users choose what entities to display. For instance many GPS system let users decide or not to display *Points of Interest* while driving or not, depending on whether they are looking for a particular POI (a gas station for instance) or if they want to reduce visual clutter on the screen while they are driving.

Data Structure

The data structure is the set of relationships that exist between the different entities of the dataset. There can be many kinds of relationships as in "a key is part-of a keyboard" or "two soccer players belong to the same team". Sometimes the

structure is provided explicitly as in the visualization of a company's organizational chart. The visualization aims at conveying the structural information. Some other times, it can be a mean to support navigation. For instance, while browsing the internet, users can bookmark pages and group them into folders. They create a structure that allows them to access items of the dataset faster. Sometimes it is the very purpose of the visualization to find the structure. For instance while browsing pictures taken from a digital camera, users can tag photos to distinguish those to be kept and those to be discarded.

Attributes

Both entities and relationships can have attributes. Their purpose is to describe the entity or the relationship. They should not be considered entities on their own. Sometimes it is not that straightforward to distinguish entities from attributes. An attribute can be of many types and dimensions. Coordinates of a multidimensional attribute are syntactically linked together as they are all defining the position of an element in the information space.

Image

Images also originate from digital capture of the world : SLR cameras take RGB pictures, Kinect add a depth component to RGB pixels, 3D scanners build 3D meshes from real objects and medical MRI build volumetric data form leaving bodies. Such captured images are fundamentally different from representations generated from abstract data, in that entities and structures are not readily available. Representations are built to highlight some structure of the data, captured images need to be processed first to access structural information.

1.2.2 Presentation

As defined by Carpendale in [29], *presentation* "is the act of displaying [the *representation*], emphasizing and organizing areas of interest". Two main classes of presentation techniques exist. The first are the zooming presentation techniques which feature a full window representation of the data. Such interfaces usually let users navigate by adjusting what regions of the representation to display and at what zoom level. Such presentation techniques are prone to context loss problem when users zoom too much into the representation.

The other main class of presentation techniques are the multi-foci techniques, which combine into the interface several regions of the representation, possibly at various scales. Under this class fall many techniques such as traditional

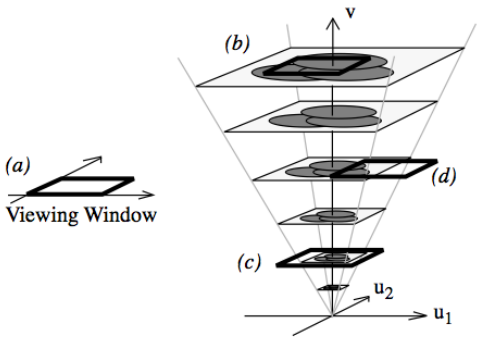


FIGURE 1.3: The space-scale diagram is a representation allowing visualization and analysis of the role of scale in multiscale navigation (From [52]). The interface's viewing window (bolt square) : (a) is shifted rigidly around the 3D diagram to obtain all possible pan/zoom views of the original 2D surface, for example ; (b) a zoomed-in view of the circle overlap ; (c) a zoomed-out view including the entire original picture ; and (d) a shifted view of a part of the picture.

overview+detail (illustrated in Figure 1.1), the DragMag technique [134], and fisheye lenses and their derivatives. The goal of such techniques is to provide users with both detail and contextual representations of the dataset at the same time so as to avoid getting lost in the representation. However, to fully benefit from such techniques, users need to understand the spatial relationship between detail and context views. This cognitive process termed linking (or 3D linking when dealing with 3D data [95]) can be very demanding. The various visual cues introduced in the interface to ease linking are referred to as the integration.

1.2.3 Interaction

As suggested by Mackinlay [82], navigating large information spaces requires more than a static presentation of the representation. Interfaces must let users interactively readjust the view parameters and select what regions of the representation to display.

Many navigation controls such as zooming or panning depend on selecting objects, specifying a direction or positioning a target within the representation. For example in some interfaces, zooming may require users to point in the representation to specify the zoom center, while other interfaces let users select a set of objects that the interface will fit on the screen by adjusting pan and zoom parameters. Therefore how is the representation mapped onto the visual display impact significantly such navigation techniques.

Other presentation techniques such as the standard overview+detail feature widgets to support linking, that users can interact with to navigate. For example, the

view proxy of the Google Map interface (in blue in Figure 1.1(c) and (d)) can be grabbed and drag, allowing users to move the detailed representation's focus point.

Those features are based on the principles of direct manipulation [109] as they allow users to control parameters of the view by directly interacting with objects of the representation or elements of the presentation technique. Interaction and presentation are intimately linked.

1.3 Multiscale Interfaces

In this section we present the main interface schemes introduced in the literature to support multiscale navigation. Cockburn [35] introduced a classification featuring four main strategies to blend detailed and contextual views. The four interface schemes are : zooming interfaces which use a temporal separation ; overview+detail which uses a spatial separation ; focus+context which blends smoothly the detailed view into the contextual view and cue-based techniques which provide navigational cues by adding visual artifacts to the scene.

1.3.1 Zooming

The first basic category of interfaces supporting both focused and contextual view is based on zooming which involves temporal separation. Zooming techniques let users get a detailed representation by displaying a smaller region of the representation into the application window, leaving more space to show more details.

Many applications feature a full-zooming interface and let users select interactively what subset of the representation to display. They present a wide range of input mappings to control pan and zoom, combining conventional input devices such as mouse and keyboard, or providing control via WIMP interfaces or multi-touch interaction. We give a more comprehensive review of the topic in Section 1.4.1.

The Pad system [90] introduced by Perlin and Fox, was the first fully zoomable desktop environment. They consider the desktop as an infinite two dimensional information plane in which data can be browse by successively zooming into subset of that space. This work introduces two fundamental concepts : semantic zooming which allows different representations of the information for different scales, and portals, which allow links between data objects and filters on their representation. To support research on zoomable interfaces, several toolkits where

introduced to ease development of such interfaces including Pad++ [14], Piccolo [13] or ZVTM [91],

Introduced by Furnas and Bederson space-scale diagram integrate representations of the dataset at several scales into a visualization allowing to view pan and zoom navigation as spatial movement into a dataset, see Figure 1.3. This framework provides a general approach to understand view navigation in multiscale datasets.

However, full zooming techniques suffer from context-loss problems. Detail-in-context interpretation required relying on short term visual memory, which is cognitively demanding and prone to loss of context. To address this issue other interface schemes were introduced such as overview+detail and focus+context.

1.3.2 Overview+Detail

The overview+detail techniques provide both a detailed view and an overview of the information space simultaneously, each in distinct presentation spaces.

The standard overview+detail interface, as Cockburn names it [35], is the one implemented by Google Maps, shown in Figure 1.1, which introduces a small overview of the map on the bottom right corner of the window, providing users with more contextual information. This technique has been widely implemented in video games, more particularly strategy games or sports games where a small representation of the entire map helps users situate their current location, or desktop applications such as image-editing tools.

Lenses

The aforementioned standard overview+detail technique arranges detail and context views by juxtaposing them, meaning that views are laid out next to each other. Other interfaces propose to separate them on the z axis, allowing to stack views one on top of each other.

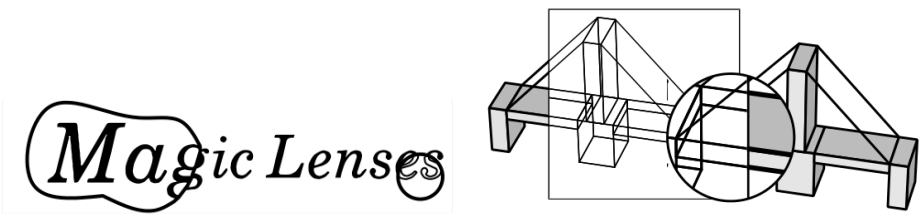


FIGURE 1.4: Magic Lenses allow to magnify objects however, the representation requires more space and the lens overlaps with the surrounding representation and lead to a loss of surrounding context– taken from [17]

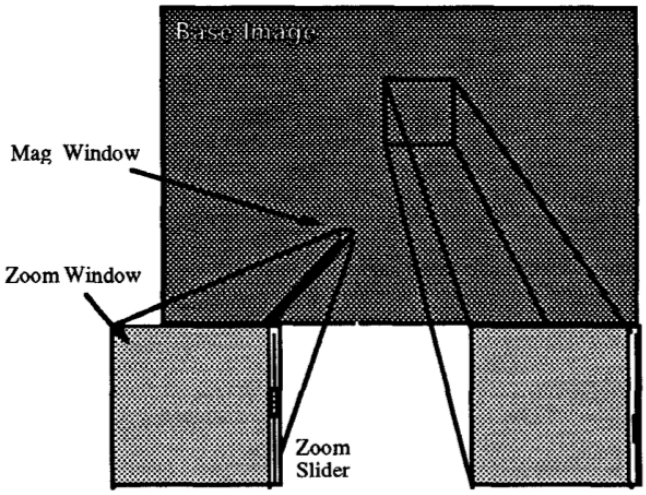


FIGURE 1.5: DragMags shift the magnified representation to a movable lens that user can move to reveal surrounding context– taken from [134].

Bier et al. [17] introduced magic lenses, which are resizable see-through windows. They allow users to transform the visualization of underlying objects, to allow focused viewing of specific attributes. Magic lenses allow to magnify underlying object however, the representation requires more space, and the lens overlaps with the surrounding representation and leads to a loss of surrounding context as illustrated in Figure 1.4.

Introduced by Ware and Lewis, the DragMag[134] interface addresses the surrounding context loss problem by shifting the magnified representation to a movable lens that users can freely move and resize to reveal the surrounding context (see Figure 1.5). View proxy and line tethers link the lens to the location of the magnified information object in the overview to help interpret the presentation. DragMag provides two strategies of interaction to control position and zoom level of the magnified view. Either users interact "through" the lens, which thus behaves as a zooming window : dragging adjusts the position of lens' viewport and a zoom slider controls the scale of the magnification ; or users interact "through" the context view directly pointing at regions of interest to adjust the lens' focus accordingly.

1.3.3 Focus+Context

The third interface scheme, called focus+context, integrates focus and context into a single display, displaying the focus seamlessly within its surrounding context.

Focus+context techniques were inspired by fisheye lenses used by photographers

which would capture the world in a similar manner than how a fish would perceive the outside world from under the water. Governed by Snell's laws, rays of light would be deflected when entering water depending on their slope with the surface. Rays almost normal to the surface won't be deviated resulting in an almost un-distorted magnified image at the center. While rays with a shallow angle are deviated, what results is a compressed picture on the periphery.

This image was then captured by Furnas [51], resulting in a metaphor saying that people need a way to pay attention to particular detail they are focusing on, yet also need some surrounding context. Later, the metaphor was implemented into graphical views that express Furnas' formalism. More screen real-estate is allocated to a magnified representation of the data – the *focus* region – laid out near the point of interest and directly encircled by a less magnified representation of the surrounding information space – the *Context* region. Between the two regions, a distorted presentation of the image achieves a smooth integration of both : the *Transition* region (see Figure 1.6 for an illustration).

Mackinlay *et al.*[82] provided the first implementation of a focus+context technique with the Perspective Wall shown in Figure 1.7. The technique compresses the context using a linear transformation, which achieve the perspective effect, and allows to fit a larger representation into the display space.

1.4 Design Approaches to Multiscale Navigation

Two main trends emerged in the design of multiscale navigation systems. The first is a *content-agnostic* design approach, that only depends on the dimensionality of the representation and provides generic navigation techniques suited for any type of data. Pan&Zoom, fisheye lenses or standard overview+detail (see [36]) are examples of techniques that follow this approach. The second is a *content-driven* design approach, that assumes some geometrical characteristics of the representation (size, shape, volume of the objects or a particular layout) specific to a dataset so as to provide navigation techniques that help users display relevant views of the datasets.

While these two approaches cover most of the techniques introduced in the field, another approach emerged recently : a content-aware design approach. In this approach, the navigation dynamically adapts to the representation being visualized to improve user navigation. The approach is conceptually situated between the two aforementioned approaches. While the *content-driven* design approach relies on global characteristics of the representation to provide static optimization of the navigation technique, the *content-aware* approach relies on local characteris-

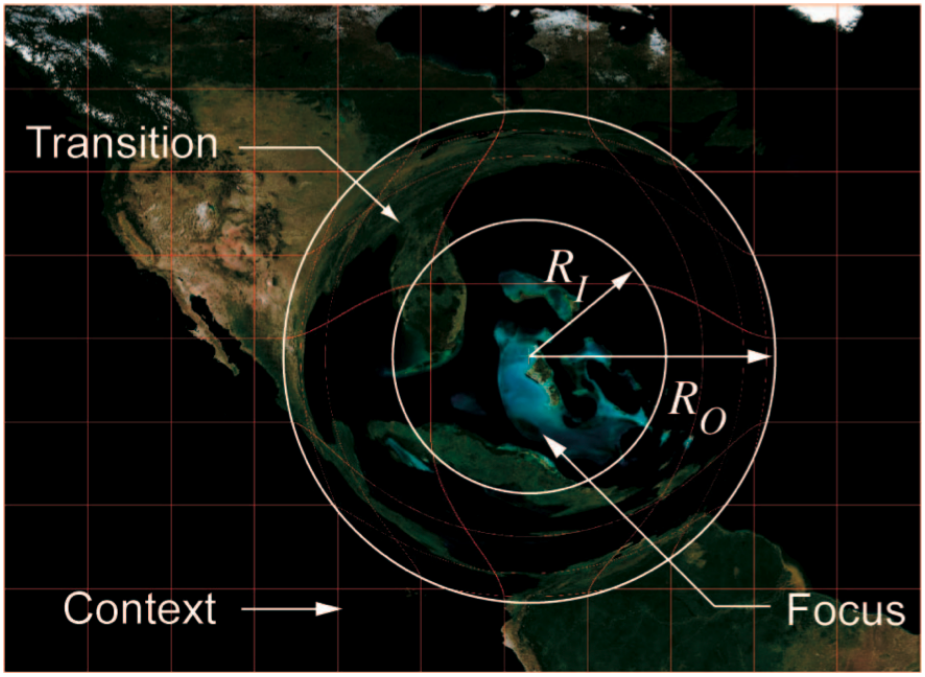


FIGURE 1.6: Graphical Fisheye views are focus+context multiscale interfaces : the *Focus* region provides an in-place magnification of the picture, smoothly integrated by the *Transition* region into the *Context* region

tics to locally optimize navigation. which tends to result in navigation techniques that apply to a larger range of data and representation.

We will introduce successively the three approaches and provide some design examples taken from the literature. We will then discuss potential outputs of the content-aware design approach, challenges involved by such design approach and future work.

1.4.1 Content-Agnostic Design Approach

This design approach provides interfaces suitable for navigation of any data type. It only takes into account the dimensionality of the representation, i.e three dimensional (3D), two-dimensional (2D) or linear (1D) spaces. Such techniques provide controls for adjusting the camera configuration allowing navigation in the targeted space.

3D Spaces

Three dimensional spaces allow for camera movements and rotations about each of the three axes. Control of the camera varies across interfaces and across input modalities provided by the platform.

Most desktop applications will allow camera manipulation via keyboard and mouse input. They let users sequentially operate the camera by either translating along the viewing plane (panning), translating along the viewing direction (zooming) or rotating around the current pivot point (rotating). Usually users are required to first activate the desired mode (usually via WIMP interface components or shortcuts mapped to keys or mouse buttons) and perform dragging gesture on the viewport, which adjust the corresponding camera parameter accordingly. Implementations can be found in popular 3D modeling applications such as Blender [119], Autodesk Maya [118] or Freecad [116] among others. The recently release, Leap-Motion [117] hand-tracking device allows mid-air 3D camera control in a desktop environment.

Other environments such as high-resolution wall-sized displays or immersive environments such as CAVEs provide more exotic input devices : motion-tracking system, gesture enabled trackpad, tablets or mid-air pointing devices just to name a few. They allow users to walk into virtual spaces [34], control camera through head motion or mid-air gestures.

These interfaces are generic, they are suited for navigation of both volume and surface (implicit or explicit) representations of the data.

2D spaces

Two dimensional space is the most common type of virtual space for data visualization. Camera configuration involves positioning the viewpoint relative to a surface. Controls for zooming are commonly provided as toolbar widgets such as slider or push button. For applications that support a zoom mode (such as Adobe Reader), mouse drags are interpreted as zoom controls. Pinching allows to control zoom on multi-touch enabled devices, as well as mouse wheel when available. Controls for panning are, as well, provided via toolbar widgets, and more commonly via drags gesture on the viewport. Other gestures specific to touch-screen such as flicking allow to pan.

Such interfaces are very popular and part of many end-user application. Most PDF viewers, web browsers, word processor, image editing software, all implement pan & zoom technique to navigate content.

More recently, CycloPan & CycloZoom+ [84] allow to control with a single continuous repetitive gesture both pan and zoom. Wall-sized displays let users control pan and zoom with many different devices, in particular motion-tracking allows for mid-air interaction [89].

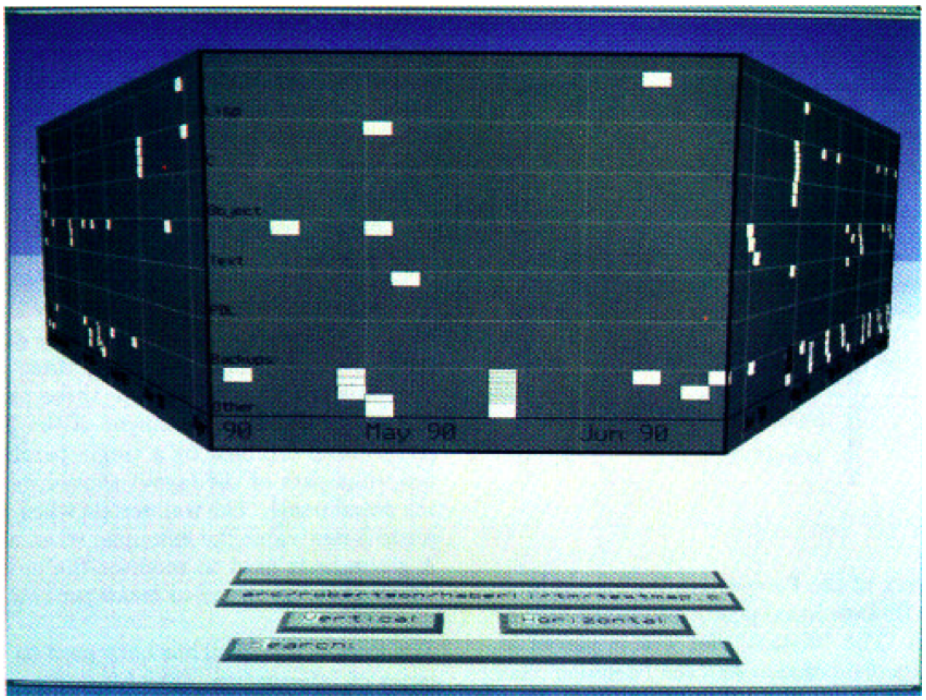


FIGURE 1.7: The Perspective Wall is a distortion oriented, focus+context presentation technique : a rectangular region, providing a zoomed-in view of the data, is surrounded with two panels, showing the data linearly distorted, to provide an overview. The technique is based on the data-driven design approach : it was designed for navigating data featuring a linear structure, here projects' files of an architect are laid out by creation time–Taken from [82].

Some applications extend pan & zoom with the standard overview+detail interface (presented in Section 1.3.2). Geographical map systems, strategy videogames, are among the most representative of such applications.

Fisheye lenses are generic interfaces as well. However despite their versatility, they are seldom adopted because of some usability issues, refer to Section 2.2 for a review and discussion of related work.

Linear spaces

Fewer degrees of freedom are available in linear spaces. The most common implementation only provide controls for position. Scroll bars, slider, lists allow to position the view through drag events. Fisheye lenses implemented in the Mac OS X Dock, provide both zooming and positioning. The OrthoZoom technique [4] allows to control both panning and zooming parameters from a single gesture, for navigating along linear structure.

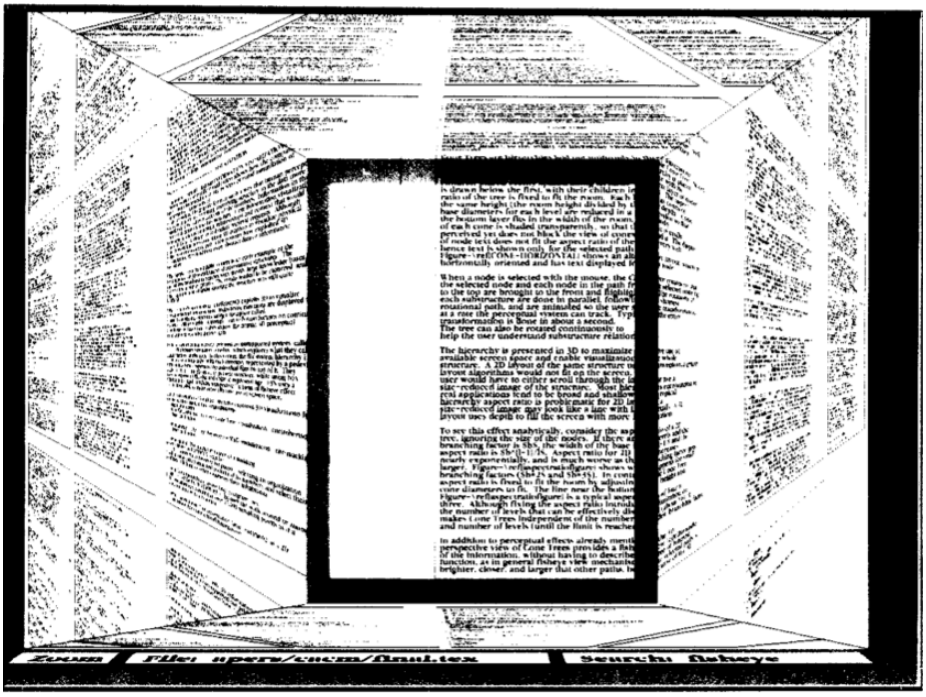


FIGURE 1.8: The Document Lens magnifies a region of the data and distorts linearly the surrounding documents both in x and y axis. This technique, based on the content-driven design approach, was designed for navigating documents featuring a grid layout. Indeed, a linear distortion preserves grids and is thus well suited for showing overview of such documents – taken from [99].

1.4.2 Content-Driven Design Approach

Design of a visualization starts by an analysis of the data and depends on the understanding of what users are expecting from those data. From this analysis will result choices about the way to represent the information, and some structure or entities will be highlighted by mean of visual representation. The content-driven design approach to multiscale navigation takes advantage of visual structure to optimize navigation techniques to help users display relevant views of the dataset.

We will illustrate this process with brief analysis of the design of two navigation techniques taken from the literature.

The first case we consider is the design of the Document-Lens [99] introduced by Robertson *et al*. It was designed to navigate collections of paper documents laid out in rectangular array in a large table. The purpose of the interface was to show both the global context and details of documents as well. While pointing out that fisheye lenses failed to adequately show the context, they drew two design choices from the layout of the documents to match the desired goal and improve

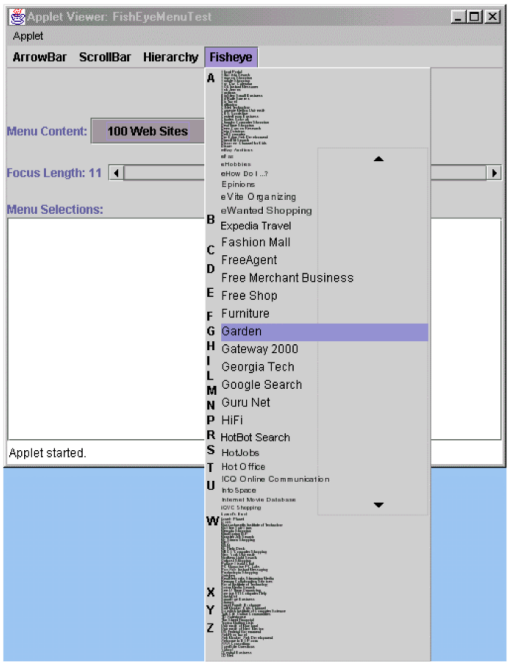


FIGURE 1.9: Fisheye menus allow to navigate large lists of items. The presentation technique takes advantage of the linear organization of the items to provide a smooth integration of magnified items surrounded by less magnified items – taken from [11].

understanding of the global context. First the technique would feature a rectangular lens larger than the width of a line to encompass a whole chunk of text with no need for horizontal scanning. Second, affine transformation would be used to stretch the rest of the document to make it fit into the window. A property of affine transformations is that they preserve lines. Hence the grid structure is still understandable after transformation and provides users with understandable global context : questions such as “How many columns are before or after the document ?” or “How many paragraphs are left before the end of the column ?” can be answered easily from the visualization.

The second case is the design of the Perspective-Wall [82] by Mackinlay *et al.* Studying work processes of an architect, they pointed out that “work practices cause information to have a linear temporal structure” [82] and designed the Perspective-Wall to visualize the number of work documents emitted and observe their distribution over time. They report that the technique successfully allowed to draw insightful discovery of patterns in the architect’s work-flow. The first design step was to identify time as a spanning property which they use to lay out document emission in an horizontal axis accordingly. Using bar height to encode number of emissions. As the resulting representation features a large ratio, they use an affine transformation to make it fit on screen, allowing linear

structure to be viewed as a whole. In particular, linear distortion allows to allocate more compression on the horizontal axis (time axis) and less on the vertical axis (document emission) and allows viewers to see greater time intervals while preserving understanding of document emission evolution. Here, the interpretation of evolution of document emissions takes precedence over interpretation of time for understanding trends or identifying patterns.

This design approach has been implemented in a number of cases reported by literature in the field.

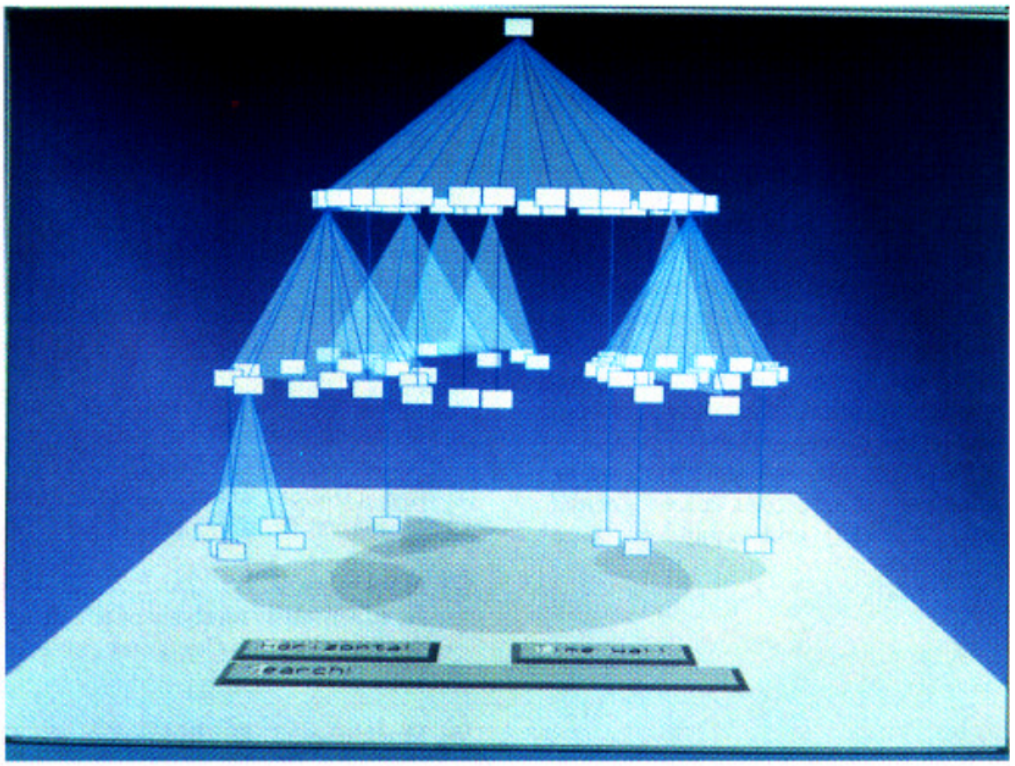


FIGURE 1.10: Cone Trees represent trees by 3D cones. The representation conveys efficiently hierarchical structure by cascading cones – taken from [100].

Cone-Tree [100] represents trees as 3D cones, where each node forms with its children a cone aligned with the viewing plane (see Figure 1.10). Navigation is achieved by rotating the cones resulting in the permutation of the closest child. The technique takes advantage of the 3D cone representation to support navigation. Lamping *et al.* designed a navigation technique that maps huge hierarchies to a unit disk, using hyperbolic geometry [76]. Navigation is achieved by moving a focus point within the hierarchy. This technique relies on radial layout of the hierarchy suitable for mapping to a disk. Schaffer *et al.* introduced variable-zoom[105] technique for navigating, which relies on a hierarchical clustering of

the nodes which allows for dynamic expansion and shrinking of clusters being visited by users. Readers can refer to [59] for a survey of the many techniques introduced to navigate graphs.

Some techniques were introduced to navigate calendars, a particular type of temporal data. Spiral Calendar [83]relies on the cyclic structure of time to represent calendar in a spiral layout. Users select the part of the calendar they want to view, which results in rotation and translation of the spiral to bring the selected part into focus. Again the navigation technique relies on a representation layout drawn from data analysis. Another calendar navigation technique was implemented into a calendar application for PDA. The datelens [12] automatically resizes selected event to show more details.

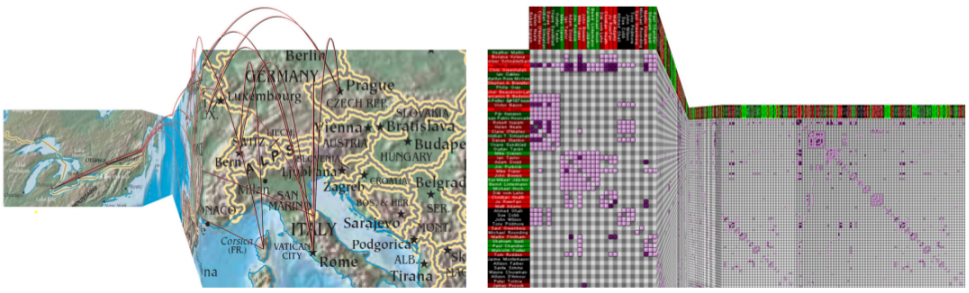


FIGURE 1.11: Melange fits large information spaces by folding the space between several flat focus regions. The affine transformation used to fold the space is particularly well suited for grid structured representations as it preserves lines – taken from [47].

Many techniques to navigate table like structured data relies on the grid layout specific to tables. Melange [47], a technique that fits different focus regions of large information spaces by folding the space between the different focus regions, was shown to be more efficient on adjacency matrices (see Figure 1.11). Melange implements an affine transformation suited for grid structured representations (affine transformation preserve lines, thus grid layout understanding). The table lens is a focus+context technique designed for navigation in large tables [98]. Magnification of a cell is achieved by magnifying its corresponding row an column. Fisheye menus [11] aim at the visualization of large lists of menu items. The design implements a fisheye effect on a per row basis meaning that a magnification factor is assigned to each row to achieve smooth integration of zoomed-in view of focus items into the menu (see Figure 1.9).

1.4.3 Content-Aware Design Approach

The content-aware design approach is based on the concept of extending traditional techniques by taking into account various characteristics of content begin

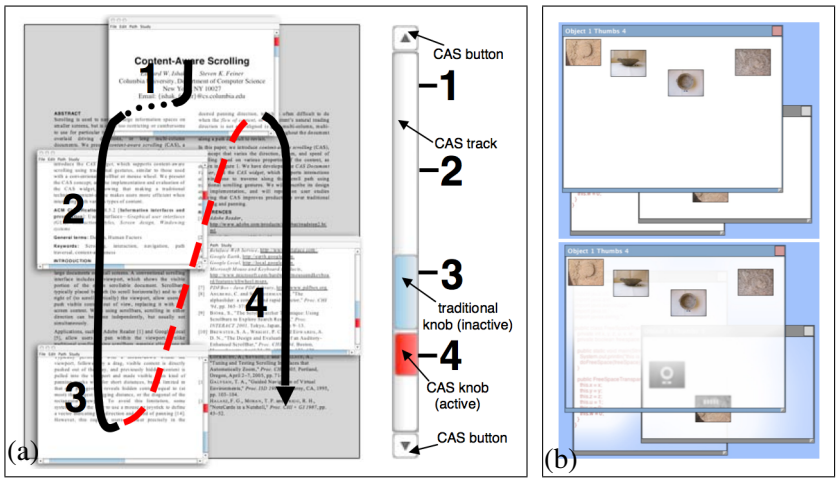


FIGURE 1.12: Two techniques implementing content-aware design approach. (a) content-aware scrolling [64] adapts the scrolling direction to the content of the document, here viewport automatically jumps from the bottom of a column to the top of the next one. (b) a window management technique that adjust window transparency to reveal underlying content [63].

visualized to determine how to view and interact with it.

Under this approach fall several techniques to navigate documents. Content-aware scrolling was introduced by Ishak and Feiner [64]. The technique improves traditional scrolling by adapting the scrolling direction, speed and zoom depending on characteristics of the document. As an example, the technique supports scrolling of multi-column documents by automatically repositioning the viewport at the top of a column when it reaches the bottom of the previous one, as shown in Figure 1.12-(a). Cockburn *et al.* improved scrolling device performance (such as the scrolling wheels, scroll-point joystick [1] or trackpad) by adapting the scroll gain depending on the document length. A controlled experiment showed an increase in performance for both large and small documents.

The content-aware design approach was implemented as well for layout management techniques. Ishak and Feiner introduced a content-aware free-space transparency technique to interact with the otherwise hidden content of obscured regions [63]. The technique adjusts transparency settings of the windows on a per-pixel basis to allow users to see-through and reveal underlying content (see Figure 1.12-(b)). Ishak and Feiner also introduced a content-aware layout technique [65] that automatically adjusts the layout of multiple-window applications depending on the content of each window. An example application for this ap-

1. A scroll point is a joystick used as a scrolling device, typically mounted in a mouse, movement on the stick are echoed in movement of the document on screen.

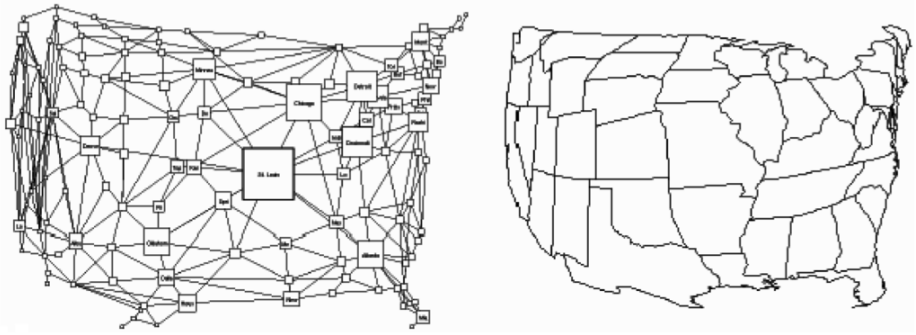


FIGURE 1.13: Sarkar *et al.* introduced graphical Fisheye views of graphs that take advantage of the graph structure, and the nodes sizes to provide magnification of sub-parts of the representation integrated into the less-magnified surrounding context – taken from [102]

proach was to improve word search in a text document. The technique rearranges each window showing the word searched for, so as to align occurrences of the word to reduce eye movement.

Other implementations target trees or graph navigation. Topology-aware navigation, introduced by Moscovich *et al.* [88] allows navigation in large graphs and introduces two techniques. "Link sliding" is a technique for sliding along the links. Once engaged in an edge, panning is constraint to a path that follows the edge, navigation comes down to controlling the traversal progression along the link. Zoom level is adjusted so as to provide the user with additional context while sliding along the link. "Bring & go" brings adjacent nodes close to the node upon selection to help users to better decide what link to follow [88]. Both techniques exploit the connection information provided by the network to support users navigation. Sarkar and Brown designed graphical fisheye views for navigating graphs [102] (illustrated in Figure 1.13). Their techniques achieve magnification of a node smoothly integrated into an undistorted context. The technique relies on node size and graph connectivity to achieve a dynamic layout management that is suited for any type of graph. Blanch and Lecolinet introduced multiscale navigation for navigating treemaps [19]. The technique allows to zoom into hierarchical treemaps, adapting zoom-level to the desired targeted layer.

Finally we found examples of this design approach for improving 3D navigation. Khan *et al.* introduced a new interaction technique for supporting users in close inspection of 3D surfaces. Hovercam [71] keeps the camera at constant distance from the surface while users are panning the view, automatically adjusting tumbling, panning, and zooming of the camera. This technique relies on the mesh

description of the 3D surface. McCrae *et al.* introduced a multiscale 3D navigation technique, that extends a 3D interface with a *look-and-fly* capability to reach remote parts of the scene. Trajectory and flying speed are adjusted depending on size, distance and visual clutter of the scene.

The content-aware design approach was successfully involved in the design of several techniques in a wide range of application. This approach is being applied throughout this thesis to address issues raised by multiscale navigation.

Chapitre 2

Fisheye Lens for Steering Paths

In this chapter, we present Pathlens, a fisheye lens whose regular shapes dynamically adapts to the shape of the underlying objects of interest. Fisheye lenses often cause perception issues by changing the apparent shape of objects falling under the spatially distorted transition region. Pathlenses provide visualizations of higher relevance by optimizing what regions fall under focus, context and transition domains, preserving as much as possible objects of interest from the negative effect of distortion.

In the first section we introduce the problem of mismatch between lens' and objects' shapes that causes misinterpretation of objects of interest. We introduce also an interaction metaphor to address this issue. The second section discusses related work. In the third section, we will present an implementation of the metaphor based on implicit surfaces. The fourth section presents the image warping technique we designed for achieving arbitrary shaped lenses. We will conclude with discussion and future work.

2.1 Introduction

2.1.1 The Shape Mismatch Problem

Fisheye lenses use distortion to guarantee visual continuity between the focus and context regions. However, as reported by Carpendale in [32], distortion causes confusion and disorientation : objects of interest get distorted, which prevents users from recognizing them.

Indeed, most techniques are based on statically defined lens shapes relying on

distance functions obtained through $L(P)$-metrics [29] that often fail to provide relevant magnifications of the object(s) of interest : either the lens is small, preserving context but requiring extensive navigation to explore the region of interest when the latter does not fit in the focus (Figure 2.1-a) ; or the lens is big, showing a larger portion of the region of interest at the expense of the context (Figure 2.1-b). Moreover, distortion impedes comprehension, even more so when the lens' and objects' shapes differ significantly.

We present an interactive focus+context technique called Pathlens, that dynamically adapts to the geometry of object(s) of interest. Pathlenses optimize what regions fall into the focus, context and spatially-distorted transition regions based on user interaction, providing detail-in-context visualizations of higher relevance than existing distortion-oriented magnification lenses (Figure 2.1-c).

2.1.2 The Water Drop Metaphor

Pathlens consists of a lens attached to the mouse cursor, that adapts its shape, circular by default, to the geometrical representation of the nearest objects considered of interest. Intuitively, Pathlenses behave approximately like water drops on a spider net, or more generally speaking like drops on an irregular surface featuring elements of varying affinity [127].

In the following, we show how this metaphor fits the formalism introduced by Furnas in the seminal paper "Generalized Fisheye Views"[51]. This formalism expresses the idea that users need to focus on details but as well need context information. It was later implemented into graphical interfaces leading to the graphical fisheye views and its derivatives.

In its original formulation, this formalism is more general. Here we apply it to the context of pixel-image distortion-based presentation, considering pixels as the core unit of information. Each pixel is assigned a "degree of interest" (DOI) defined as the contribution of two functions :

$$\mathbf{DOI}_{fisheye}(x|. = y) = \mathbf{API}(x) - \mathbf{D}(x, y)$$

where $\mathbf{API}(x)$ is the *a priori* importance and $\mathbf{D}(x, y)$ is the distance to the users' current focus of attention. This formula expresses two ideas : that users are less concerned by what is far away from their center of attention ; and that users are variably interested in the content of a scene, some content have more importance than others for understanding information – for instance the actual content of a webpage is more interesting that an advertisement banner.

FIGURE 2.1: (a) Venice and the Lido. (b) a small fisheye can magnify a portion of the island from the Adriatic sea shore to the Laguna Veneta, but fails to show the entire island, requiring extensive navigation to see it in detail ; (c) a large fisheye magnifies a bigger portion of the island, but at the cost of severe distortion of almost the entire image, hiding other islands ; (d) a Pathlens automatically adapts its shape to the region of interest, magnifying as much relevant information in the focus region as (c) while better preserving the context : surrounding islands are left almost untouched from (b).

Then the design of the fisheye lens comes down to defining an *a priori* interest function, choosing the distance function and defining the proper threshold k which decides what to keep and what to filter-out from the focus region : pixels with a higher "degree of interest" $\mathbf{DOI} > k$ are displayed at full magnification and those with a lower DOI are displayed at smaller scale. Threshold k captures the fact that users can process a limited amount of information at a time and then filter out the surplus, less interesting information. Furnas suggests that systems should support this process by filtering out less relevant information.

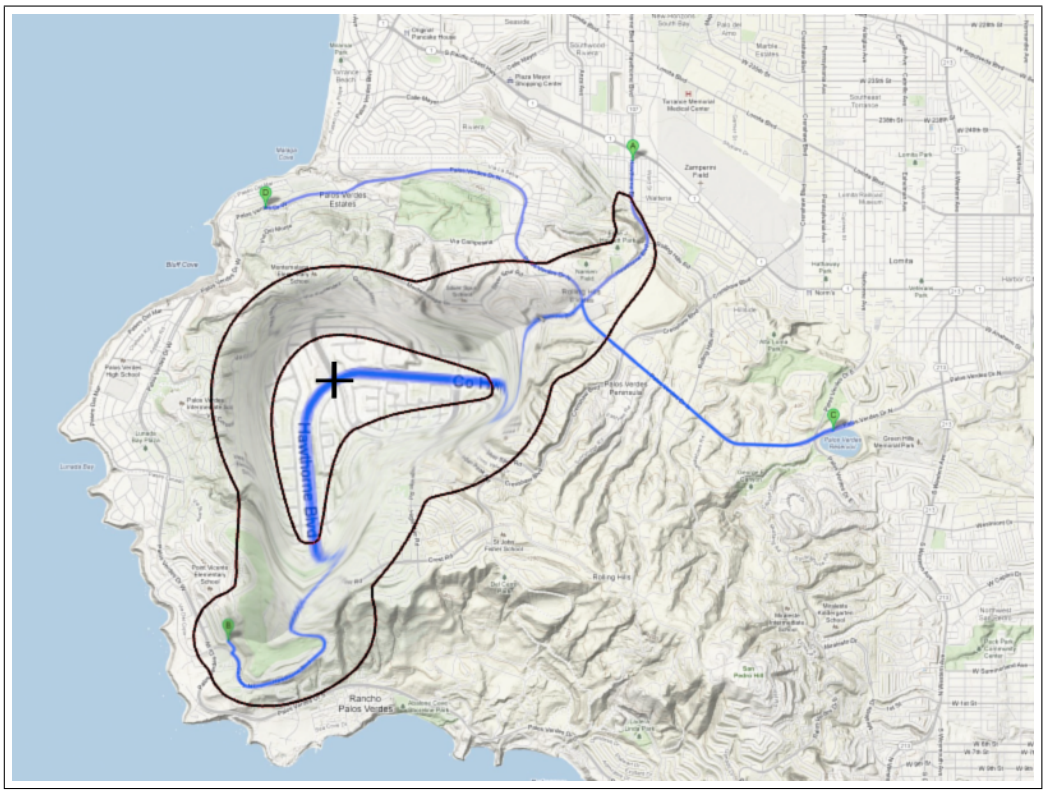


FIGURE 2.2: The three regions defined by a constrained lens : focus region $\mathcal{F}$ magnifying the region of interest, context region $\mathcal{C}$, and smooth transition $\mathcal{T}$ between $\mathcal{F}$ and $\mathcal{C}$ achieved through distortion.

The water drop metaphor ca capture the latter for pictorial representations.

The amount of information users will process corresponds to the focus region's surface area. The focus region will morph, maintaining a constant surface area, to encompass as few pixels of low interest as possible, relegating them to the transition or the context regions. Doing so, it will leave more room in the focus region for encompassing more pixels of higher interest. Which visually behave like a water drop on a spider net.

An important feature of this metaphor is its universal characteristics. Inspired from the physical world, it becomes straightforward for users to understand how it is supposed to behave, and then allows predictive behavior which is always a desirable feature when designing adaptive interfaces.

2.1.3 Overall Process

Adapting a Pathlens to match the geometry of the region of interest is a three-step process. The first step consists of obtaining information about the geometry

FIGURE 2.3: Navigating a subway line with a Pathlens let the users view magnification of both the subway line and the corresponding stations.

of objects of interest in the scene, that the lens will adapt to. This step is heavily dependent on the nature of the graphics visualized : the geometry information is readily available in 2D vector graphics scenes, as well as in 3D scenes, but not in pixel-oriented scenes where the objects of interest are arbitrary sets of contiguous pixels on a bitmap. In the latter case, the geometry information has to be obtained through some external means, such as feature extraction algorithms applied dynamically to the image, or metadata generated through manual annotations packaged with the original image. At this point we simply emphasize that a Pathlens does not necessarily have to adapt to all graphical objects in a scene ; it can ignore those considered as not particularly relevant for some particular task. For instance, the lens in Figure 2.2 only takes roads on the blue itinerary into account, ignoring all other graphical features in the adaptation process.

The second step consists of computing the lens shape according to its position in

the visualization and to the geometry of nearby object(s) of interest (information obtained through the first step). This step, which is the core of the adaptation process, defines how the lens behaves as it gets repositioned by the user in the visualization using the mouse. In Figure 2.2, the lens adapts its shape to match that of the portion of the object of interest (the itinerary) around the mouse cursor. The adaptation method of Pathlens is presented in Section 2.3.

The last step is the rendering of the region seen through the lens and is described in Section 2.4.

2.2 Related Work

Distortion-oriented visualizations often rely on metaphors inspired by the physical world : magnifying glasses [99, 56], stretchable rubber sheets [104] and, more generally, surface deformations [30]. Other techniques work with more fundamental concepts : hyperbolic projection [76], non-linear magnification fields [69] or complex logarithmic views [22].

Early systems made the distortion extend to the boundaries of the representation [76, 77, 99, 102, 104], thus affecting the entire display. More recent techniques use a locally-bounded distortion function, leaving a large part of the context untouched, which reduces the negative impact of distortion [57, 93]. Such lenses, usually termed *constrained lenses*, can be created using 3-dimensional pliable surfaces [30] and the framework for unifying presentation space [29, 28], non-linear magnification fields [69], conformal mapping [139], or the Sigma Lens framework [3, 93].

Magic Lens filters [17] were among the first interface components based on constrained lenses to actually support elaborate, non-regular shapes. Magic Lenses are graphical filters that can modify the appearance of objects seen through them in various ways. Magnification is only one of the many powerful transformations that they make possible. However, to our knowledge, their shape is defined statically (no dynamic content-aware adaptation), a limitation shared with more recent lenses that support irregular shapes but do not adapt their geometry [30, 139], including the recent undistort lenses [23].

Other techniques have been developed for 3D datasets. A first set of techniques deform 3D representations by projecting a texture on a mesh that models the distortion, as do pliable surfaces for 2D representations [30]. LaMar *et al.*'s magnification lenses [75] are based on homogeneous texture coordinates and special geometries. They can be applied to both 2D and 3D representations but are li-

mited in the type of distortion and lens shapes they can model. Non-linear perspective projections [137] project the RGB image produced by a 3D pipeline on a surface shape inserted in front of the flat projection plane. Related to the latter is Brosz et al.'s single camera flexible projection framework [24], which is capable of modeling non-linear projections through the parametric representation of the viewing volume.

There is also an impressive set of space deformation techniques, ranging from early works on the deformation of solid primitives [7, 107] to view-dependent geometry [97] and deformation based on hardware-accelerated displacement mapping [40, 106] and deflectors [73]. These techniques distort 3D geometry, but often do so in an object-centric manner, and are thus not well suited to the implementation of focus+context navigation lenses, which deform a region of the current display, i.e., a subsection of the current viewing frustum that intersects a set of objects, some of them only partially. Camera textures [112] are among the few to actually apply constrained magnification lenses to 3D meshes, but the technique requires a sufficient level of tessellation of the target mesh to produce distortions of good quality. Wang *et al.*'s technique [129] is designed to minimize distortion, but applies to 3D objects only and, relying on a grid-based energy optimization model, is limited to basic shapes between the magnified and compressed regions.

Böttger *et al.* [21] recently introduced a domain-specific warping technique that distorts a city's geographical map to match the layout of subway stations from the corresponding schematic transit map. Distortion is not used to magnify a region of interest, but rather to establish a correspondence between an ordinary map, that is geographically accurate, and a schematic map optimized for the readability of a specific network. While very different from our approach in terms of visual output, interaction, and usage, map warping and Pathlenses have conceptual similarities : they both make use of the geometry of particular objects of interest in the visualization to adapt the distortion.

Finally, Pathlenses are also conceptually related to the numerous content-aware image resizing techniques that have emerged recently, from seminal work on seam carving [5] to Laffont *et al.*'s image zooming technique [74]. However, those are not focus+context interaction techniques, as they do not provide users with explicit control of the magnification (factor, region), and in many cases do not preserve context.

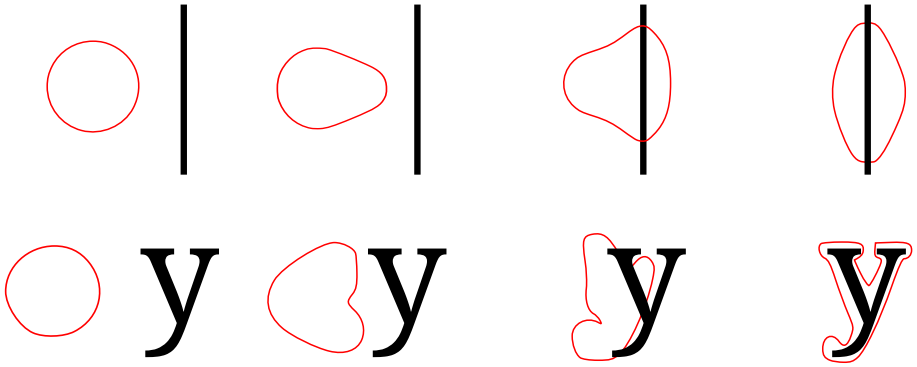


FIGURE 2.4: PathLens morphing effect : the shape of the lens morph into the shape of the objects of interest. We use the morphing technique for identification of both the focus and the context regions.

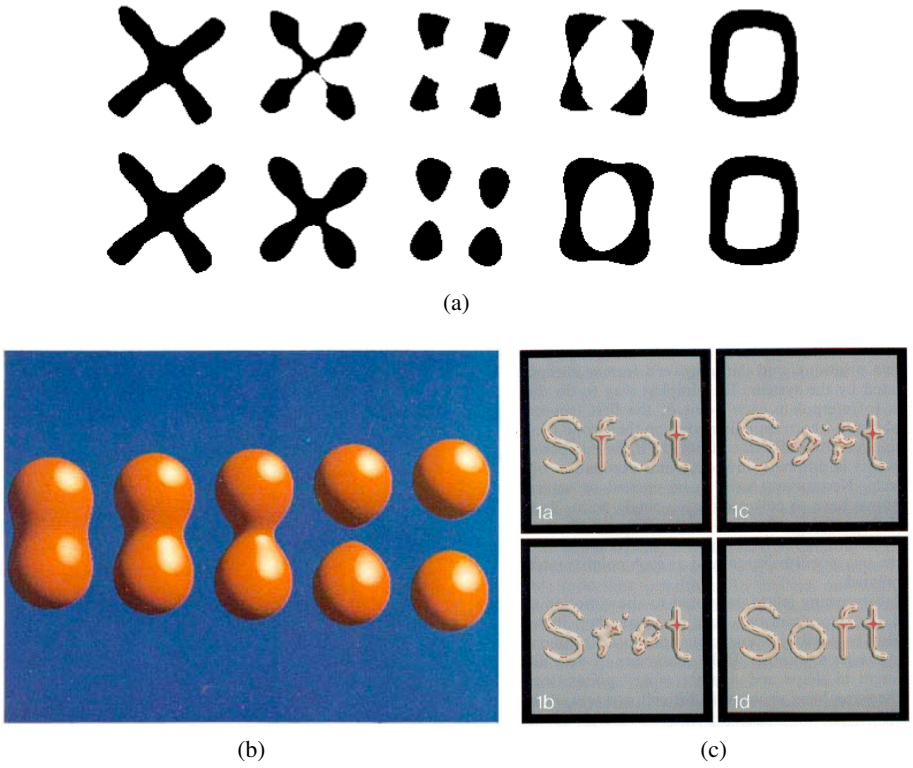
2.3 Instantiating the Metaphor

Design of an adaptive fisheye, whose shapes change to adapt to the content of the scene, is a delicate feature. First, because the animation caused by the adaptation might generate distraction. Changing the shape of the lens while users is inspecting a representation, generate moving patterns that can draw their attention away from their current center of attention. Misused animation might impair performance.

According to the principle of Congruence [115], to lower the distraction impact, animation should conform to actual changes in the content of the scene. In the case of a moving lens, this means that animation should blend to the shape of the changing objects surrounding the moving lens' focus point. Hence, success of the animation depends on its ability to reflect changes that occur in the vicinity of the focus point.

Secondly, adaptation might generate frustration. Adaptation of the interface is made to provide better visualization, however, if users do not understand why is the interface being updated, they become anxious [109], and expect new changes to occur at any time. The adaptation mechanism should be clear to users, so as he can predict changes of the interface and adapt their own behavior accordingly to take full advantage of the technique. The interface must no mislead users, once users have acquired such an understanding, the adaptation must comply to the model.

The water-drop metaphor we introduced, to base the adaptation upon, addresses both these issues. A water drop sliding along spider net, morphs its shape and



(a)

(b) (c)

FIGURE 2.5: Implementation of the Pathlens morphing effect is based on an implicit definition of the shape. Implicit definition of objects were heavily investigated for modeling [114], animation [20] and transformation [135].

reflect changes of the underlying spider thread. And, it provides users with an comprehensible picture of how the interface behave. However, success of the adaptation technique, depends heavily on the fidelity of the implementation to the metaphor.

2.3.1 Water Drop Implicit Modeling

In terms of implementation, we need a technique that, according to geometrical description of objects of interest, the cursor's position, and a desired surface area, generates a shape that morphs to adapt to the geometry of objects of interest. The shape should adapt smoothly to the objects in the vicinity of the cursor, as it is being moved. "Metaballs", "Blobbies", "Soft Objects" [135, 136], "Algebraic Surfaces" [20] or "Implicit Shapes" [114]– all referring to the same concept of implicit definition – are convenient models for modeling water effects.

The definition of surfaces in computer graphics usually falls into two categories : explicit and implicit. An explicit definition would give a straight description of

the surface as a list of vertices and edges. Where an implicit surface is defined as the set of all points which satisfy some equation

$$F(x, y, z) = s$$

where s is the iso-value determining the surface. The technique was first introduced by Blinn who applied it to algebraic surfaces[20]. The Wyvill brothers later introduced Soft-Objects [135, 136] that extend the approach further for modeling and animating objects in 3D space. The term Metaball was later introduced to designate the general approach of implicit definition of 3D models.

In practice, implicit surfaces are used for modeling and animating. Surfaces are then converted into polygonal meshes (explicit definition) for rendering or additional processing. This is usually achieved by extracting the mesh using a marching cube algorithm [81]. The space is partitioned into a grid of cubes whose dimensions depend on the required precision. Then the algorithm travels the cubes, evaluating the function at each vertex and comparing it to the threshold s, a heuristic based on what vertices are greater and lower than the threshold, which allows to approximate the portion of the surface crossing the cube.

We adapted this approach to 2D spaces for implementing the water drop effect. The desired morphing effect is illustrated in Figure 2.4. Many effects achieved by mean of implicit surfaces (some of them exhibited in Figure 2.5) feature smooth blending of blobby entities that recall behavior of water drops. The challenge is to define the functions that model correctly enough the metaphor so as to find the proper threshold. The extraction of the polyline is then achieved by means of a marching square algorithm, an adaptation of the marching cube algorithm to 2D spaces.

2.3.2 Function Definition

Implicit surfaces allow to achieve smooth blending effects as exemplified by the well-known Metaballs demo effect (see 2.5-a) showing two spheres merging together into one unique sphere. The idea behind this effect is to consider each sphere as a radius of influence around each particle. Then, the merging can be considered as the accumulation of both particles' influence. Each particles is assigned a density field that decreases and tends toward 0 as distance to the particle increases. The surface is implied by taking an iso-surface of the accumulation of both density fields. When particles are far away from each other, the surface implied features two spheres whose radius depends on the iso-value – the higher the value, the nearer it will be to the particle. Then the merging effect is achieved

by moving particles towards each other. The resulting density field increases and results in the merging effect.

The key to implicit surfaces is the definition of the density fields to achieve the desired effect. Implementation of the metaphor comes down to the definition of two density fields, one for the lens ($lens(\mathbf{x})$), centered on the users' focus of attention (the cursor position : $\mathbf{c}$) and one for the objects of interest ($data(\mathbf{x})$).

As for Metaballs, the contribution of objects of interest to the field will drop down to 0 as distance increases, relying only on $lens(\mathbf{x})$ for the definition of the shape. While when distance decreases, $data(\mathbf{x})$ increases, perturbing the lens' shape, which eventually results in the desired blending effect. Finally, as distance approaches 0, objects' contribution takes over, fully defining the shape which, as a result, morphs itself to match the shape of those objects. At this point, the lens contribution still influences the shape by constraining its extent.

Lens Contribution

The definition of the lens must feature a circular shape with the focus point at its center, when beyond the radius of influence of objects of interest. This is achieved by making its definition rely only on distance to the cursor. We want to arrange the field to tend toward $-\infty$ with distance approaching 0 and tend toward $+\infty$ when distance increases, as these properties allow to select the shape's surface area from a wide range of values. Large thresholds yield small shapes, and conversely, small thresholds yield large shapes.

Following the aforementioned requirements, we define the lens' contribution field as follows :

$$lens(\mathbf{x}, \mathbf{c}) = \frac{1}{||\mathbf{x} - \mathbf{c}||} - ||\mathbf{x} - \mathbf{c}|| \tag{2.1}$$

where $\mathbf{c}$ is the cursor's position.

Objects Contribution

As mentioned earlier, how to obtain the geometry of objects of interests is heavily context dependent (see Section 2.1.3) : it can be readily available via the accessibility API ; or it can result from feature extraction algorithms ; or it can be authored manually by users. In all cases, it is very likely that the geometry comes as an explicit definition.

Translation from explicit to implicit shapes is a common process in computer graphics. Popular methods usually rely on object distance fields, which can be approximated by various methods : fast sampling on the GPU [60], adaptive

sampling [50] or vanishing-gradient approximation [18]. Their popularity goes beyond pure computer graphics ; they can be used for proximity queries [61], and apply to various fields such as digital design, organic forms, arts and fractals [49, 50].

For our model, we extend the function introduced by Wyvill *et al.* [136] $G(t)$, to arbitrarily shaped objects of interest. It features the following desirable properties. The field is continuous, and it has no influence beyond a certain distance β (its radius of influence). Its maximal value is achieved when distance is null (i.e. within the objects of interest). And it features the following properties, $G'(0) = 0$ and $G'(\beta) = 0$ which are suitable for achieving a smooth blending effect [136]. The contribution is then defined as :

$$data(\mathbf{x}) = G\left(\frac{D(\mathbf{x})}{\beta}\right) \tag{2.2}$$

with $G(\mathbf{x})$ being Wyvill's function :

$$G(r) = \begin{cases} 2r^3 - 3r^2 + 1, & if \ \ r < 1; \\ 0 \ \text{ otherwise} \end{cases} \tag{2.3}$$

and $D(\mathbf{x})$ being the distance between $\mathbf{x}$ and the closest object of interest. We notice that the description of the entire scene relies on a single distance field. As naive implementations of $D(\mathbf{x})$ might be computationally demanding, depending on the number of objects and the precision of their description, developers can rely on one of the aforementioned techniques to sample the distance field. As precision is not critical for modeling the lens, we opted for the regular grid sampling on the GPU described in [60]. An example of such a distance used on a map of the greater Boston area is shown in Figure 2.6-(3)

Again, we make the overall field value the accumulation of the two contributing functions.

$$f(\mathbf{x}, \mathbf{c}) = lens(\mathbf{x}, \mathbf{c}) + data(\mathbf{x}) \tag{2.4}$$

where $\mathbf{c}$ is the cursor's coordinates.

2.3.3 Focus, Context : Balance Control

A lens is defined by two shapes. One for delimiting the focus region and one for delimiting the context region. We rely on the same function for the two shapes, we only change the surface area adjusting the iso-value for each region.

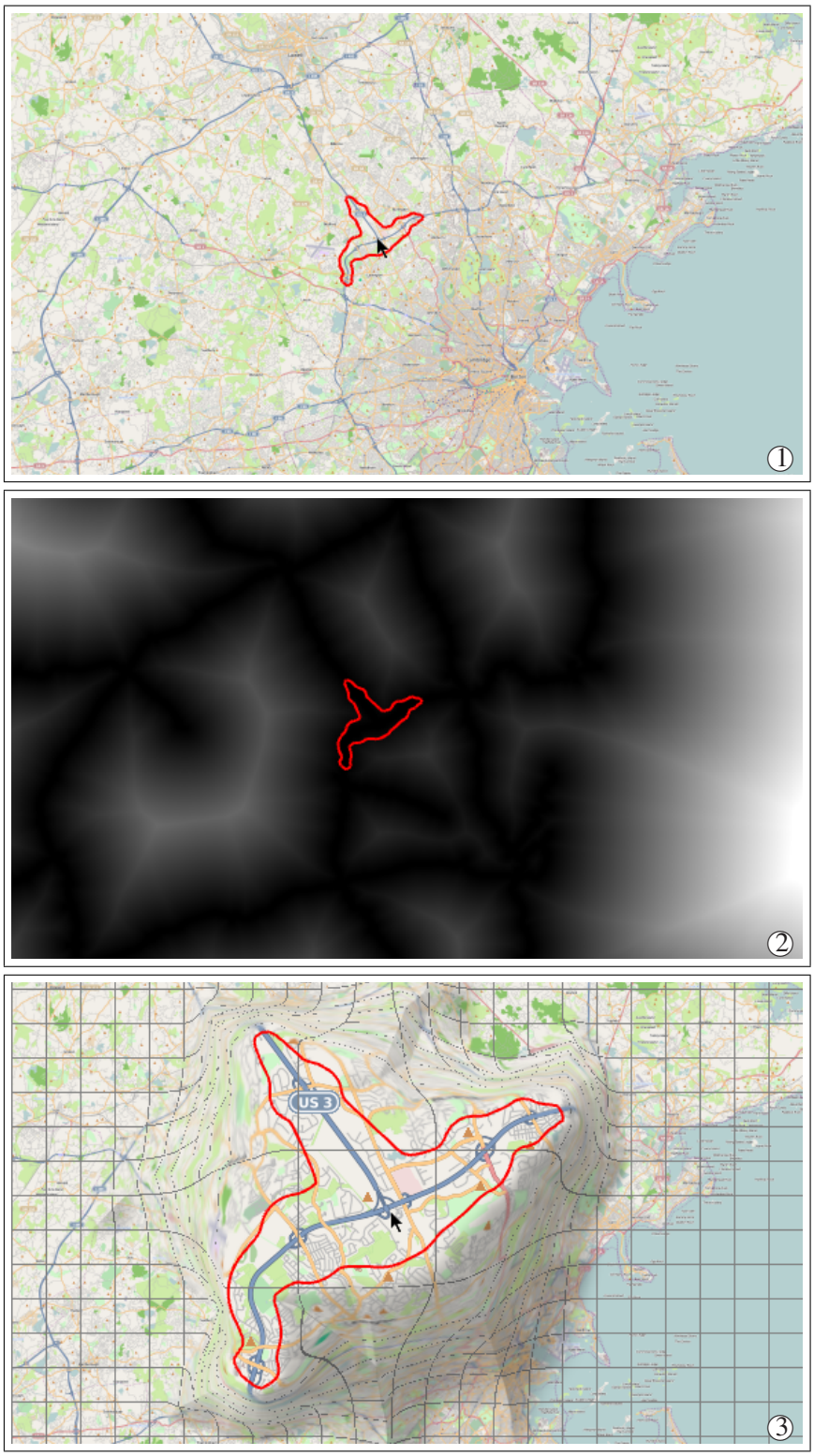


FIGURE 2.6: Illustration of the adaptation and rendering pipeline for PathLens : the region of interest is identified as a function of cursor position (1) and distance fields (2) ; the lens is then adapted to match this shape and rendered with a shading effect (3) – actual rendering displayed to the user, red contour excepted.

As the model introduced does not provide a straightforward control over the surface area, we implemented an energy based optimization method, where we iteratively adjust the iso-value s up to the desired value.

To this end we define the following energy :

$$u(s) = (area(\mathbf{c}, s) - A)^2 \tag{2.5}$$

where A is the targeted area and $area(\mathbf{c}, s)$ is the actual area of the shape implicitly defined by $f(\mathbf{x}, \mathbf{c})$ and the threshold s. To evaluate $area(\mathbf{c}, s)$ at a given threshold s, we evaluate an explicit extraction of the shape by a marching square algorithm. In the following $\Gamma(s)$ defines the shape defined at the iso-level s : $\Gamma(s) = \{P \in \mathbf{R} | f(\mathbf{x}, \mathbf{c}) = s\}$.

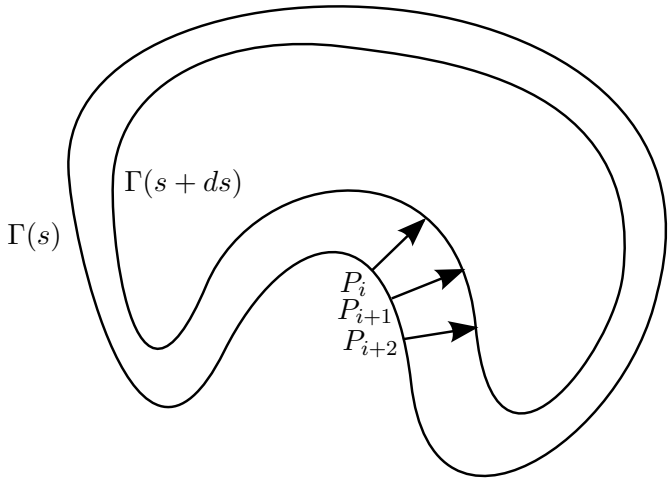


FIGURE 2.7: We use a gradient descent optimisation method to estimate the iso-level satisfying the desired surface area. The gradient require the derivative of the surface area of the shape that we evaluate by estimating the surface area of the region in-between the two shapes $\Gamma(s)$ and $\Gamma(s + ds)$.

We use a gradient descent method for solving this system, we compute the gradient based on a estimation of the variation of the surface area.

$$u'(s) = 2\frac{\partial area(\mathbf{c}, s)}{\partial s} \times (area(\mathbf{c}, s) - A) \tag{2.6}$$

The derivative $\frac{\partial area(\mathbf{c},s)}{\partial s}$ is the surface area of the region in-between $\Gamma(s + ds)$ and $\Gamma(s)$ (see Figure 2.7). To evaluate this area, we define v_i the distance between the two shapes at each point P_i surface elements over the shape. The derivative is then evaluated as :

$$\frac{\partial\ area(\mathbf{c}, s)}{\partial s} \simeq -\frac{\sum_{i=0}^{n} ||P_i - P_{i+1}|| v_i}{ds} \tag{2.7}$$

The level set theorem tells us that the $\vec{\nabla} f(P_i)$ is orthogonal to the shape at P_i. The gradient is then the direction of the shortest path from each point to the other shape. v_i satisfies then :

$$f(P_i + \frac{\vec{\nabla} f(P_i)}{||\vec{\nabla} f(P_i)||} v_i) - f(P_i) = ds$$

which leads to :

$$f(P_i) + v_i \times dot(\vec{\nabla} f(P_i), \frac{\vec{\nabla} f(P_i)}{||\vec{\nabla} f(P_i)||}) - f(P_i) = ds$$

with $dot(.,)$ being the euclidean scalar product,

$$v_i \times ||\vec{\nabla} f(P_i)|| = ds$$

and finally :

$$v_i = \frac{ds}{||\vec{\nabla} f(P)||}$$

Replacing v_i into (2.7) we obtain an evaluation of $\frac{\partial\ area(\mathbf{c},s)}{\partial s}$:

$$\frac{\partial\ area(\mathbf{c}, s)}{\partial s} \simeq -\frac{\sum_{i=0}^{n} ||P_i - P_{i+1}||}{||\vec{\nabla} f(P)||}$$

And finally the formula to evaluate $u'(s)$:

$$u'(s) = -2 \times \sum_{i=0}^{n} \frac{||P_i - P_{i+1}||}{||\vec{\nabla} f(P)||} \times (area(\mathbf{c}, s) - A) \tag{2.8}$$

As stated earlier, a lens is defined by two shapes. The above method is applied twice, with two different targeted surface areas A, to compute both the focus and context regions' contours. Care must be taken to avoid that both contours overlap, bearing in mind that the focus' contour is going to be magnified by a factor of μ. The context contour must be made large enough.

2.3.4 Achieving a Seamless Morphing Effect

The above method achieves the morphing effect. However, while the lens approaches an object of interest, some discontinuities in the resulting shape might happen as the two contributions $lens(\mathbf{x}, \mathbf{c})$ and $data(\mathbf{x})$ merge. To avoid any unpleasant artifact, parameter β , which influences the lens' aspect when approaching an object of interest, must be tuned with great care.

Parameter β defines the radius of influence of the objects of interest. To understand the influence of this parameter, let us first consider the case when β tends towards ∞. The contribution of function $data$ is canceled, resulting in a lens with constant circular shape. When β tends towards 0, the contribution of objects of interest drops off abruptly, which tends to fit the object's geometry more accurately but leads to discontinuity problems. When the cursor is hovering over an object, we want $data$'s contribution to be as in this second case, making the lens shape match the object's geometry more strongly. Conversely, when the cursor gets away from any object, we want to transition to the first case and avoid any discontinuity. This is done by making β dependent on the distance from the cursor position to the object : ranging from a minimum value as distance tends towards 0 and a maximum value as distance increases and reaches a maximum radius of influence.

2.4 Rendering Arbitrary Shape Lens

Constrained lenses are defined by (Figure 2.2) a focus region $\mathcal{F}$, often termed *flat-top* ; a context region $\mathcal{C}$ left untouched by the lens ; a transition region $\mathcal{T}$, sometimes termed *compression region* ; a magnification factor μ ; a focus point **P** that will be the center of magnification. Focus region $\mathcal{F}$ is the region of interest to be magnified. This region is a flat magnification that is not affected by spatial distortion. It is bounded by a non self-intersecting contour (a simple polygon). Context region $\mathcal{C}$ is the region left untransformed by the lens. It is bounded by two contours : another non self-intersecting contour on the inside and the image's boundary rectangle on the outside. Regions $\mathcal{F}$ and $\mathcal{C}$ are disjoint : $\mathcal{F} \cap \mathcal{C} = \emptyset$. Region $\mathcal{T}$ then corresponds to the region in between the two contours. This is the region where distortion occurs, in the form of compression to allocate more screen real-estate to the magnified region of interest.

We use the adaptation algorithm twice to generate both the focus and context shapes. The focus shape is then magnified by μ to define the focus region, requiring the generation of a context shape big enough to encompass the focus shape once magnified.

Common focus+context distortion techniques operate on vector-based representations [104], pixel-based representations [69, 93] or both graphic modes [29]. We implemented a pixel-based distortion technique for two reasons. First this is the most general approach as all representations eventually get rasterized to map screen pixels and eventually allow to adapt the technique to any kind of data. Second, the amount of pixels to displace is bound by the size of the screen while this is not the case for vector-based representation where the size of the representation can increase and impair performance. As an example, an export of the Paris area with OpenStreetMap for a 1440x900 pixel image leads to a 2.4MB PNG pixel file and a 71.5MB SVG vector file (tested on OpenStreetMap's official website [120]).

In the following, point coordinates are **bold**, scalars are in $italics$, and values defined relative to the source image are primed. The transformation to render points in the focus is straightforward :

$$\mathbf{x} = T_F(\mathbf{x}') = \mathbf{P} + \frac{\mathbf{x}' - \mathbf{P}}{\mu} \tag{2.9}$$

The transformation is formulated as a reverse mapping. For each pixel coordinates **x** on screen, i.e., in the destination image, we must find the corresponding point in the source image, $\mathbf{x}'$. Pixels in the context region are mapped to themselves : $\mathbf{x}' = \mathbf{x}, \forall \mathbf{x} \in \mathcal{C}$. The inverse of function T_F is applied to pixels falling into the focus region $\mathbf{x}' = T_F^{-1}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{F}$. For pixels that fall in region $\mathcal{T}$, we compute the weighted average of the context and focus regions' contributions :

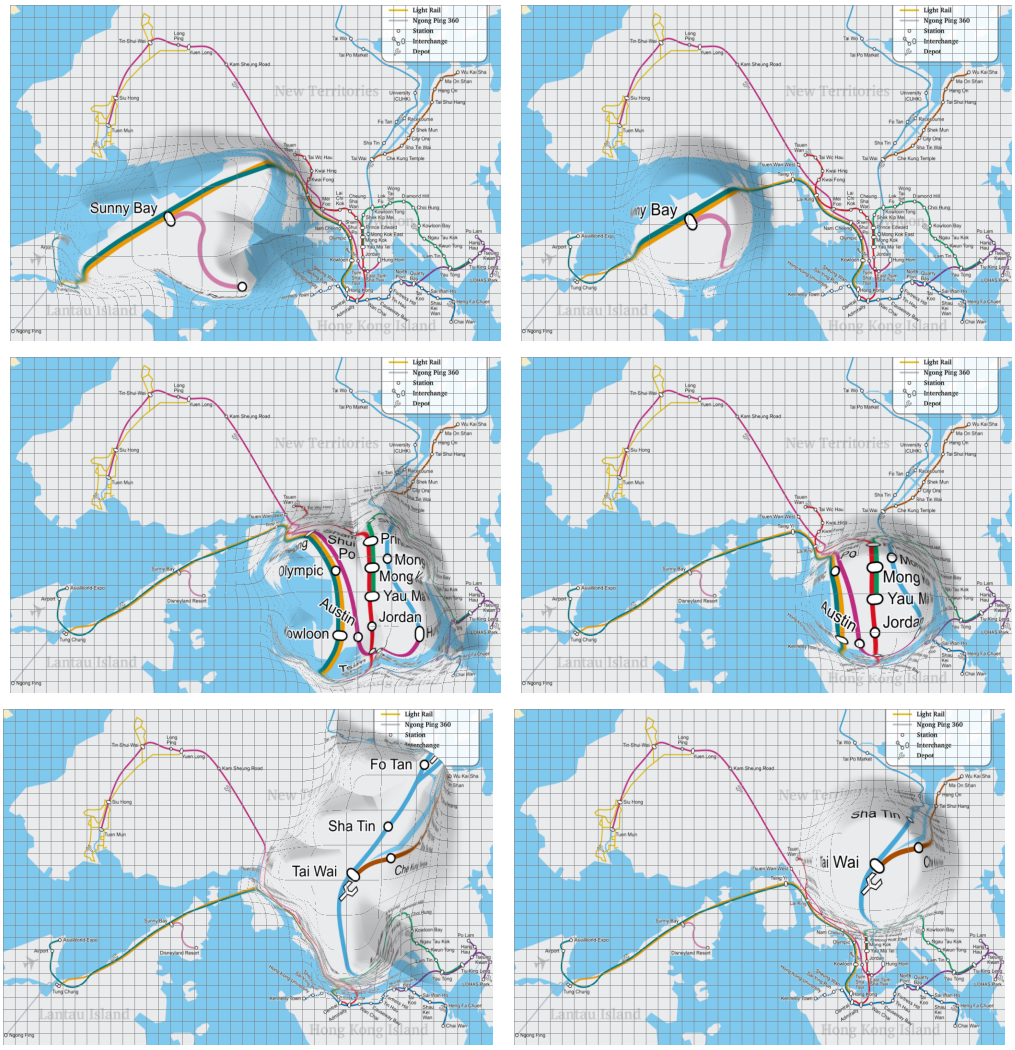
$$T(\mathbf{x}) = \frac{w_F(\mathbf{x}) \cdot T_F^{-1}(\mathbf{x}) + w_C(\mathbf{x}) \cdot \mathbf{x}}{w_F(\mathbf{x}) + w_C(\mathbf{x})}, \forall \mathbf{x} \in \mathcal{T} \tag{2.10}$$

where weights are defined as follows :

$$w_F = (\frac{1}{dist_F(\mathbf{x})})^{b_F}, w_C = (\frac{1}{dist_C(\mathbf{x})})^{b_C} \tag{2.11}$$

with $dist_F$ and $dist_C$ functions returning the distance to the focus and context region contours, respectively.

Parameters b_F and b_C determine how the relative influence of the two regions fall off as a function of distance. As pointed out in [29] : *"creating a distortion presentation is all about finding a balance between the magnification required and some compensatory compression"*. Various profiles have been proposed to specify how to distribute the compression in the distortion, including linear, Gaussian and inverse cosine drop-off functions. Setting $b > 1$ will achieve a smooth transition at the boundaries of the corresponding region. Setting $b < 1$ will result in a sharper transition.



(a) (b)

FIGURE 2.8: (a) Pathlens adapting its shape to the subway lines in various regions of interest on a Hong Kong Mass Transit Railway map (Sunny Bay, Yau Ma Tei, Tai Wai). (b) Comparison with a conventional graphical fisheye of constant shape and size, magnifying areas of lesser relevance.

We implemented this algorithm in C++ and OpenGL, delegating the spatial transformation to programmable graphics hardware. Mapping function T (Equation (2.10)) is based on the computation of distance fields to each region involved in the adaptation. Which distance fields get used can vary from one frame to the next. Before each frame update, distance fields are rendered offscreen by the GPU-accelerated technique introduced in [61], and stored into a depth texture, which is then accessed through simple bilinear interpolation. A shading effect can be added at rendering time, to enhance distortion comprehension [32].

2.5 Discussion and Future Works

We introduced Pathlens, a content-aware adaptive technique extending standard fisheye lenses by adapting the shape of the lens to the content of the scene. We introduced a rendering technique allowing to generate free-form magnification lenses. We introduced a metaphor conveying the idea that the focus region adapts and morphs to the underlying object of interest and we described an implementation of this metaphor.

Pathlenses break the usual behavior of fisheye lenses by proposing to dynamically change their geometry while preserving visual continuity between the focus and context regions. While optimizing what regions get distorted yields more relevant visualizations, it is still unclear how dynamic changes of the shape and distortion can affect users perception and interaction. Some studies are necessary to help better understand some phenomenons from a human-computer interaction perspective. How do the more complex lens shapes affect distortion comprehension and focus targeting ? Should the focus and context contour adaptations take into account the same objects or does independent adaptation yield more meaningful magnifications ? More generally speaking, what is the optimal way of specifying objects/regions of interest ? All these questions are likely dependent on the context of use (task, characteristics of the representation), that must be addressed through controlled experiments.

Adaptation of the navigation tools also raises potential benefits for high-level navigation processes such as construction of the internal representation of the information space [111]. In the same way as disruptions in the physical environment (such as stairs, elevator, doors) are navigational cues and help people find their way in a physical world, as reported by Jul and Furnas in [67], the evolution of the shape of the lens provides navigational cues that could help users remember what part of the virtual space they already examined. Experimental evaluation and comparison against standard technique are needed to better understand such impacts.

Pathlens is based on a content-aware design approach. The adaptation process relies on the geometry of some objects considered of interest. As discussed earlier, identifying objects to take into account during the magnification process is both a matter of deciding what objects are relevant for a given task, a highly domain-dependent problem, and obtaining the geometry of those objects. The latter question is highly dependent on the nature of the graphics. Figures 2.2, 2.3, 2.8 and 2.6 illustrate the use of Pathlenses on vector graphics, that make geometry information readily available. Figures 2.1-c and 3.1 show lenses ap-

plied to a satellite picture, i.e., a bitmap that only contains pixels. In such cases, geometry information can be obtained, e.g., using image processing algorithms that can extract relevant features from the original image, or through manual annotations performed by a domain expert.

However when the information about the geometry is not available, allowing users to specify objects of interest on-the-fly, e.g., by sketching, involving them more explicitly in the adaptation process, is an interesting venue for future work. Possible modalities to consider for such interaction could involve brushing[69], gesture interaction or multi-touch modeling on tactile surfaces. On the contrary, sometimes, several layers of geometry are available : for instance a public transport map featuring buses, subways and trains networks. Steering along an itinerary involving each type of transportation, users might want to interactively select what networks Pathlens will adapt to, enabling to jump from a bus to catch a train connection and follow this line further, and later jump back to a bus line that they will follow. Design and evaluation of such techniques would contribute to better understand the impact of the content-aware approach on navigation, both in terms of human-computer interaction and integration into end-user application.

Chapitre 3

Fisheye Lens for Expanding Objects in 2D Fields

In this chapter we introduce Arealens a content-aware focus+context visualization technique that gradually expands objects of interest depending on user input, smoothly integrating them into the surrounding context.

Arealenses implement a new distortion-oriented presentation technique that enables rendering magnification lenses featuring multiple, arbitrarily shaped foci. This allows to preserve the aspect of objects of interest when they are begin magnified.

The first section introduces in greater depth the issues raised by expanding objects in dense scenes. The following sections present our algorithm for expanding objects and the technique for rendering lenses. We then report on the results of a controlled experiment that compared Arealens to regular graphical fisheyes on a visual search task. We conclude by discussing and presenting future work for Arealens.

3.1 Introduction

Fisheye techniques come in two categories. Techniques that magnify a region of interest around the user's focus of attention, and techniques that expand objects of interest depending on the user's focus of attention. Pathlens belongs to the first category : a focus region is attached to the mouse cursor and dynamically adapts its shape to encompass objects falling beneath, as users move the cursor. However, this approach showed its limits when navigating dense scenes. A Pathlens

needs space between objects of interest to accommodate the distorted transition region, as it would otherwise distort objects of interest, which leads to problems of interpretation [32].

We introduce Arealens, a fisheye technique based on the second category that addresses problems of objects of interest getting distorted by a fisheye (the shape mismatch problem that we introduced previously in Section 2.1.1). Expanding objects of interest smoothly allows to better preserve their visual aspect while transitioning between context regions and full magnification.

Expanding objects on a 2D layout raises the following challenges.

Object visibility : expanding objects might make them overlap with surrounding objects and lead to occlusion. We need a layout management technique that spatially rearranges and shrinks objects so as to both handle magnification of objects of interest and preserve them from self-occlusion. This management technique must bound the spatial rearrangement to a certain radius around the focus of attention so as to preserve context from any perturbation.

Visual continuity : objects of interest highlight interesting regions of the information space. However, useful information might still be present between objects. We need to ensure visual continuity for the whole information space as objects get expanded, moved or shrunk.

Arealens introduces a new mapping algorithm achieving smooth expansion of objects of interest within dense fields while ensuring objects visibility. Arealens also implements a new distortion-oriented presentation technique allowing to magnify piecewise the multiple objects of interest accommodating distortion in-between.

3.1.1 Related Work

Arealenses are related primarily to the following topics : fisheye visualization ; focus+context visualization ; and content-aware image resizing techniques ; that we all presented earlier in the Section 2.2.

Arealenses are also related to the target expansion techniques. Few implementations of this approach were introduced in the literature Bederson presented Fisheye Menus [11] that magnify items of a linear menu. McGuffin [86] studied the usability of target acquisition in such interfaces. However both works only deal with linear 1D layouts. Some implementations can also be found in commercial systems. Integrated in the Mac OSX operating system since early versions, it is

a popular feature of the Dock [1] – a launch bar allowing users to pin icons of applications, folders or files for quick access. With the accumulation of items, the bar, which quickly grows larger than the screen, is shrunk and a magnification effect increase visibility of icons. However we need support for 2D layout. The above only support linear representation, and magnification is made possible by stretching the bar to make room for the expanding icons, which impedes target acquisition [86, 138].

3.1.2 Overall Process

Arealens allows navigation of large datasets by successively magnifying objects within a representation. An information space does not represent only meaningful information and, depending on the scenario, the task, or the interest of users the information they are seeking for is very likely to be encountered in a set of information objects of the scene. Overall, Arealens is based on the same overall process as Pathlens techniques presented in previous Chapter. As explain in Section 2.1.3, how to obtain geometrical description about these information objects is very dependent on the nature of the graphics. In the example presented throughout this Section we obtained the geometrical description with various means. We authored pictures for Figures 3.1, 3.2 and 3.5-(b) because neither Google API nor Mac OS X API provide easy access to this information. We extracted it from vector files for Figures 3.3 and 3.5-(a) and HTML file for Figure 3.5-(c).

Once we obtained the geometrical description, we have to compute a new layout for the objects depending on the actual users input so as to achieve smooth expansion of the objects. We describe our method in the next Section.

3.2 Expansion of 2D Objects with Occlusion Management

We introduce a mapping technique that achieves magnification of objects of interest as user's focus of attention approaches them while preventing objects from overlapping others.

Coupling expansion with occlusion management leads to the following scenario for an object that gets crossed by the user's focus of attention. When distance is greater than a certain radius of influence (a parameter of the technique, equivalent to the radius of the context region in a regular circular fisheye lens), the object is left untouched at context scale. Once within the lens' radius of influence, the

1. http ://en.wikipedia.org/wiki/Dock_(OS_X)

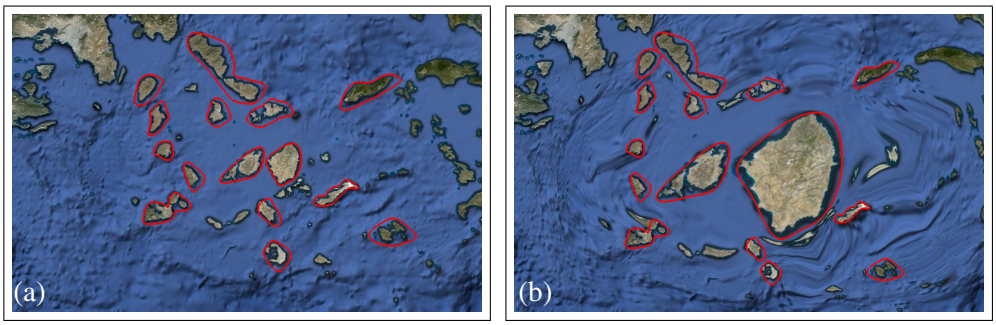


FIGURE 3.1: (a) The Aegean islands (no lens applied). Red contours indicate regions of interest. (b) an AreaLens preserves the islands' appearance (piecewise magnification), putting most of the distortion in the ocean.

object is smoothly shrunk and moved toward the border of the lens as if it were stuck to it, so as to make room for magnified objects. Then, as distance decreases, objects stop shrinking and start to magnify and move toward the focus point, eventually reaching their final magnification. As the cursor goes over them and distance eventually increases, objects go through all these stages in reverse order, to finally revert to their initial position.

Achieving this behavior entails the following requirements :

***R1* Visibility :** magnified objects should not overlap any other object.

***R2* Smooth animation :** sudden changes in the layout of a visualization are disruptive since they prevent users from tracking changes over time [115]. Hence objects should move smoothly.

***R3* Map consistency :** rearranging data to better fit the available space does not necessarily impairs comprehension, as demonstrated by the use of distortion in many maps of transport systems [122, 123]. However layout should preserve topological consistency, i.e., the relative positioning of objects with respect to one another.

***R4* Steady Context :** bounding lens perturbations to a certain area around the focus of attention allows for better focus acquisition [87, 138].

An AreaLens affects each object that falls into its area of influence. Objects that fall outside of it form the context and are left untouched. As illustrated in Figure 3.1, the object(s) closest to the mouse cursor get magnified, piecewise, so as to preserve their aspect (no distortion). To make room for these objects, the objects that are farther away from the cursor but that still lie within the lens' area of influence get offset and scaled down to prevent overlap between them.

The area of influence takes the form of a box centered on the lens' focal point, usually corresponding to the cursor. This box delimits the region beyond which

the lens no longer affects the visualization, thus leaving the context region untouched, thus complying with *R4*.

We use two concurrent mapping algorithms to achieve the other three requirements : the *magnification* mapping, and a *dispersion* mapping. The purpose of the *dispersion* mapping is to push objects away and shrink them to accommodate the objects that will get magnified. The purpose of the *magnification* mapping is to pull objects towards the cursor and magnify them. The *magnification* mapping takes as input the cursor position and identifies the closest objects, to be magnified. The *dispersion* mapping takes as input this set of magnified objects and spreads out the remaining objects (*R1*). We term *magnified objects* the objects identified by the *magnification* mapping, and *movable objects* the remaining ones, spread out by the *dispersion* mapping.

The overall algorithm processes objects one by one iteratively according to their distance to the cursor (closest first). The routine assigns the object its final configuration by applying *magnification* mapping and then assigns the remaining objects to a temporary configuration by applying the *dispersion* mapping. The algorithm then considers the next closest object and stops when no more objects are left to be processed.

A description of the *magnification* mapping and the *dispersion* mapping follow.

3.2.1 Expanding Objects

Full expansion of an object is achieved by applying the following transformation, where **P** is the focal point (cursor) and μ the lens' magnification factor :

$$T(\mathbf{x}) = \frac{\mathbf{x} - \mathbf{P}}{\mu} + \mathbf{P}.$$

However, when processed, the objects might not be lying at their initial position and might have been moved and shrank by the *dispersion mapping* (presented in the next section) to prevent occlusion at a previous iteration. Hence full expansion must be mixed with objects' actual position to ensure smooth transition between full expansion and full shrinking as the cursor crosses the object. This is achieved by applying the following transformation :

$$w \times T(\mathbf{x}) + (1 - w) \times \mathbf{x}'$$

where $w = 1 - \frac{dist}{radius}$, **x** the initial position, **x'** the current position, $radius$ a parameter specifying the lens' operating range and $dist$ its distance to the cursor.

3.2.2 Content-Aware Occlusion Management

The purpose of the *dispersion* mapping is to ensure that no objects overlap. We apply it at each iteration on the objects that are not yet assigned to their final position (the *movable* objects), so that at each step no objects overlap in the scene.

Objects of a scene vary in size and in shape, and do not generate the same visual occlusion. As a result, they do not require the same amount of compression and offset to avoid overlap. For instance, small objects do not need to be shrunk as much as big objects do. Because a naive content-agnostic approach would apply the same treatment to all objects, it would have apply the same compression and displacement to all object, which would lead to a far-from-optimal solution. On the contrary, adopting a content-aware approach, we can adapt scaling and offset depending on the shape and size of objects and consequently accommodate compression and displacement in a more relevant manner. An object-dependent occlusion technique allows for better use of screen real-estate.

The *dispersion* mapping algorithm shrinks and moves *movable* objects to make room for the objects already magnified, which are non-movable objects. The *movable* objects are free of any occlusion prior to the expansion of magnified objects.

The *dispersion* mapping consists of two steps. First, it moves every *movable object* according to a displacement vector. Then, it shrinks every one of them according to a scale factor. The displacement vector is computed as a weighted average of displacement contributions from each *magnified object*. Each displacement contribution is responsible for preventing a *magnified object* from overlapping a *movable object*. Each contribution is computed as $\mathbf{D}_m = \mathbf{x}' - \mathbf{x}$, where $\mathbf{x}$ is the point within the *magnified object* at its initial position (i.e., before magnification) that is closest to the considered *movable object*, and where $\mathbf{x}'$ is the same point on the *magnified object* at its final position (i.e., after magnification). The displacement contribution moves objects away, guaranteeing that objects don't overlap (*R2*) and roughly preserving the relative position of objects (*R3*).

Weights are computed as the inverse distance from a *movable object* to each *magnified object*. An additional contribution is assigned to the area of influence's bounding box to stabilize the position of objects as it approaches them, with a null displacement vector and a weight that is the inverse of the distance to the object. As the first step of the *dispersion* might not be sufficient to guarantee that no overlapping occurs (*R1*), we shrink movable objects by iteratively applying a scaling transformation to them using a constant reduction ratio until they do not

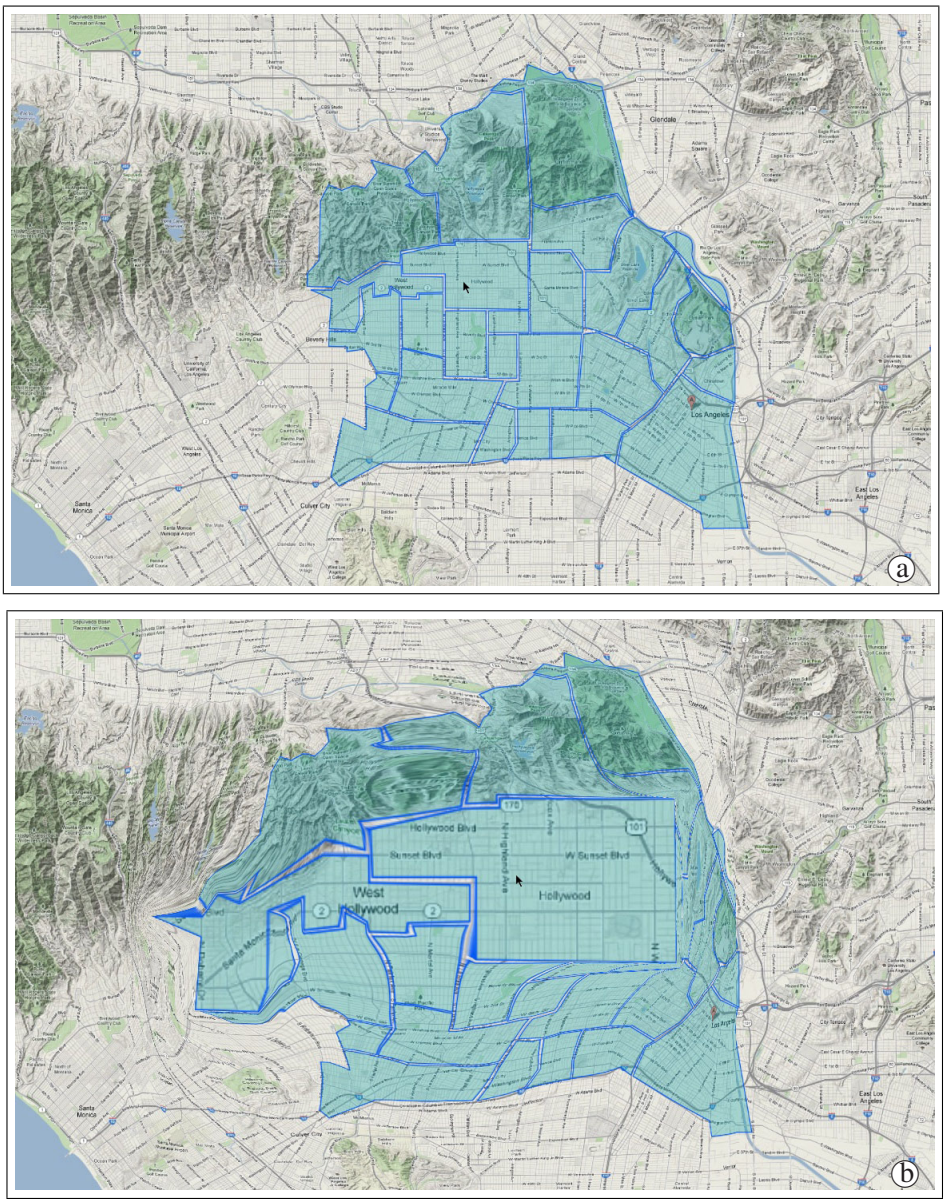


FIGURE 3.2: (a) Map of Los Angeles neighborhoods (no lens applied). (b) AreaLens smoothly expanding neighborhoods of Hollywood and part of West Hollywood.

overlap anymore.

3.3 Rendering Multiple Foci Lenses

The expansion mapping algorithm results in the displacement and scaling of each object. While the *dispersion* mapping guarantees that no overlap occurs between objects of interest, objects of interest can still occlude the information space

between objects. We need a distortion technique that maps objects of interest to their final configuration and map the information space between objects of interest so that it integrates smoothly with objects of interest.

We present a technique that extends the warping technique introduced in Section 2.4 by allowing to render magnifying lenses, with possibly multiple focus regions featuring arbitrary shapes.

The Warping Technique

The warping technique is based on the same approach presented earlier for rendering Pathlenses, except that instead of interpolating only between two regions (the focus region inside the context region), it considers as many focus regions as needed inside the context region.

The expansion mapping technique introduced in the previous section assigns each object a displacement vector $\mathbf{P_i}$ and a scale factor μ_i leading to the magnification transformation :

$$\mathbf{x} = T_i(\mathbf{x}') = \mathbf{P_i} + \frac{\mathbf{x}' - \mathbf{P_i}}{\mu_i} \tag{3.1}$$

The general mapping function is then a weighted average over the contributions of all involved regions :

$$T(\mathbf{x}) = \frac{\sum_{i=0}^{n} w_i \cdot T_i^{-1}(\mathbf{x})}{\sum_{i=0}^{n} w_i} \tag{3.2}$$

where weights are defined as follows :

$$w_i = (\frac{1}{dist_i(\mathbf{x}) + a_i})^{b_i} \tag{3.3}$$

with $dist_i$ functions returning the distance to the each objects of interest contours, respectively. One of the regions is the context region. It encompasses all other ones and is assigned the identity transformation.

Tuning the Distortion

As parameters b_F and $b_\mathcal{C}$ determined how the relative influence of the two focus and context regions fall off as distance increases for Pathlenses, b_i control the influence of each regions of interest. Tuning those parameters makes it possible to adjust the compression relatively to each region. They allow to independently control the shape of the transition at each regions boundary. Setting $b > 1$ will

achieve a smooth transition at the boundaries of the corresponding region. Setting $b < 1$ will result in a sharper transition.

The other parameters a_i are parameters that change the relative flatness of the flat-tops for optimal adaptation. When $a_i = 0$, strength is infinite for pixels on the contour ($dist_i(\mathbf{x}) = 0$) leading to a flat mapping inside the corresponding focus region. When $a_i > 0$, this flatness constraint is relaxed a bit, enabling a better packing of foci through partial distortion.

This can be interesting in cases where the information space is densely populated with regions of interest, as illustrated in Figure 3.2. Indeed, in dense environments, the transition region is small compared to that of the regions of interest and cannot absorb all the distortion. Allowing distortion in the regions of interests distributes it over a wider area and attenuates it.

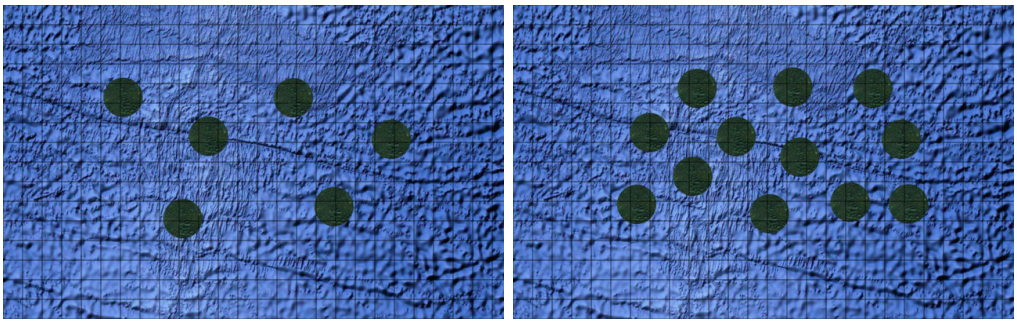
3.4 Evaluation

The different multi-scale interface schemes (focus + context, overview + detail, zooming) all have their advantages and drawbacks, depending on the user's task and on the nature of the visualization ; their empirical comparison has been the topic of several papers [35, 62, 92]. Our primary goal is to empirically evaluate the potential benefits and pitfalls of adaptive lenses compared to statically-defined lenses. The purpose of this experiment was both to evaluate the actual performance gain under different conditions, if any, and to assess the potential negative impact of the dynamically changing geometry, that might cause confusion and visual discomfort. For this first experiment, we compared regular fisheyes (circular shape, Gaussian drop-off) to AreaLenses, that have a stronger impact in terms of visual changes than PathLenses, as they affect more objects and are thus more likely to suffer from this, especially in dense configurations.

The task was an abstract multi-scale visual search task, operationalized so as to test the following two hypotheses, based on the expectation that by adapting their shape to match that of object(s) of interest, AreaLenses should provide more relevant detail-in-context visualizations than regular fisheyes, requiring less virtual navigation while minimizing the negative effects of distortion that impede comprehension :

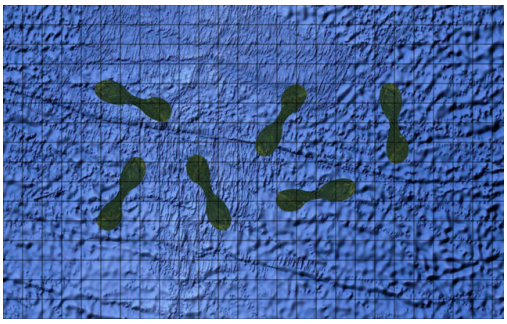
H_1 AreaLens performs better than fisheye when the objects of interest have irregular shapes, as the focus of AreaLens adapts to those shapes while that of fisheye does not ;

H_2 AreaLens performs worse when the density of objects of interest is high, as this leads to more visual distraction caused by the movement of a larger

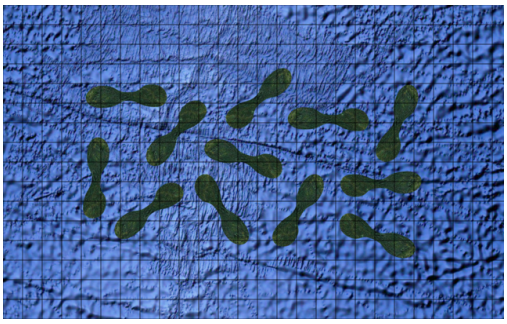


SHAPE = *Circle* & DENSITY = *Low*

SHAPE = *Circle* & DENSITY = *High*



SHAPE = *Blob* & DENSITY = *Low*



SHAPE = *Blob* & DENSITY = *High*

FIGURE 3.3: Screenshots of the four maps combining two different SHAPE and DENSITY. Users were to find a target hidden in one of the island, visible only when magnified with the lens. We compared performance of an Arealens against performance of a static fisheye lens of same focus region surface area.

number of objects.

3.4.1 Task and Procedure

In all trials, the information space was made of islands on an ocean (Figure 3.3). The task consisted in finding a red cross that was positioned on one of the islands, by magnifying each island in turn. The red cross was not visible at context scale and could only be revealed when the corresponding region was magnified through the lens. We used the same texture to represent the ocean in all trials (bathymetric data). Forest textures used to fill the islands all looked similar, so as to ensure that the red cross would be equally difficult to visually differentiate from the background on all islands.

To operationalize *H1*, we considered two types of shapes (factor SHAPE) : islands were either circles (*Circle*, Figure 3.3-top) or elongated blobs (*Blob*, Figure 3.3-bottom). All islands had the same surface, to make sure that differences could be attributed across conditions to variations in shape, as opposed to area. To ensure fair treatment of both techniques, we made the *Fisheye*'s flat-top the exact

same size as the circular islands. Thus, for each condition, the flat-top of both lenses and the islands had the same surface. We did not make the flat-top of the *Fisheye* lens larger to avoid problems caused by the distortion of larger context areas, that impede focus targeting, and thus task performance, and make context information difficult (at best) to understand in real world cases, as illustrated in Figure 2.1.

To operationalize *H2*, we tested two different object densities (factor DENSITY) : a *Low* density with a significant amount of empty space (ocean) between islands (Figure 3.3-left) ; and a *High*er density (Figure 3.3-right) with more islands packed in the same space. AreaLenses would cause more visual disturbance in the *High* condition : more objects had to be moved and resized to accommodate the magnified object of interest, as their was less empty space to "absorb" distortion.

Randomly placing the cross before a trial starts would obviously have been a significant source of noise in the experimental data depending on how lucky participants were in each trial. As this chance factor cannot be balanced across conditions, we artificially forced chance upon participants, following the approach described in previous work on the operationalization of multi-scale search tasks [92]. We introduced a secondary factor, NAVLENGTH, that determines *when*, rather than *where* the cross appears, based on what fraction of the entire scene (considering island surfaces only) the participant had already visited with the lens. Thus, for a value X of factor NAVLENGTH for a given trial, the red cross only appeared after participants had visited $X\%$ of the total surface of all islands, no matter in what order those islands had been visited. The cross appeared right next to the lens, according to the current exploration trajectory. NAVLENGTH was set to one of three values, 30%, 55% or 80%. Participants were completely oblivious of this trick, and were told about it only after the end of the experiment.

Twelve unpaid volunteers (two females), daily computer users, age 24 to 36 year-old (average 30.3, median 30.5), served in the experiment. All had normal or corrected-to-normal vision and did not suffer from any form of color blindness. We conducted the experiment on a 2x2.26GHz Mac Pro workstation running Mac OS X, equipped with an NVIDIA GeForce GT 120 512 MB driving a 30" LCD monitor (2560×1600, 100 dpi), and a standard optical mouse set at 400 dpi resolution and default system acceleration.

To summarize, the experiment was a 2×2×2×3 within-participants design with the following factors :

— 2 techniques (TECH) : *Fisheye* and *AreaLens* ;
— 2 types of shape (SHAPE) : *Circle* and *Blob* ;

— 2 types of shape density (DENSITY) : *Low* and *High* ;
— 3 navigation lengths (NAVLENGTH) : 0.3, 0.55 and 0.8.

We grouped trials into two blocks, one per TECH. Half of the participants started with *AreaLens* ; the other half with *Fisheye*. Both blocks started with a training session consisting of 3 repetitions of each of the 12 SHAPE × DENSITY × NAVLENGTH conditions presented in a random order. The operator used the first few training trials to explain the task and techniques. After the training trials, each block continued with a series of measured trials, consisting of 6 repetitions of the same 12 conditions presented in a random order. Participants were instructed to perform the task as fast as possible while minimizing the number of errors. To complete a trial, participants had to reveal the red cross through the lens and click the mouse button with the cross still visible. To avoid participants rushing through the experiment, mouse clicks that occurred while the cross was outside of the lens were counted as errors, but did not end the trial ; participants still had to find the cross to complete the trial. The average duration of the experiment was 30 minutes per participant.

3.4.2 Quantitative Results

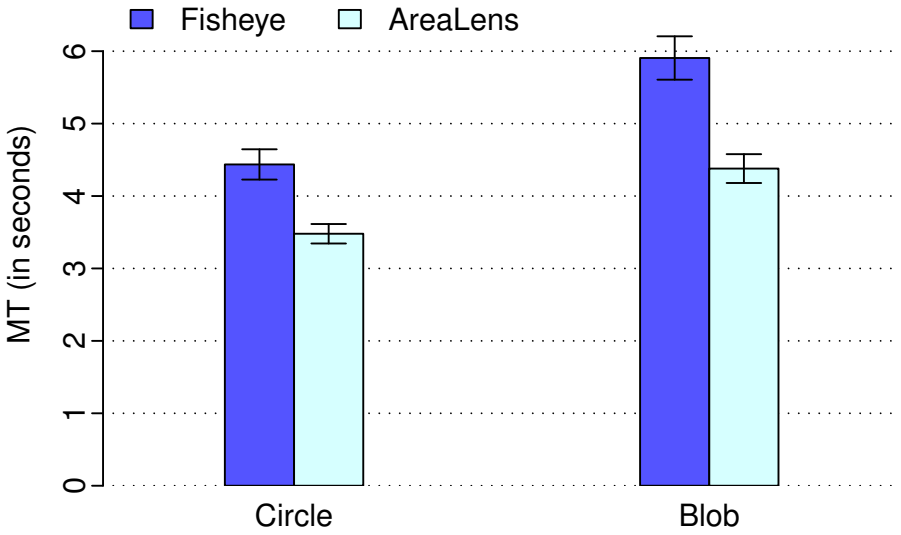


FIGURE 3.4: Quantitative results of the controlled experiment. *MT* (tasks completion time) for both Fisheye and Arealens techniques and for the two SHAPE conditions. Error bars show the 95% confidence limit.

We used R (`http://www.r-project.org`) for our statistical analysis and in particular the `aov` command from the `stat` package. Our main measure was task completion time *MT*. As participants had to successfully complete each trial, errors were integrated in *MT* and were thus not studied on their own. We performed

a full factorial nominal ANOVA for the following model :

MT $\sim$ TECH$\times$SHAPE$\times$DENSITY$\times$Random(PARTICIPANT).

We did not include NAVLENGTH in the model as we consider it as a secondary "ecological" factor whose purpose is to prevent noise due to chance, as explained earlier. We removed a few outliers : 1.44% of all trials, with unreasonable (> 1) residual/predicted ratio. Those were equally distributed among *AreaLens* and *Fisheye* conditions.

The ANOVA reveals a significant effect of TECH on *MT* ($F_{1,11} = 17.3, p = 0.0016$). *AreaLens* is significantly faster than *Fisheye*, yielding a 24% speed-up. As expected, we find a significant interaction TECH$\times$SHAPE ($F_{1,11} = 13.3$, $p = 0.0039$) that can be observed in Figure 3.4. A post-hoc t-test with Bonferroni correction shows that *AreaLens* is significantly faster than *Fisheye* for both *Circle* ($p = 0.0002$) and *Blob* ($p < 0.0001$) island shapes, but the magnitude of the difference is larger for *Blob* (25.8%) than for *Circle* (21.6%). Thus, hypothesis *H1* is supported by our data.

The ANOVA also reveals a significant effect of DENSITY ($F_{1,11} = 119.5$, $p < 0.0001$), participants being significantly faster in *Low* density conditions. This was expected as, by design, the *High* density condition requires more navigation for the same value of NAVLENGTH. However, the ANOVA reveals neither a TECH$\times$DENSITY nor a TECH$\times$SHAPE$\times$DENSITY interaction. This implies that we cannot say more than what is said in the previous paragraph on the difference between *AreaLens* and *Fisheye* with respect to the DENSITY factor (*AreaLens* is significantly faster than *Fisheye* for both densities, but the data does not show an effect of DENSITY on this difference). Thus, *H2* is not supported by the results of the experiment. Note, however, that *AreaLens* is significantly faster in *Circle* than *Blob* conditions. This might be due, in part, to the higher amount of visual change in the latter case.

3.4.3 Qualitative Results

At the end of the experiment, participants were asked to rank *AreaLens* and *Fisheye*, and were encouraged to provide feedback. Eight participants out of twelve ranked *AreaLens* first. The other four ranked *AreaLens* and *Fisheye* ex-aequo (no participant ranked *Fisheye* first). This is consistent with quantitative results and supports *H1*. Among participants who ranked *AreaLens* and *Fisheye* ex-aequo, two said that they preferred *AreaLens* in the *Blob* condition and *Fisheye* in the *Circle* condition. Two participants found the *AreaLens* deformations sometimes annoying, though this did not impact navigation between shapes significantly.

Overall, participants did not find the changing lens shape particularly distracting ; even if more predictable, distortion when moving classical fisheyes also entails constantly changing shapes. Some participants commented on the difference between passively watching the interface and actively operating *AreaLens*, confirming our own observations that while shape changes can be distracting in the former case, they are not in the latter. We tentatively attribute this to operators being more tightly integrated in the feedback/ feedforward loop than people passively watching the screen, and already having their attention focused on the region of interest when relocating the lens, thus better anticipating changes. One remark that came often (formulated differently) is that *Fisheye* requires more concentration because both navigation and visual search (looking for the cross) have to be performed simultaneously. Performing the task with *AreaLens* is cognitively less demanding, as the two tasks can be sequentialized : one can jump to the next shape and then perform the search on the entire island at once. Another frequent remark was that *Fisheye* is more annoying because it deforms (distorts) the islands, something that does not happen with *AreaLens*.

3.4.4 Summary

The results of this experiment are very encouraging : the *AreaLens* significantly outperformed a classical fisheye lens in all conditions, including *Circle*. It was not obvious that dynamic adaptation would provide any advantage in this case, given that a *Fisheye* already provides a good match in terms of shape adaptation, and that it does not suffer from potential problems of visual distraction due to dynamically changing geometry. The results indicate that the latter is actually not a significant usability issue, as it did not negatively impact *AreaLens* performance. In terms of subjective preferences, we received positive feedback from the participants. In accordance with quantitative measurements, participants did not report being distracted by the visual changes in regions of interest when using *AreaLens*. This confirms that this potential problem has a very limited impact, if any, though further investigation is required to generalize this particular finding.

3.5 Discussion and Future Works

We introduced Arealens, a multi-foci fisheye lens that achieves interactive expansion of objects in dens fields while preserving continuity. We introduced both a mapping technique for expanding objects within 2D fields, and a distortion-oriented presentation technique allowing for lenses featuring multiple foci of arbitrary shape.

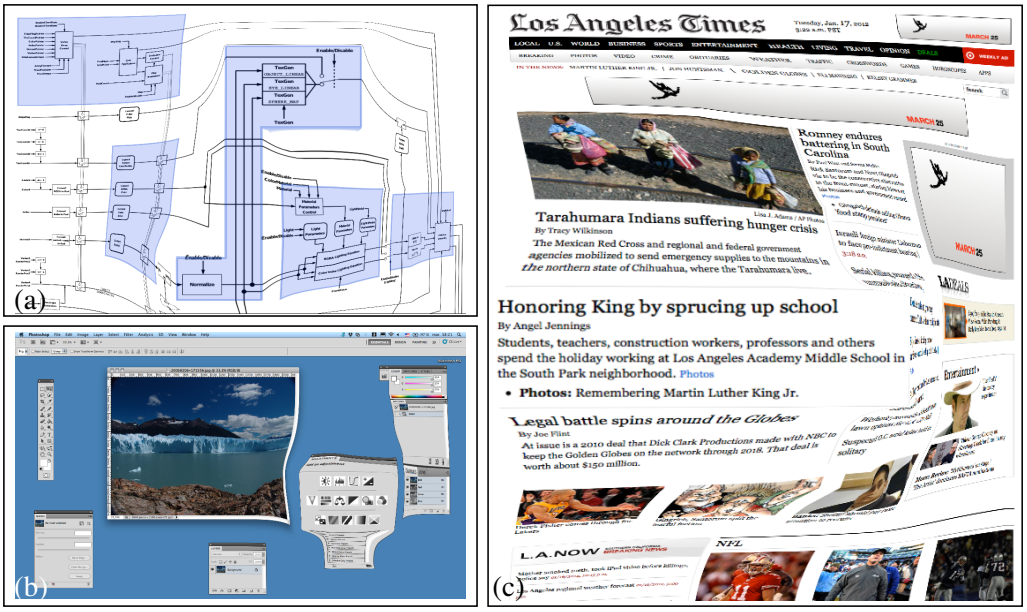


FIGURE 3.5: The AreaLens technique applied to : (a) a part of the OpenGL state machine block diagram ; (b) magnifying a group of widgets in a user interface (Adobe Photoshop) ; and (c) the Los Angeles Times Web homepage.

Arealenses can also apply to different contexts. For instance, they could be used as accessibility helpers for visually impaired users, to magnify widgets or groups of widgets in desktop user interfaces (Figure 3.5-b), or to magnify portions of Web pages such as a newspaper homepage (Figure 3.5-c). The technique makes it possible to display an overview of the full page, and to browse it at a readable level, somewhat like Fishnet [8] does. This can be useful in the context of users who need accessibility features, but also on mobile devices such as tablets and e-readers, that have limited screen real-estate. Note that in both of the above examples, finding objects of interest and obtaining information about their geometry is relatively straightforward : in the case of user interfaces, accessibility/automation APIs enable the automatic discovery of the UI widget hierarchy ; in the case of Web pages, the CSS box model and HTML page structure can help identify and segment relevant parts of the web page. Another interesting family of graphics are block diagrams. Figure 3.5-a shows an example depicting part of the OpenGL state machine. Again, it is possible in such cases to automatically find objects of interest, e.g., sets of boxes, paths connecting components, or coherent sets of connected entities that form components of higher abstraction.

One of our main concerns about Arealens was the potential distraction caused by the constantly moving shape of the lens. Our experiment showed that this had no significant negative effect on the user who is operating the lens. Users reported

being distracted while observing others using the lens, though. This might be due to the fact that when operating the lens, user's focus of attention matches the center of the lens, lying within the main flat top, where only few changes occur. While, on the contrary, observers might focus on other locations in the scene, and be more affected by the moving shape. This effect might have more consequences on collaborative navigation for instance, and further empirical evaluation is required to better understand this phenomenon.

Arealenses rely on geometrical description of the content of the scene identified as being of main interest. While in many cases this information is readily accessible, sometimes the description is overwhelming : too much information is available and users need to select layers, group or a granularity within the content that they want the Arealens to adapt to. For instance, in the HTML source of a web page, users might be interested in pictures, blocks of text, menus or hyperlinks. Techniques letting users select, within a scene, a specific semantic structure, would allow for improved usability of Arealenses.

Arealens and Pathlens are based on the same approach. Both propose adaptive fisheye lenses, whose shape adapts to the geometry of the content of the scene, identified as being of interest for the user. However Pathlenses adapt to path, and Arealenses expand objects in dense 2D fields. Whether to use one or the other is a matter of content, and also of user's interest. For instance, inspecting a 2D graphics scene users might first choose Pathlens to inspect shapes of objects, and then choose Arealens for inspection of the filling content. Actually choosing between Arealens and Pathlens is a matter of what users are interested in.

Chapitre 4

The Jellylens Library

In this chapter, we present the Jellylens library that was created for the development of distortion-oriented presentation techniques, featuring multiple foci of arbitrary shape. The library is based on a simple mathematical formulation providing easy controls : users only manipulate the geometry of each foci, the mapping inside each foci, and a few parameters to generate a wide range of presentation techniques integrating each foci into the surrounding context continuously. The formulation is compatible with a GPU implementation achieving interactive rendering, which allows for dynamic adaptation of lenses.

The library was fully implemented and was used for prototyping the two techniques introduced in chapters 2 and 3.

4.1 Introduction

Focus+context multiscale interfaces integrate smoothly several representations of a datasets at different levels of magnification. In-place magnification of regions of a dataset entail some compensatory compression to avoid occlusion of the surrounding contextual information. Although distortion enables a smooth transition between the different scales of representation, it also causes problems of recognition and interpretation.

Many approaches have been described in the literature for stretching and distorting spaces to produce multiscale presentations of data. Such techniques have been reviewed previously in Section 2.2. A first major limitation of the work introduced earlier, was referred to as the “tyranny of the foci” by Keahey [69] : most of the techniques introduced rely on the definition of centers of magnifica-

tion and provide magnification fields that smoothly interpolate with the various foci. Such techniques are : polyfocal projection [68], fisheye views and its derivative [51, 103] or distortion-oriented presentation[77, 76]. Other techniques allow to define foci with arbitrary shapes such as : 3-dimensional pliable surfaces[30] and non-linear magnification fields[69]. They integrate continuously the foci using distortion. However these technique do not allow to define the shape of the context region to constrain distortion to a certain region.

Most of previous work in that area focused more on how to tune the distortion profile in the transition region than the shape of the distortion region. The problem of mismatch between the shape of the lens and the geometry of objects of interest, described in Section 2.1.1 pointed out that problems of interpretation are caused by objects of interest falling in the spatially distorted region. In other words, we argue that what regions distortion affects is of greater importance than how it affect it. We then need techniques affording greater control over the shape of the distorted region.

We introduce a new presentation technique that enables fine control over the shape of the region where distortion happens. Users manipulate handles, which are regions where data is magnified with no distortion – resulting in the so called flat-tops– and define an overall encompassing shape delimiting the context region. Our technique accommodates distortion in the regions in-between to integrate continuously each focus region and the context region. The library provides as well controls allowing to tune distribution of compression, allowing to achieve several distortion profiles such as sharp or smooth transitions at the boundary of each focus.

We formally introduce the technique and then present our GPU implementation. We will present then other use that we investigated with this technique and conclude with discussion and future work.

4.2 Lens Formulation

When developing the Jellylens library, a distinction was made between representation of the dataset and its presentation. The Jellylens library does not consider the representation of the information. Users of the library are required to provide an image representing the data. Although the library could handle both pixel-based and vector-based images, during this thesis we only provided an implementation for pixel-based representations.

The base elements of the library are the *glasses*. *Glasses* are handles to control

the presentation. A *glass* is defined by a region of the information space. A few parameters are used to tune the overall distortion in the transition region, and they apply a transformation to magnify, compress, translate or rotate locally the presentation.

To understand the role of a *glass*, let's first consider the case where only one *glass* is placed on the presentation space. In this case, actions on the handle such as : moving, scaling or rotation, are directly echoed on the entire presentation space, resulting on the moving, scaling and rotation of the whole representation.

When two *glasses* are placed on the presentation space, each *glass* echoes directly the transformations to the region it is attached to. And the region in-between is distorted to produce a single continuous presentation integrating smoothly both regions' mapping.

The second entity of the library is a *lens*. *Lenses* are made of several *glasses*. A typical *lens* includes *glasses* to define each focus with their respective magnification transformation. Another *glass* that encompasses all the other *glasses* is defined and associated to the identity transformation to handle the unmagnified context region. The distortion will be constrained to regions in-between all the *glasses* that we call the *transition* region.

Mathematics :
Glasses allow to define linear mappings into regions of the presentation space. The distortion within the transition region achieving the smooth integration of all the *glasses* is defined as an interpolation of the transformation of each *glass*. The mathematics were already introduced in Chapter 3. They are briefly summarized here :

$$T(\mathbf{x}) = \frac{\sum_{i=0}^{n} w_i \cdot T_i^{-1}(\mathbf{x})}{\sum_{i=0}^{n} w_i} \tag{4.1}$$

where T_i are the transformations associated to each *glass*, and weights are defined as follows :

$$w_i = (\frac{1}{dist_i(\mathbf{x}) + a_i})^{b_i} \tag{4.2}$$

with $dist_i$ functions returning the distance to each *glass*, respectively.

This formulation allows to define precisely the geometry of foci free of any distortion and the transition region smoothly integrating all *glasses*. It provides also two parameters to adjust the profile of the distortion assigned to each *glass*.

Tuning the distortion

Parameters b_i control the influence of each *glass*. Tuning those parameters makes it possible to adjust the compression relatively to each *glass*. They allow to independently control the shape of the transition at each *glass* boundary. Setting $b > 1$ will achieve a smooth transition at the boundaries of the corresponding *glass*. Setting $b < 1$ will result in a sharper transition.

The other parameters a_i are parameters that change the relative flatness of the flat-tops for optimal adaptation. When $a_i = 0$, strength is infinite for pixels on the contour ($dist_i(\mathbf{x}) = 0$) leading to a flat mapping inside the corresponding *glass*. When $a_i > 0$, this flatness constraint is relaxed a bit, enabling a better packing of foci through partial distortion.

This can be interesting in cases where the lens is composed of several *glasses* close to each other . Such configuration results in a small *transition* region for accommodating the distortion which yields high compression factors that can lead to discontinuity or create disruptive visual artifacts such as Ghosts (reported by Beier [15] when studying image warping techniques to achieve image metamorphosis). Allowing distortion in the *glasses* distributes the distortion over a wider area and attenuates it.

Design Considerations
The Jellylens library allows to define magnification lenses with as many foci as necessary with the only restriction that *glasses* must not overlap. Otherwise, discontinuities will occur at the boundary of the foci. However users must bear in mind that combining *glasses* whose magnification, offset or rotation varies with great amplitude yield great compression ratio in the *transition* region which need space to absorb the distortion. In some extreme cases, where the layout call for small *transition* regions, users can tune parameters a_i to leave distortion inside *glasses*.

Combining Multiple Lenses
The Jellylens library supports multiple lenses into a presentation space, to create rich presentations featuring several fisheye lenses. However, combining several lenses that do not constrain their distortion into a bounded context region is not the purpose of this library and leads to discontinuity in the overall presentation. Equation (4.1) defines presentation for each lens independently from others.

4.3 Implementation

One of the main requirement for the Jellylens library was to achieve interactive framerate rendering, while shapes of lenses could change from one frame to

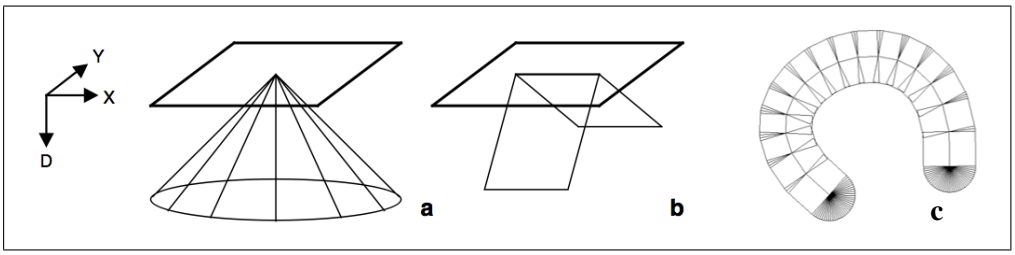


FIGURE 4.1: Implementation of the Jellylens library involve the Fast Distance Fields Computation on the GPU introduced in [60]. The technique involves representing each shape by a distance mesh : the distance meshes used for a point (a) and a line segment (b) shown with the XY plane above each mesh. The distance mesh for a polyline (c) seen from above. Taken from [60].

the next. Hence Jellylens library is compatible with parallel rendering and we implemented the library on the GPU using C/C++, OpenGL and GLSL shaders. In this section we give details on its implementation.

The overall image is first rendered to provide the overall context region on a first pass. Then, each lens is rendered independently. For each lens, users specify the viewport on which to render the lens. To save some computation, a good practice is to define the viewport as the bounding rectangle of the lens.

Distance Fields

The definition of the overall mapping requires to compute the distance of each point on the presentation to each *glass* which can be resource demanding and thus time consuming. Before each frame update, distance fields are rendered offscreen by the GPU-accelerated technique introduced by Hoff in [60]. The method consists of converting each polyline into a distance mesh (see Figure 4.1). Rendering into a depth texture results in the sampling of the distance fields. The texture is then accessed through simple bilinear interpolation to obtain the distance at each point.

Distance fields are generated for all *glasses* into as many depth textures. The number of textures allowed by OpenGL drivers, which varies across platforms and hardware, can induce a limitation on the number of *glasses* allowed by the library.

Displacement Map

Once the distance fields are generated for each *glass*, we can render the presentation. The overall mapping (4.1) is applied to each point on the presentation space to render the presentation.

The interpolation method was implemented in a GLSL shader. The shader to use

depends on the number of *glasses* to interpolate, which can vary over time. To this end, we implemented a run-time generation of the shader, that generate the appropriate shader depending on the number of *glasses*.

Shading
Carpendale [32] proposed to use shading to make distortion more comprehensible by users and alleviate problem of interpretation. Our library allows to add shading to the rendering of lenses.

To add shading to the rendering, we first render an off-screen height map. Each *glass* is associated with a height corresponding to the magnification factor of their respective transformation. Theses heights are interpolated before the rendering and rendered into a texture resulting in a height map. The interpolation is defined as a weighted average over all involved *glasses*.

$$T(\mathbf{x}) = \frac{\sum_{i=0}^{n} w_i \cdot h_i(\mathbf{x})}{\sum_{i=0}^{n} w_i} \tag{4.3}$$

where h_i encode the respective height of each *glass*. The weights are defined as in Equation (4.1) by

$$w_i = (\frac{1}{dist_i(\mathbf{x}) + a_i})^{b_i} \tag{4.4}$$

During rendering, the gradient of the height map is used as a surface normal $\mathbf{n}$ in a Lambertian shading model :$I = C \times dot(\mathbf{n}, \mathbf{l})$, with $\mathbf{l}$ the light source direction, and C a constant. I is the resulting luminosity, see Figure 2.8 for results.

4.4 Applications

Although the Jellylens library has originally been designed and implemented for prototyping both the Arealens and Pathlens techniques, we found it convenient for investigating other problems. Creating presentation techniques simply by controlling the flat-tops (*glasses*) appears to be a clear and intuitive paradigm for fast prototyping, which eases investigation of distortion-oriented presentation techniques. Here we present two examples. The first addresses issues related to the use of multiple lenses on a display. The other one addresses problems related to physical navigation in wall-sized display environments. Both prototypes only required less than 50 lines of code, using the Jellylens library.

Fisheye Merging
Using multiple fisheye lenses on a display has multiple benefits : it allows several

FIGURE 4.2: Two fisheye merging. (a) flat-tops with the same magnification factor merge, requiring similar magnification factor ; (b) flat-tops with distinct magnification factors are cropped.

people to collaborate for navigating a dataset and it eases comparison tasks as two lenses can focus at the same time on remote parts of the information space. However, a fisheye alters representation of the region beneath, and conflict must be handle as two lenses come in the vicinity of each other. A simple solution is to prevent them from overlapping by blocking the lenses. Thus, regions beneath a lens cannot be explored with another one. Another solution, introduced by Carpendale [28], is to let lenses overlap and combine their magnification power. However, this solution allows to explore the information "through the eye" of the first lens. The transition region of the first lens still appears distorted when magnified by the second lens.

Addressing this issue, we designed two solutions that we were able to prototype quickly with the Jellylens library. Both are based on the same idea, that transform the two single-focus fisheye lenses into one double-foci fisheye lens when their transition regions overlap. This allows a lens to explore the region falling under the transition region of the other lens, flattening the presentation locally. The two techniques diverge on how to handle conflict of the two focus regions. They apply to two different context.

The first technique makes the two focus regions, merge into one single large focus region, encompassing the two others. We implemented the merging with a Metaball effect as illustrated by Figure 4.2-(a). However, merging two focus regions require them to have the same magnification factor, which constrains

navigation strategies.

The second technique does not impose any requirement on the fisheye lenses. The idea is to crop focus regions when they come in the vicinity of each other, thus preventing them from overlapping (see Figure 4.2-b). This approach results in two smaller focus regions, which hinder visibility a bit.

Some evaluation would be necessary to compare these two techniques, and to better understand their benefits and drawbacks from a usability perspective. Implementation of both techniques involved controlling 2 to 4 *glasses*, setting their shape, mapping transformation and distortion parameters : their implementation with the library required less than 100 lines of code.

Offsetting Views

Another problem Jellylenses could provide a solution for is a visibility issue raised by wall-sized displays.

Wall-sized displays made of tiled monitors feature both a high resolution (typically 100 DPI) and large display surface area. Such environments provide unprecedented capabilities to display large amounts of data with high level of details. They usually require, users to move inside the room to view all the details of the representation. We usually refer to this as *physical navigation* which can be coupled with virtual navigation, such as techniques described in [89, 43]. To inspect the details of a representation, users typically approach the display, and move away to embrace a more general overview of the dataset. However, the different regions of the display are more or less accessible to the users, due to their remote location. For instance, the bottom row of monitors requires users to bend over to view details. The top row might not even be accessible to users depending on how tall they are.

Some works address virtual navigation into wall sized displays [89], however, physical navigation is still needed to appreciate the full resolution provided by the display. Some other address the problem of occlusion generated by the frame so the monitor [43]. Problem of visibility of remote region of the screen was studied from users perception point of view in [16], and [70] introduced the Frisbee widget allowing to display remote regions, however do not guarantee continuity.

We introduce a technique based on the observation that users tend to allocate themselves columns of the display while inspecting the display closely. Indeed, users generate occlusion while approaching the wall. This occluded region gets momentarily inaccessible to other users that thus have to wait, or to deport their

attention to other regions. As a result, users often end lining-up in a row facing the display, switching their position for navigating. Building upon this idea, we decided to allocate to each user a column of the display. A user can use this row to offset the top or bottom regions (see Figures 4.4 and 4.5). The technique uses distortion to ensure visual continuity between the offset region and the rest of the display. With such a technique, users can temporarily raise the lower part or pull down the upper row, improving their visibility, and the navigation comfort at the same time.

We implemented a prototype on desktop computers with less than fifty lines of code, using the Jellylens library. Only two *glasses* are need, one to offset the region and the other one to constraint the distortion to a bounded region. Providing support for Jellylens rendering on cluster-driven wall-sized displays, requires some future work that we discuss in the following Section. However, with such support, the code could be ported as is to achieve a fully functional prototype, running on wall-sized displays.

4.5 Discussions and Future Work

We introduced a library for creating distortion-oriented presentations techniques. The library introduces a simple interface consisting of composing lenses with *glasses* elements. The library was fully implemented and used for prototyping both Arealens and Pathlens techniques. The library allowed us to investigate two other problems related to the merging of fisheye lenses, and to the accessibility of remote display region on a wall-sized display.

However several issues remain to be addressed.

Ghost Effect

The library successfully provides fine grain control over where to accommodate the distortion, and how to accommodate it. However, in some extreme cases, when transition regions are small and magnification ratios are high, some disruptive visual artifacts can occur (for example, the black hole in Figures 4.4 and 4.5), namely Ghosts which were first reported by Beier [15]. The library provides users with controls to let *glasses* absorb the distortion, which minimizes the effect. However, this increases the burden of prototyping. Other interpolation methods guaranteeing bijection between the source and the target images could be investigated, bearing in mind that the method must be compatible with parallel rendering.

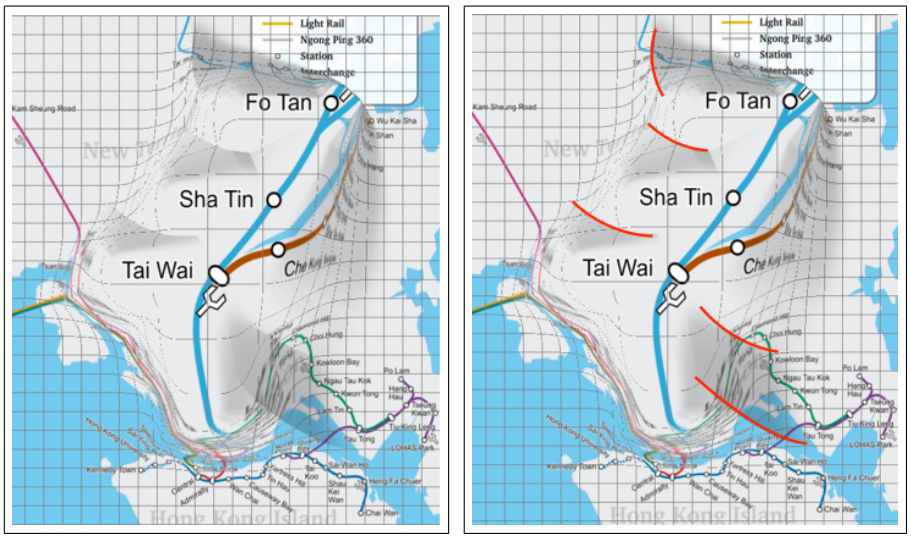


FIGURE 4.3: Sharpe edges caused by the distance fields' Topological Skeleton, can appear in lenses whose shapes feature sharp angles.

Porting to Cluster-Driven Wall-Sized Display

As illustrated by the application introduced in Section 4.4, the Jellylens library could be used for designing wall-sized display presentation techniques. However, such displays are usually made of multiple monitors tiled together and driven by a cluster of computers, each computer driving one or two screens. Most graphics cards cannot handle textures large enough to cover wall-sized displays, because of hardware video memory limitations and restrictions on maximum texture size in OpenGL. The mapping allowed by the library requires, in extreme situations, loading and rendering the entire representation on a single screen, putting a heavy load on the associated cluster node. This require techniques allowing management of large tiled images by modern graphics hardware. A possible solution would use virtual texture management, such as the one implemented in [43].

Topologicol Skeleton

We observed some sharp edges generated by some lenses created by the Jellylens Library (see Figure 4.3). Those edges are caused by the topological skeleton of the distance fields of convex shapes. This problem can be addressed by sampling the distance field at a lower resolution, which would blur the distance fields, thus the topological skeleton. However this would as well blur the boundary of *glasses*. A solution would be to combine two levels of sampling, to perserve *glasses* boundaries and blur internal edges.

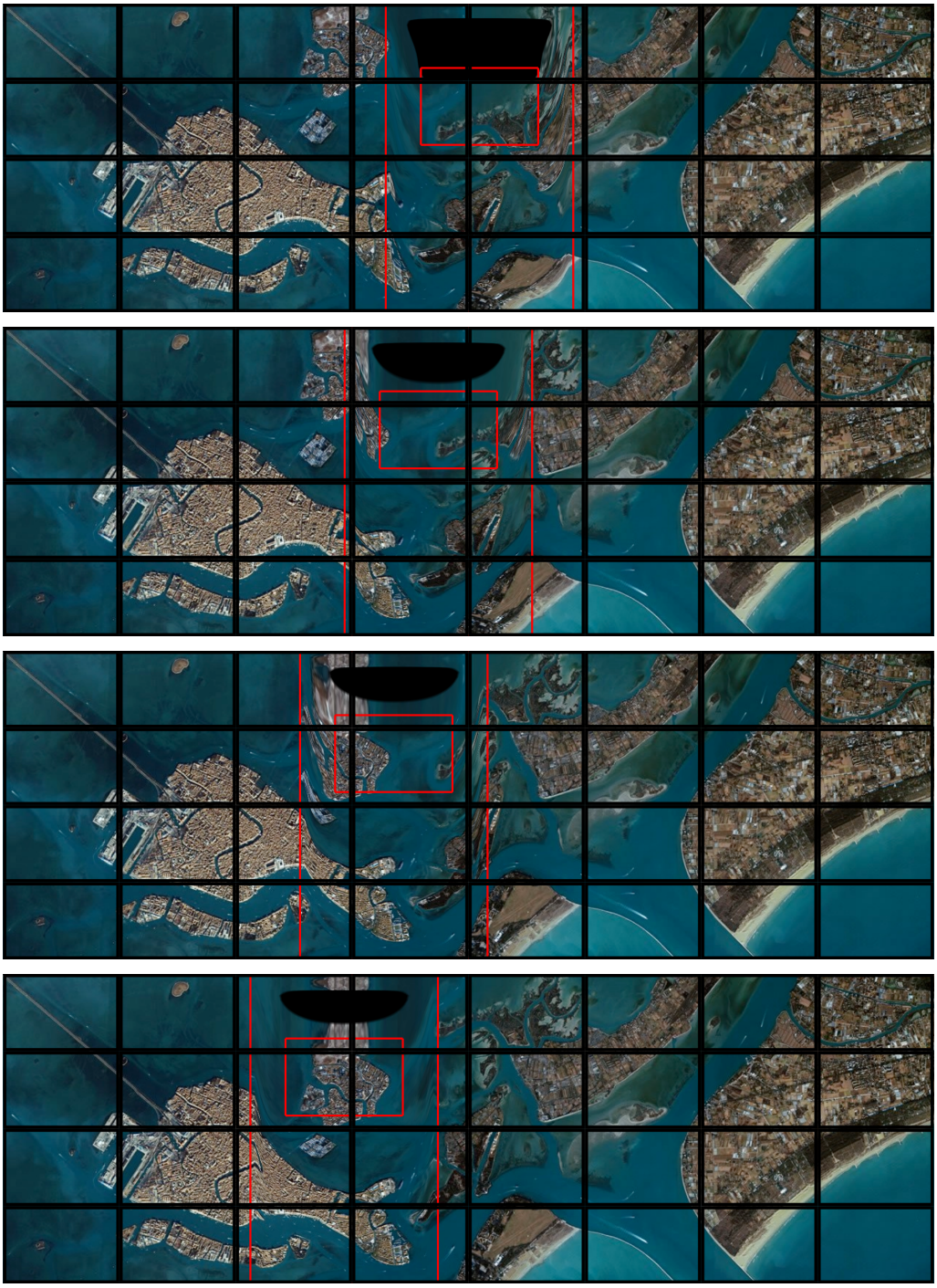


FIGURE 4.4: The top row of a wall-sized display made of several monitors tiled together, is offset down to the row below, to improve its visibility by users. This presentation technique has been implemented with the Jellylens library with less than fifty lines of code.

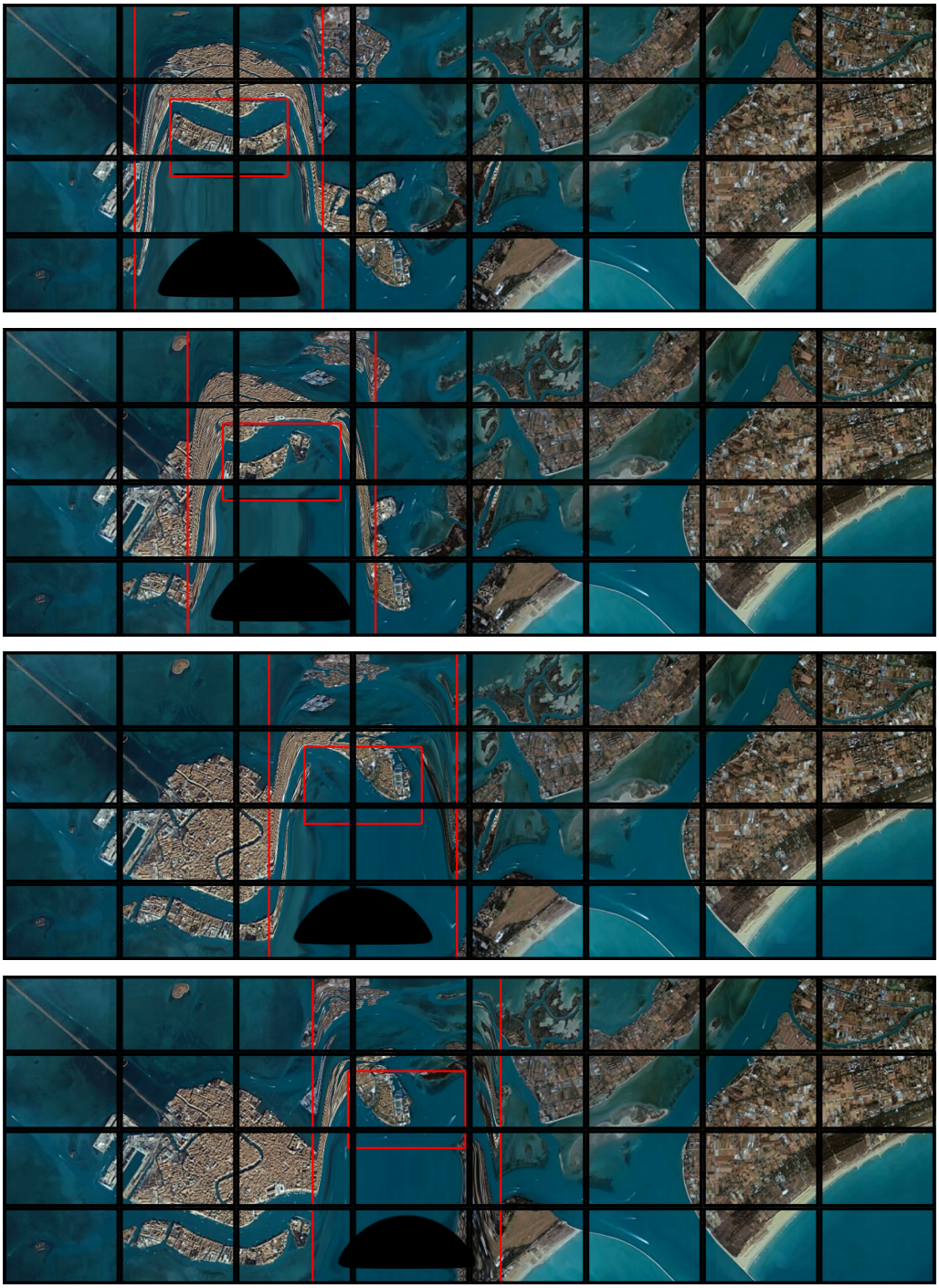


FIGURE 4.5: Pulling up the bottom row by two rows improves its visibility by users who do not need to bend over.

Chapitre 5

Magnifying Lens for Drilling into Complex 3D Scenes

Complex 3D virtual scenes such as CAD models of manufactured objects or representation of the human body are hard to visualize. Those models are made of numerous parts of varying shape, size and geometry, that are clustered closely together and generate a lot of visual occlusion. Indeed, some parts partially occlude, or, sometimes fully surround another, which are then impossible to view from any angle.

Navigating 3D models with conventional zooming interfaces is prone to the "Context lost" problem which leads users to be lost when zooming to much into the model. In this chapter, we introduce Gimlenses (Figure 5.1), detail-in-context visualization technique that enables users to navigate complex 3D models by interactively drilling holes into their outer layers to reveal objects that are buried, possibly deep, into the scene. Those holes get constantly adjusted, depending on the parent view and the current object currently under focus, to guarantee visibility of objects of interest from the parent view.

Gimlenses also provide support for navigation using several lenses allowing to inspect simultaneously remote parts of the scene. And lenses can be cascaded and constrained with respect to one another, providing synchronized, complementary viewpoints of the objects under focus. Gimlenses enable users to quickly identify elements of interest, get detailed views of those elements, relate them, and put them in a broader spatial context.

Gimlenses implement a content-aware design approach : they adapt to the content

of the scene to provide visualizations of higher relevance. Awareness of the objects of interest allows to better tune both visualization and interaction techniques easing the display of objects of interest.

The first section identifies three main challenges entailed by the design of detail-in-context interfaces for navigating 3D scenes : 3D linking – support the understanding of spatial relationships between the magnified and context views ; 3D navigation – allow to quickly select objects to focus-on, adjust zoom-factor and view angle ; and combining multiple lenses – handle lens conflicts and coordinate them to support navigation. We then give an overview of the interface, and address each question in the following sections. We will then conclude with future work and discussion.

5.1 Introduction

Three-dimensional computer modeling has become an essential activity in many domains : the movie industry, medicine, scientific visualization at large, and the automotive, aerospace and electronics industries, which sometimes rely on 3D modeling throughout the design and production chain (Computer-Aided Design and Manufacturing). Processing and graphics rendering capabilities now make it possible to model and visualize extremely complex datasets on widely varying types of displays : desktop workstations featuring one or multiple screens, ultra-high-resolution wall-sized displays [89], and virtual reality environments such as CAVEs [41]. For instance, Boeing's 777 airliner was entirely modeled using CAD software, resulting in a very complex and dense 3D scene featuring more than 350 million polygons [44]. Highly-detailed models of the entire human anatomy, of complex mechanical components and even entire cities are now available, usually for a price given the difficulty of creating such datasets.

As their real-world counterparts, complex 3D models are difficult to inspect, because they are made of numerous distinct parts assembled together into dense scenes that generate a lot of visual occlusion. Constituent elements will often partially occlude or even fully surround one another, and can also have widely varying sizes (e.g., an airplane's wings compared to a bolt). Quickly identifying elements of interest, getting detailed views of those elements from different perspectives, putting them in a broader spatial context, and relating them to other elements are all challenging tasks. Often, several designers will be working collaboratively on these datasets, requiring support for merging their work into a single assembly : detecting and understanding conflicts such as, e.g., intersecting parts due to problems of scaling and positioning.

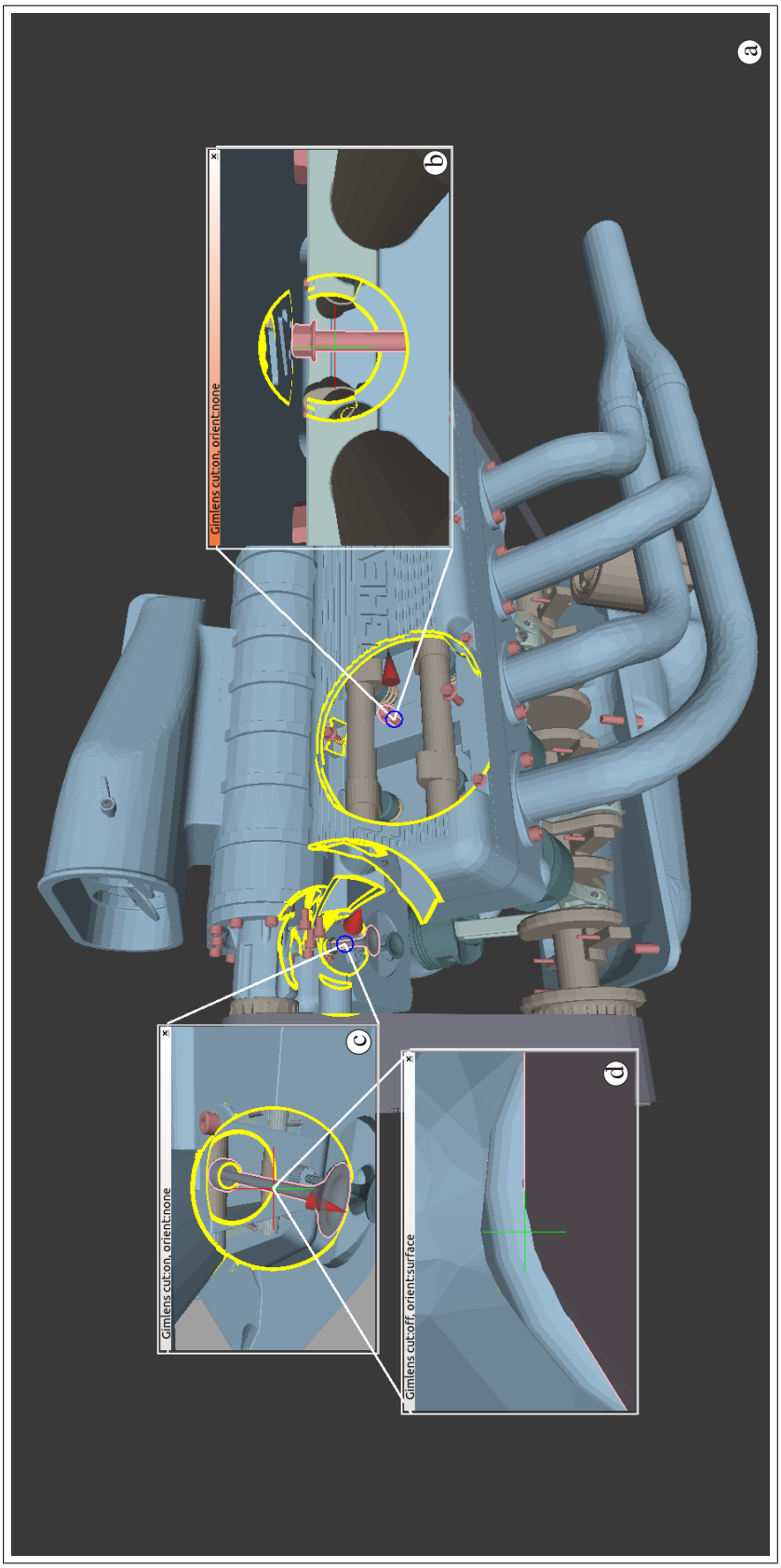


FIGURE 5.1: Exploring the CAD drawing of a car engine. The three Gimlenses provide detailed views of different constituent parts of the engine, at different magnification levels and with varying orientation, while revealing their location inside the global 3D model. a) Context view. b) Magnified side view of a knot behind, and thus originally hidden by, the cylinder head cover. c) View fully revealing a poppet valve in-context from a different angle than the main view, with d) another Gimlens configured so as to provide a low-angled point of view on the valve's surface.

These different tasks involve setting up multiple cameras and manipulating them to inspect elements from different angles and at different scales, as well as changing the visibility settings of elements that lie in the line of sight between the observer and the objects of interest. Those interactions are tedious to perform and cause significant distraction.

Focus-in-context interpretation of the object focused by the lens depends on users' ability to understand spatial relationships between the focus and the context views. This cognitive process, called 3D linking, was identified as one of the main challenges of the design of multi-view interfaces, by Plumlee and Ware[94]. This problem is being further complicated by the visual occlusion generated by the many constituent parts typically found in complex 3D models, which are often clustered closely together. Indeed, most of the time, the part magnified by the lens is occluded by other parts and not visible in the context view, which prevents users from interpreting it with respect to the context view. Gimlenses introduce a new content-aware 3D occlusion management technique, called the *cone-cut* technique, that reveals the object under focus within the context view. The *cone-cut* technique trades the visibility of the focused object against some surrounding contextual information which cant get partially or even fully hidden. Gimlenses rely on a content-aware approach, dynamically adjusting the *cone-cut* technique to the geometry of the objects and the parent view, so as to provide a visualization of higher relevance.

Successful 3D navigating involves allowing the quick identification of an object of interest, putting it under focus and adjusting the zoom level, and camera orientation. Gimlenses draw upon magnification lenses interfaces [134] and provide two methods to control view settings : "through" the lens interaction which allows to inspect focused objects and their surroundings, and "through" the context interaction, where users interact directly within the context view, allowing to switch easily between remote objects of interest. Enabling users to reach for objects buried into the model, Gimlenses introduce a new *drilling* navigation technique which, coupled with the *cone-cut* technique, allows to drill holes to put in focus the successive underlying objects. The *drilling* technique eases depth navigation by implementing the content-aware approach. It adjusts the lens' focus point's depth depending on objects' geometry, and makes the focus point switch from one object to the one beneath.

Providing support for multiple lenses has benefits for comparison tasks[96] and collaborative navigation. However, the 3D occlusion management technique impairs context representation and generates conflict when two lenses come in the vicinity of each other. Lenses also provide support for combining complemen-

tary views on a focused object. Gimlenses thus allows to cascade and constrain the lens' orientation to automatically adapt it when users move the lens' focus point. We identified four types of constrains, that prove useful for distinct scenarios. The constraints define different frames of reference for the orientation, that depends on the focused object's geometry such as the surface normal, or parent view setting such as the view direction.

We will first give an overview of related work before describing Gimlenses' main interface. We then discuss how Gimlenses addresses each one of theses challenges.

5.2 Related Work

Gimlenses build upon work in multi-scale visualization, 3D navigation and smart visibility techniques for CAD.

5.2.1 Multi-scale Visualization

Gimlenses are essentially detail-in-context representations, providing multiple simultaneous points of view on the 3D model at different scales. Gimlenses are categorized as an overview+detail technique [35] : they relate one or more detailed views, called focus regions, into a larger, less detailed contextual view of the dataset. Among the multi-scale visualization technique reviewed in Section 2.2, Gimlenses are more specifically drawing upon the DragMag [134], a technique that offsets the magnified view and visually relates it to the corresponding region in the context view using simple line segments.

In the realm of 3D multi-scale visualization, Carpendale *et al.* [31] transposed the general focus+context approach [35] based on spatial distortion to 3D data cubes. The technique guarantees the visibility of particular cells in a cube by moving and resizing the surrounding ones. The concept was generalized to arbitrary meshes using an energy grid optimization model [130]. The technique only magnifies objects in-place, though, and does not address the problem of visibility in dense scenes. Magic Volume Lenses [128] provide in-place magnification and can reveal some details hidden behind outer layers, but the detailed view is necessarily oriented according to the context view, and magnification factors are severely limited by both the spatially-distorted transition and problems of quantization [3] due to the single focus region. The DragMag technique mentioned earlier was also transposed to 3D in [95], providing users with a multi-window detail-in-context visualization technique that somewhat resembles ours. However, the technique was designed for navigating in oceanic data, i.e., relatively flat

3D scenes compared to car engines or layered models of the human anatomy. Thus the technique does not deal with problems of visual occlusion typically encountered in complex, dense 3D scenes.

Magic Lenses [17] are lenses that are used to modify the rendering of objects seen through them. The concept is very generic and powerful. Examples more closely related to our work are lenses that render, e.g., a wireframe version of the original object. Originally designed for 2D graphics, Magic Lenses were later extended to work with 3D graphics [124, 101].

5.2.2 3D Navigation

Numerous techniques have been designed to make navigation into 3D environments more efficient than when using the usual pan, zoom and rotate functions found in most 3D user interfaces, using different metaphors such as that of a flying vehicle. [113] combines speed-coupled flying with orbiting. McCrae *et al.* [85] adapt the travel speed of the flying vehicle in a space-scale-aware manner using a cubemap technique to enable navigation in multi-scale environments. Navidget [58] is a gesture-based interactive widget that enables users to get a preview from a different perspective on the scene before actually repositioning and reorienting the main camera to match this new point of view. Another way to ease navigation is to constrain camera movements to objects dependent surfaces [71] or authored surfaces [27].

5.2.3 Smart Visibility

Smart Visibility originates from static technical illustrations [125]. It aims at conveying the most information possible using a single, often static illustration by unveiling its most important parts. Smart visibility techniques are based on cut-away views [48, 79], ghosted-views [25, 126, 72, 38, 39] or spatial rearrangement [78, 87]. Burns *et al.* [26] propose adaptive cutaways that adapt their geometry to guarantee the visibility of predefined parts of interest. While some of these techniques do make it possible to interactively select the main point of interest to be revealed [79, 78], these are still aimed at producing static illustrations. They do not provide a true solution for exploring complex 3D scenes.

5.2.4 3D Occlusion Management

Gimlenses are conceptually related to the 3D occlusion management techniques surveyed in [46], and more specifically to the x-ray category of tools : perspective cutout [37], x-ray tunnels [6], and the looking glass from [80]. Such tech-

niques also provide solutions for revealing hidden parts of a 3D scene, as do Gimlenses. However, those techniques are used *in addition* to a 3D navigation technique, which increases the burden put on users, who have to handle both navigation and visual occlusion management conjointly but sequentially. On the contrary, Gimlenses integrate occlusion management with 3D navigation, automatically revealing the focused part of the scene to the observer. This integration facilitates exploration no matter the type of display, but is especially useful in contexts where input capabilities are limited or where interaction is more challenging than on a desktop workstation such as, e.g., wall-sized displays operated from a distance or via a touch-sensitive surface. Another distinctive feature of Gimlenses compared to earlier work are their capacity to get coordinated and cascaded, as detailed in Section 5.6.

5.3 Interface Overview

Gimlenses are magnification lenses. They provide one or more detailed views, called focus regions, into a larger, less detailed contextual view of the dataset. This presentation space, called the *context view* (Figures 5.1-a and 5.2-a), provides conventional navigation techniques, pan, zoom and rotation, to adjust the view frustum and fit the whole scene into the view. Users can freely rotate around the model and zoom into the data as needed.

Once they have adjusted the *context view*, users can instantiate one or several Gimlenses to inspect details of the scene, displaying magnified representations within the *view window* of the lens. The *view window* (Figure 5.2-b) is a viewport that users can freely resize and move. Usually, the *view window* will be adjusted so as no to occlude, within the *context view*, the region to be inspected. Each Gimlens is also composed of three other main elements : the *object selector*, the *view proxy* and the *cone-cut*.

The *object selector*, is a ring (in blue in Figure 5.2-c), that both identifies within the *context view* the current object under focus, and, shows precisely the location of the focus point lying over this object. A Gimlens visually links together its *object selector* and its *view window* with two lines, called tethers. Such interface elements were introduced by Ware and Lewis in the *DragMag* magnification technique for 2D visualization [134].

The *view proxy* is a cone (in green in Figure-5.2-d) orbiting around the *object selector* that indicates from what angle is the focused object being viewed by the Gimlens. The cone orientates itself such that its axis aligns with the Gimlens' line of sight, its tip pointing at the observer. The cone is rendered as an opaque

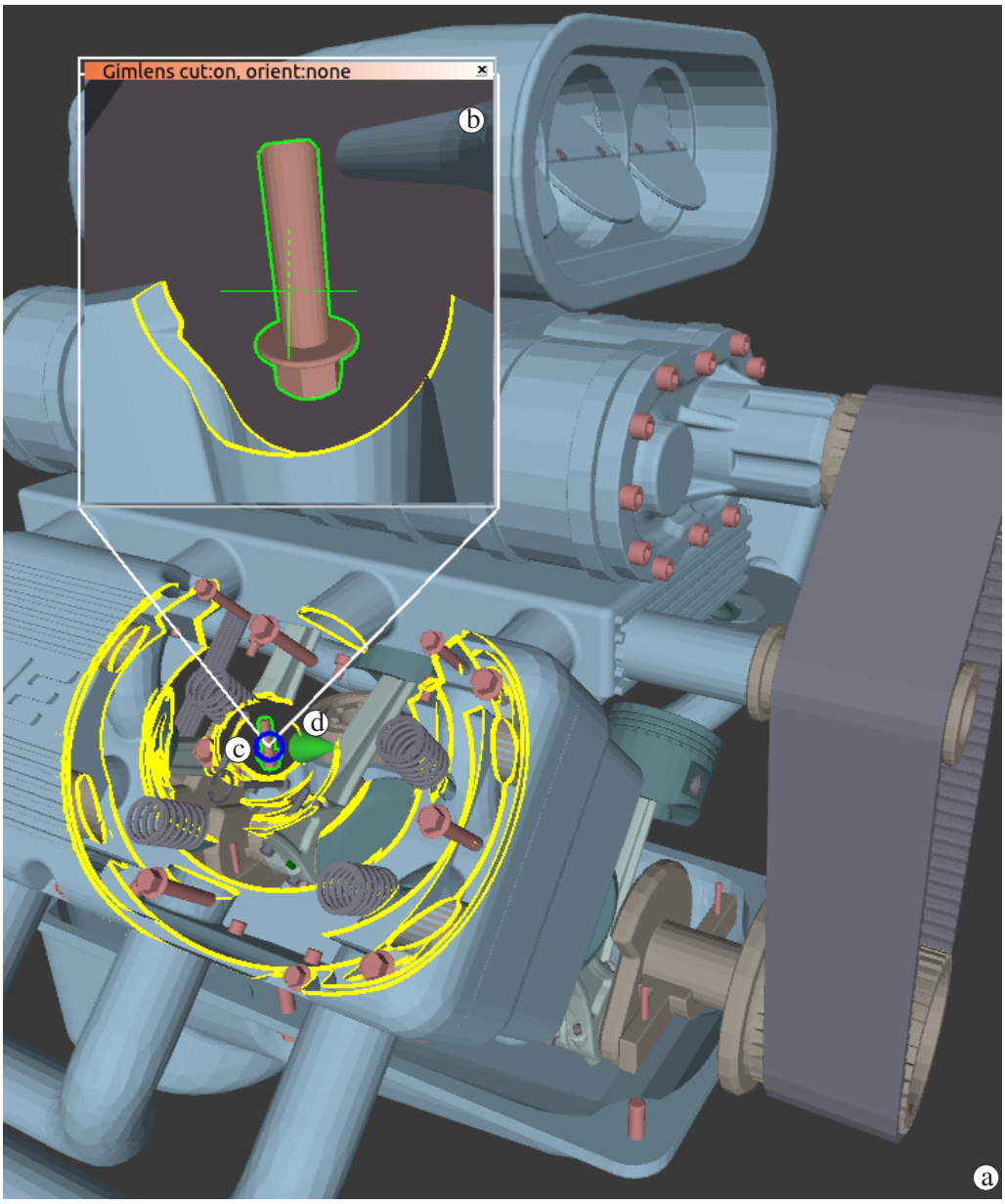


FIGURE 5.2: A Gimlens allows to magnify parts buried deep inside the model, here a knot inside a car engine. The main interface component are : the context view (a), the Gimlens view window (b), the object selector (c), and the view proxy (d).

object, with a smooth Lambertian shading effect to better convey the direction information. To help evaluate its position within the scene, the cone is rendered with its occluded part slightly blended with the occluding objects, as if it were translucent (as illustrated in Figure-5.1-c (red cone)).

We did not introduce dedicated interface elements to convey information about

the distance between Gimlens' viewpoint and the focused object. Designing the interface, we first mapped the *view proxy* to the location of the viewpoint, as in the GeoZUI interface designed by Plumlee and Ware[95]. However, constantly changing its distance to the focused object, the *view proxy* was less easy to find within the scene, which impeded the readability of the interface. The problem was further complicated when several lenses were populating the scene. Eventually, we discarded this feature. Although other visual cues could have been implemented, such as a slider showing the distance to the focused object, we thought it was causing visual overload for no clear added value. Contrary to orientation, distance to the focused object is not a critical information for relating magnified and context objects. Indeed, when seen from different angles, the aspect of an object can vary a lot, while, it does not when distance varies. Actually comparing the size of both representation of an objects is sufficient to understand the distance.

This set of interface elements is sufficient to relate the content of the *view window* to that of the *context view*, as long as the focused object is visible within the *context view* [95]. Gimlens features another main element : the *cone cut*, which is a cut shaped as a cone, whose purpose is to reveal the Gimlens' focused object within the *context view* by removing portions of occluding objects found within the cone. The cuts generated by a *cone-cut* are outlined to ease their understanding (Figure 5.2, outlined yellow cuts into the cylinder head cover and beyond). The *cone-cut* technique is further presented in Section 5.4.1.

To improve the overall readability of this relatively complex representation, the object of interest is further emphasized by outlining it in both the *view window* and the *context view* (knot with green outline, Figure 5.2-b and 5.2-c).

Please note that the colors of the various interface elements vary between illustrations. We adjusted them so that they better contrast with the colors used in the underlying scene.

5.4 Focused Object Occlusion Management

Gimlens integrates a detailed view into a larger less detailed contextual view of a 3D scene. Focus-in-context interpretations of the content magnified by a Gimlens, depend on users' ability to understand spatial relationships between the focus and the context views. The basic interface elements we introduced in the previous section successfully support linking for sparse scenes, i.e., when the focused object is always visible in the *context view*. To address the linking problem in dense, highly occluded scenes, Gimlens integrates a 3D occlusion

management technique, that reveals the focused object within the *context view*, by removing portions of the occluding object falling inside a cone.

We present in this section the *cone-cut* technique. How to change what object is in focus and select among the scene's objects is detailed in the next section.

5.4.1 Object-Dependent Cone Cut Technique

The design of an occlusion management technique for the purpose of detail-in-context visualization involves finding a balance between the visibility of the focused object and some compensatory elision. Revealing an object within a 3D scene typically involves adjusting the visibility settings of the occluding objects, which leads to losing some contextual information. Some conventional techniques implemented in 3D CAD viewing applications hide objects with respect to their relative position to a plane (clipping plane), that users position within the scene. Other techniques let users toggle visibility of each object. Although they successfully reveal objects of interest, such techniques typically remove important amount of the content of the scene, impeding seriously interpretation of the scene. Indeed, in case of clipping planes, revealing an objects in the middle of the model, typically entail hiding half of the model. The visibility of objects buried into the scene, is obtained at the cost of removing some contextual information, so as to clear a line of sight to the focused object.

As the occlusion generated by each object depends on its size and shape, we adopted a content-aware approach, where the parameters of the cut adjust to both the geometry of the focused object and to the geometry of the occluding objects.

The Cone

The cone, illustrated in Figure 5.3, is define implicitly as a piecewise function depending on the position of the point M being processed in eye-space coordinates. With p the position of the focus point, we define x, the projection of M on the cone's axis, and y, its distance to the cone's axis, as follows :

$$x = \frac{dot(M,p)}{\|p\|},\ y = \frac{\|M \times p\|}{\|p\|},$$

with $dat(M,p)$ being the dot product between both vectors. The near base, with a radius of S, controls the wideness of the aperture. We note n the distance from the near base to the camera's position. The far base is positioned at the focus point. f, its distance from the observer in eye space coordinates, is equal to $\|p\|$.

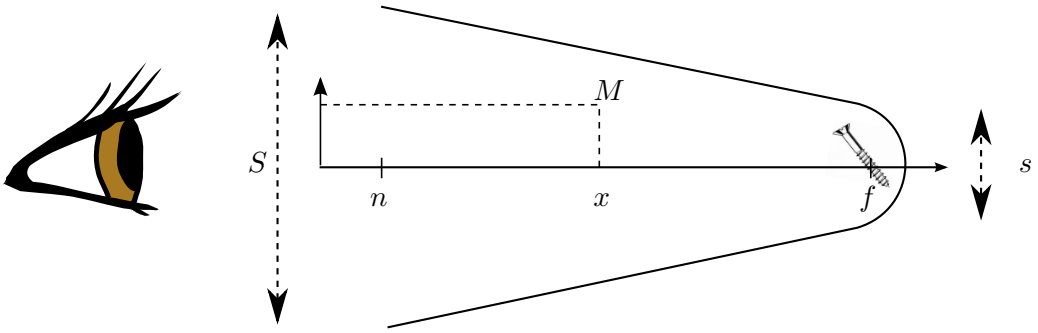


FIGURE 5.3: The *cone-cut* is a 3D occlusion management technique that is designed to reveal the object currently under the focus of a Gimlens. Its shape is defined implicitly by a piecewise function, delimiting a cone ended by a sphere to avoid any sharp edges. Every region of objects falling within the cone are hidden.

In front of the cone : if $x < n$.
From the users' viewpoint, the base of the cone delimits the shape of the cut. Hence, a point gets discarded when its distance to the cone's axis is greater than the radius of the front base. In eye-space coordinates, the cone appears to extend to the front by another cone shrinking up to the position of the eye. A point is discarded if :

$$y < S * x/n.$$

Between the front and the far base of the cone : if $n < x < f$.
This is the actual body of the cut shape as an inverted-cone. A point gets discarded if :

$$y < (x - n) * \frac{s}{f - n} + (f - x) * \frac{S}{f - n}$$

Behind the far base : if $x < f$.
To avoid sharp edges, that could be confused with actual edges in the model, the cone ends as a sphere that fits the far base. A point behind the far base gets discarded if :

$$\|(x - (f - s * \frac{(S - s)}{f - n}), y)\| < s * \|(\frac{(S - s)}{f - n}, 1)\|$$

Adjusting the Front and Far Bases

Adjusting the front and far bases controls the size of the cone, and allows to balance between focused object visibility and some compensatory information elision. Typically, enlarging the front base S favors the visibility of what is inside the cut, but more of the model gets hidden. Users can control this parameter,

but we have found a base radius that corresponds to roughly 25% of the main viewport's size to yield a good compromise. The front base is placed at the same distance to the eye than the distance to the first intersection so as to be laying on the first layer of the model.

The cone does not need to reveal more than the focused object. Hence, the radius of the far base s, gets adjusted so that the object of interest can fully fit within the cone. However, some of the objects could be larger than the front base. To prevent degeneration of the cone, we constrain the size of the far base to be smaller than that of the front base. Hence for large objects, the *cone-cut* only reveals a portion of them. Another solution would have been to enlarge the front base to fit the whole object, which would have caused severe context loss in recurring situations.

Degeneration of the cone occurs as well when the focused object is both small and close to the front base. The cone's angle reaches too high values, causing unpleasant visual artifact. In this case, we found it more convenient to enlarge the cone, revealing more information to avoid such visual artifacts. We then also constrain the cone's angle to be lower than a certain value (we found that $20°$ was convenient).

Finally, small parts that lie within the cut cone but are not causing significant obstruction should not be hidden, as they will often provide interesting contextual information (see Figure 5.2). Hence, the *cone-cut* also adapts to the size of occluding objects. It provides another cone, with smaller base radius, that cuts little holes when a small object is overlapping the focused object. The occlusion management technique is made of two *cone-cuts*. A large one for large objects that cause significant visual occlusion, and another one, smaller, to drill small holes into small objects. To discriminate between the large objects and the small ones, we compare their projected screen sizes (the size of their image on the screen) to that of the focused object.

Depth Cues

In addition to revealing the focused object, the *cone-cut* technique conveys visual cues about the depth of the objects. Indeed, thanks to the cone shape, the strokes caused by the cut are visible to the user, who can then see how many layers have been cut to reveal the object. This information allows users to roughly understand how deep inside the model the object is. Distance between each strokes also conveys information on the depth distribution of the various layers. We decided to emphasized this effect by drawing the strokes.

5.4.2 Adaptation of The Cone's Shape

Before opting for a truncated cone shape, we experimented with a content-aware dynamically-adapting cut shape similar to the technique presented in [26]. The cut was adapting to the geometry of the Gimlens' current object of interest. Our intuition was that by adapting the shape of the cut to the shape of the content, we would optimize the portion of the scene to be cut, thus preserving more context information, in the same spirit as the Pathlenses we presented in Chapter 2, which dynamically adapt the shape of a fisheye to optimize the focus, context and distorted regions to provide more relevant focus+context representations.

However, an informal evaluation performed in our laboratory revealed that users preferred to interact with cuts that always had a cone-based shape. The cut shape radically changing its geometry depending on the geometric characteristics of the various objects of interest was found to be disturbing, as it was both unstable and unpredictable.

Moreover, depending on the shape of the focused object, the cut could induce sharp edges in the scene, that were misinterpreted by users as sharp edges in the objects.

5.4.3 Discussion

As users explore the model, they inspect successively the objects of the scene. Switching from one object to another one, the *cone-cut* technique adapts automatically to provide better balance between focused object visibility and context information. This adaptation results in animations that could possibly distract users from their original goal. We tested two approaches, the first involving changes in the size of a cut shaped as cone, and the other changes in the shape of the cut. The second approach could presumably provide better visualization, as the shape was optimized to reveal exactly the object of interest, and doing so better preserves context information. However, distracting effects caused by the animation were significantly different. We thus opted for the first approach.

This design choice points out a characteristic of the content-aware design approach. If adaptation of the interface is meant to optimize the visualization depending on the content being displayed, this adaptation result in some necessary animations which could distract users and lower efficiency of the interface. The interface proposed results from trying to strike a balance between optimal adaptation and the resulting distracting animation.

Although, we originally designed the *cone-cut* technique to reveal the focused

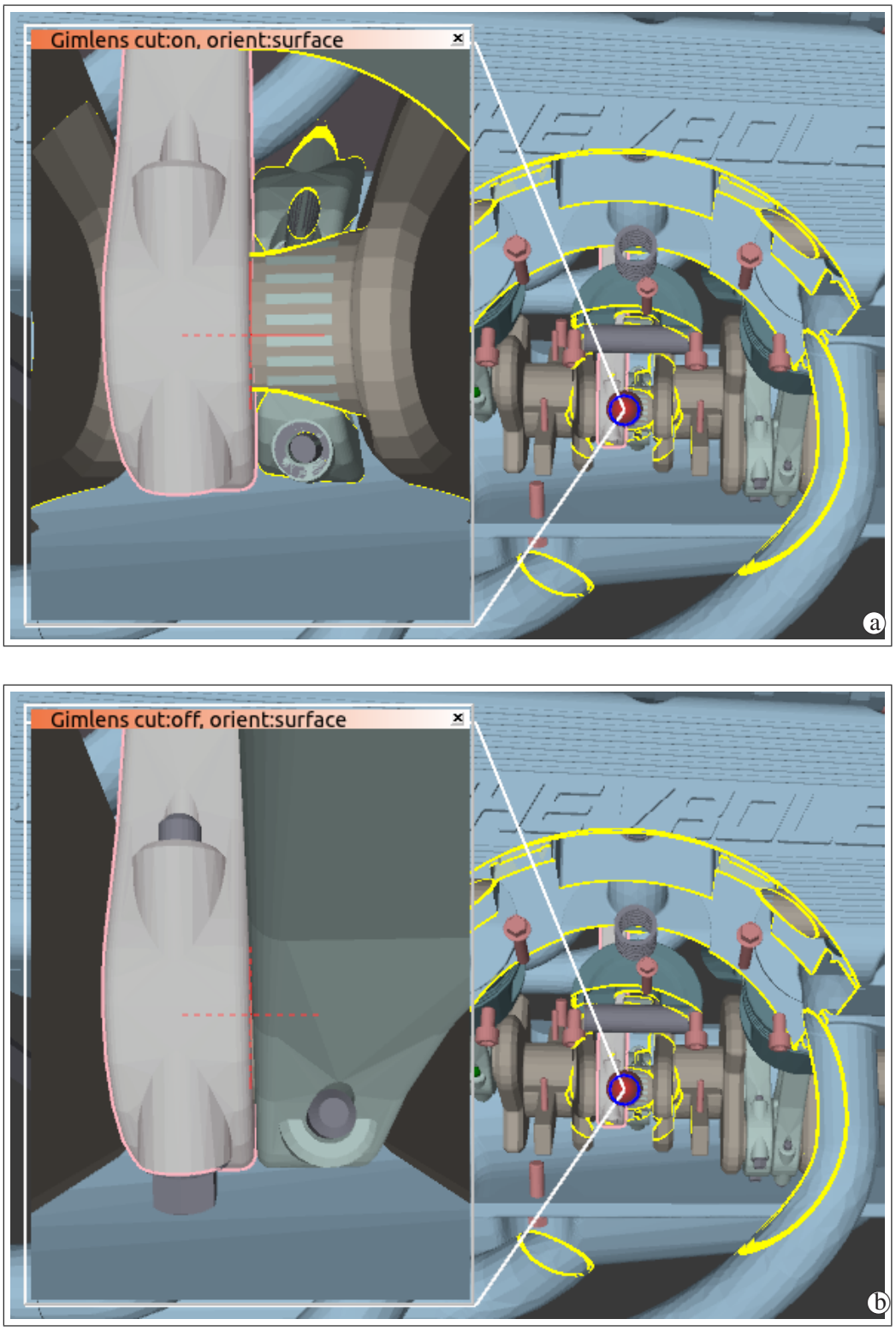


FIGURE 5.4: *Cone-cut* might sometimes hide information important to users.are might sometime hide A Gimlens used to check the surface intersection between a connecting-rod and a camshaft. (a) Enabling the cut reveals the object of interest during exploration but hides its neighbors. (b) Disabling the cut enables users to see both parts.

object in the context view, we found that it could be useful as well in the *view window*. As users move the focus of attention in the context view, it might very well happen as well that the object of interest is hidden in the focus view, requiring 3D occlusion management capabilities. We then implemented the *cone-cut* technique as well in the *view window*, coupling the near base to the apex of the lens' viewing frustum so that the object of interest always remains visible no matter how the former gets positioned. However, this behavior is not always desirable. For instance, when instantiating a Gimlens to inspect the intersection between the surfaces of two distinct parts in the model, the cut cone would hide one of those surfaces, hence hiding an important piece of information, as illustrated in Figure 5.4. Cuts performed inside the lens' view window can thus be toggled on/off.

5.5 3D Navigation

The goal of navigation is to help users explore complex 3D models. The interface must provide controls for focusing successively on the various objects of the scene, adjust the view angle and the zoom level. Users switch from one control to the next in a seamless manner, without being distracted from their primary goal.

Complex 3D scenes are difficult to explore because many objects are buried deep into the model and are not visible under any view angle. We introduce a set of navigation techniques which build upon the 3D occlusion management technique we introduced previously, that enables the exploration of such environments.

Gimlens, inspired by the DragMag technique [134], implements two different navigation methods. The first consists of interacting "through" the lens, and allows inspection of the focused object and its surroundings. The second one is the interaction "through" the context view, allowing users to directly point the context view to the object they want to focus on. To avoid constant switching between the *view window* and the *context view*, we also implemented interaction allowing to control Gimlens' orientation directly within the context. Both interaction methods are complementary and we observed users making intensive use of both during informal evaluation of the prototype.

A typical scenario involves first, interaction "through" the context to roughly explore the scene. Then, once an object of interest has been identified, users set it under focus, and then switch to the *view window* to proceed to the inspection of the details displayed by the lens.

5.5.1 Navigating *through* the Lens

Users can freely reconfigure the Gimlens' viewing frustum by interacting within the lens' *view window*. Controls should be mapped to the input device provided by the visualization platform (desktop, wall sized display or immersive environment). For a conventional implementation for desktop applications, pan, rotation and zoom to dragging gestures over the *view window*, assigning each mode to a mouse button. Mode switching could also be implemented, as in some 3D modeling software to some keyboard shortcuts.

During navigation, as users zoom, orbit and switch between neighboring objects, Gimlens other interface elements automatically update to help users relate the magnified content displayed by the lens to the context's point of view. The positions of the *object selector* and the *view proxy* are updated accordingly, representing Gimlens' view configuration. If users reach an object that is not visible in the context view, the *cone-cut* automatically adjusts to reveal the focused object.

While for now we only support conventional panning, zooming and rotating, we believe Gimlens could benefit form adding other navigation techniques, such as Hovercam, allowing to orbit around a model at a constant distance, which proved convenient for close inspection of models. However, it is not clear yet how to provide users with the possibility to switch between objects of interest with this technique.

This first method of navigation allows fine configuration of the Gimlens' view frustum, allowing to precisely adjust the content being magnified by the lens. However, it is not suited to the exploration of large models.

5.5.2 Navigating *through* the Context

Users can interact directly within the context view. This method is convenient for switching between remote objects in the scene. It allows to make big jumps from one part of the scene to another.

Users can grab the *object selector* and drag it in the *context view* causing the Gimlens' focus point to slide on the surface of the model, following the *object selector*. While moving the *selector*, the focus is given to the object currently falling under the object selector.

Gimlens adjusts the content displayed in the *view window*, following the focus point. The orientation of the lens is also adjusted depending on some properties detailed later. The default behavior is to keep the same orientation as the focus point or the focused object gets updated.

When users make the *object selector* leave the scene (no other object falls under the *object selector*), it does not control any more the Gimlens' focus point, until it re-enters the scene.

Users can also grab the *view proxy* and drag it in the *context view* to adjust the orientation of the lens. The *view proxy* orbits around the focus point at a certain radius. Thus, when users move the *view proxy* within a disk around the *object selector* that is the size of this radius, the position of the *view proxy* is mapped to the projection of the selector on the sphere. Leaving and re-entering this sphere causes the *view proxy* to slide to the other side of the sphere and allows to inspect the back side of the focused object. This interaction was inspired by a similar interaction introduced in Navidget [58].

In our prototype we did not implement controls for the zoom factor from within the *context view*. Mapping this control to the mouse wheel would have been a straightforward solution, however we already use this input channel for controlling the *drilling* technique described in the next section.

All the previous techniques altogether only allow for navigating on the first layer of the model. Many objects, are enclosed inside the model and most of them are not visible from any angles without drilling into the model.

5.5.3 Drilling Technique

We introduce the *drilling* technique, providing support for depth navigation in the model. Drilling into the model allows for navigating through the several depth layers of the scene. Invoking the technique on the current part of interest (on which the *selector* lies) cuts out a hole in it and moves the focus point on the surface below, that was just revealed by the cut. Users can also climb back, filling the holes previously cut in reverse order, with the focus point and object of interest getting updated accordingly.

Gimlenses keep a history of what parts got cut, allowing the user to drag the focus point from one part of interest to the next and drill down iteratively (Figure 5.5 illustrates the overall process). This navigation method can be seen as an easy way of toggling the visibility of the successive parts that the focus point falls on during the drilling process, except that each part's visibility is only affected locally around the line of sight so as to better preserve context.

During navigation, the *cone-cut* automatically adjusts its geometry to reveal the current focused object, adapting to its size and position, following the focus point controlled with the *selector*.

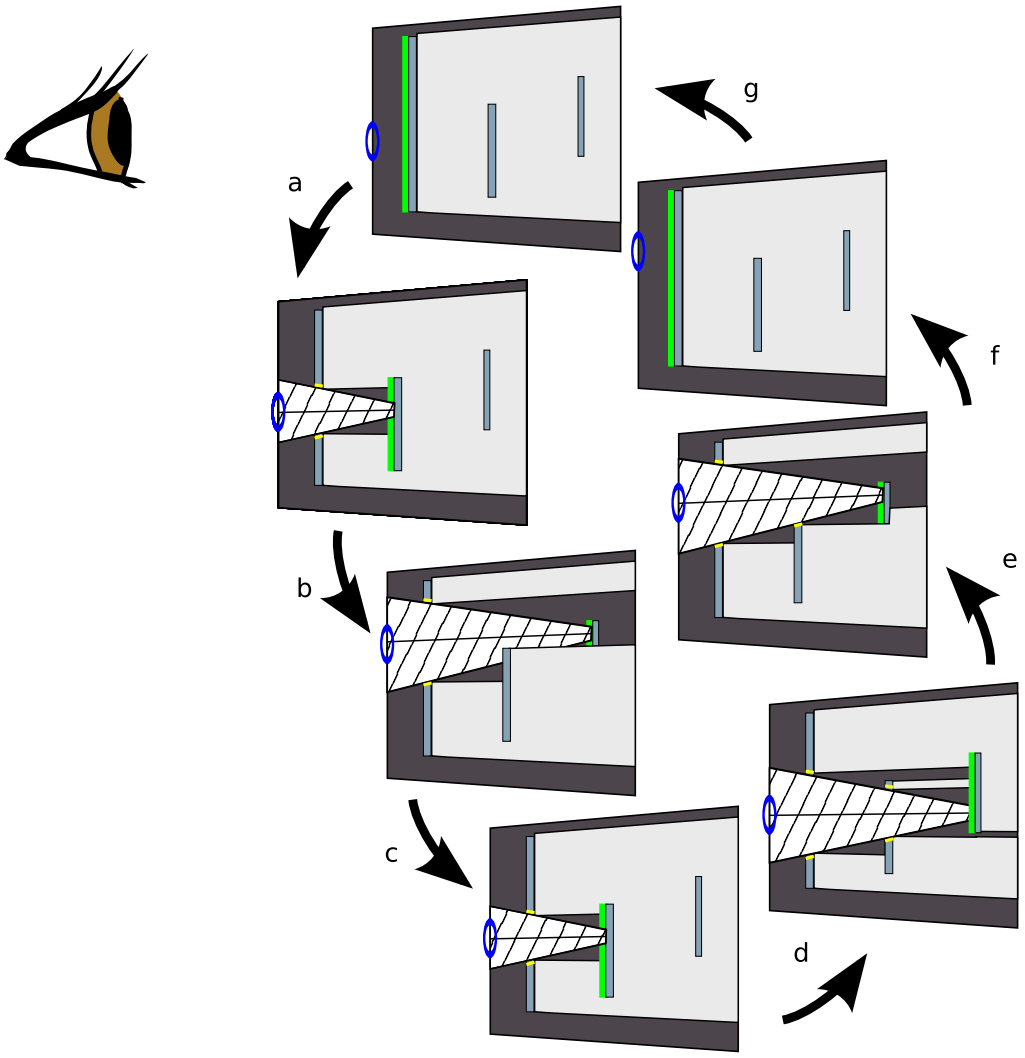


FIGURE 5.5: Side view illustrations of how the drilling technique behaves. The focus point lies on the frontmost part, which is then considered the focused object (green). (a) This part gets drilled into (in yellow the strokes generated by the cut) , revealing the other part behind it, which becomes the focused object. (b) Users drag the *object selector* (blue ring) on the screen, which slides the focus point up to the next visible part, which in turn becomes the focused object, (c) and drags the *object selector* back to the previous object. (d) Users cut through the second layer, (e) slide the focused object, (f) go back to the original object (g) and go back to original position.

On a desktop workstation, the technique is operated with the mouse wheel. In other environments, such as virtual reality platforms or wall displays, the corresponding actions have to be mapped to a linear continuous input control channel on one of the available interaction devices.

Originally designed for selecting objects in the context view, the *drilling* tech-

nique is actually useful in the focus view as well. However, it works slightly differently. Instead of being controlled by a ring (the *object selector* in the context view), the focused point is controlled by a sight situated at the center of the *view window* (the green cross in Figures 5.1 and 5.2 ; and red cross in Figures 5.4,5.7 and 5.8). When the lens' view get readjusted, the position of the focus point get adjusted in the model to be aligned with the sight of the lens, *drilling* cut the object beneath the sight.

5.5.4 Discussions

We designed a navigation technique that allows to drill into dense, highly occluded scenes. Gimlenses allow either to interact directly with the lens content, "through" the lens viewport, or with its proxy in the context view. This enables users to benefit fully from context information. Users can drill into the model, cut holes on the successive parts, revealing the part below, to inspect objects buried into the model.

A key point of this navigation technique is that it indexes the lens view configuration by objects of interest instead of view point. Once an object has been identified as of interest by users, they directly point at it to see it magnified by the lens. Instead, with conventional 3D navigation techniques, they have to figure out how to adjust panning, rotation and zooming, to eventually view the object magnified, which can be very tedious.

Pointing at an object of interest with a Gimlens is made particularly easy by the content-aware approach of the *drilling* technique. The *drilling* technique brings 3D navigation back to a 2D interaction task. Users do not need to adjust the focus point depth component. Instead, the scene is considered as stacks of 2D layers, and users only control which layers the focus point slides on. This results from the conjunction of a depth navigation technique with a content-aware approach. Instead of providing linear controls for the depth dimension of the focus point, the *drilling* technique adjusts the step at each increment of the depth component, allowing to jump directly from one object to the next below. Hence, this technique depends on proper segmentation of the scene into distinct parts, and is not suited to unstructured scenes such as sets of unorganized triangles. We discuss this issue in more detail in the Section 5.7.

5.6 Combining and Cascading Lenses

5.6.1 Handling Cone-cut Conflicts

Multiple Gimlenses can be instantiated simultaneously to provide users with different perspectives on the 3D model : different locations, different scales, and different angles. This is useful, e.g., when comparing multiple parts of a model, such as two knots inside an engine. For instance, Gimlenses can be created for both knots, the user then drilling towards each knot, adjusting the view and then orbiting these two objects of interest.

Gimlenses can also be cascaded, in a way somewhat similar to what PolyZoom does for 2D maps [66]. Cascaded lenses share the same focus point. They enable users to easily get multiple views on the same object of interest at varying scales and from different angles. Figure 5.8 gives an example. A user wishes to inspect a steal pipe in an air distillation plant. She cascades several Gimlenses, that give her points of view on the region of interest at different scales.

Multiple Gimlenses reveal as many objects in the context view which can generate conflict when the *cone-cut* are in the vicinity of one another. Indeed, one *cone-cut* is likely to hide the very object another Gimlens is focusing on. To resolve this issue, our technique renders a circular portion of the object that is the source of the conflict around the focus point using alpha blending to make it translucent. The part is thus revealed, without hiding the second focus point. See details in Figure 5.6 and the overall interface in Figures 5.9, 5.10, 5.11 and 5.12.

Although it has been reported that the use of semi-transparency in visualization may result in “high visual complexity and imposes a high cognitive load on the user” [46], it is still of interest for revealing only one object. The complexity generated by semi-transparency depends on the context, increasing with the number of layers. In our case, it is only used to reveal one layer at a time, keeping a low impact on user cognition, and resolving efficiently the *cone-cut* conflicts.

5.6.2 Cascading Gimlenses

Inspecting 3D objects involves viewing them from varying viewpoints. Users can cascade several Gimlenses, focusing each one on the same point or object. Any readjustment of the focus point or switching of focused object, are transmitted to every cascaded Gimlenses, sparing users the burden of readjusting each lens one by one.

Cascaded lenses let users adjust view angle and zoom level independently for each lens, allowing users to combine different complementary viewpoints. Which

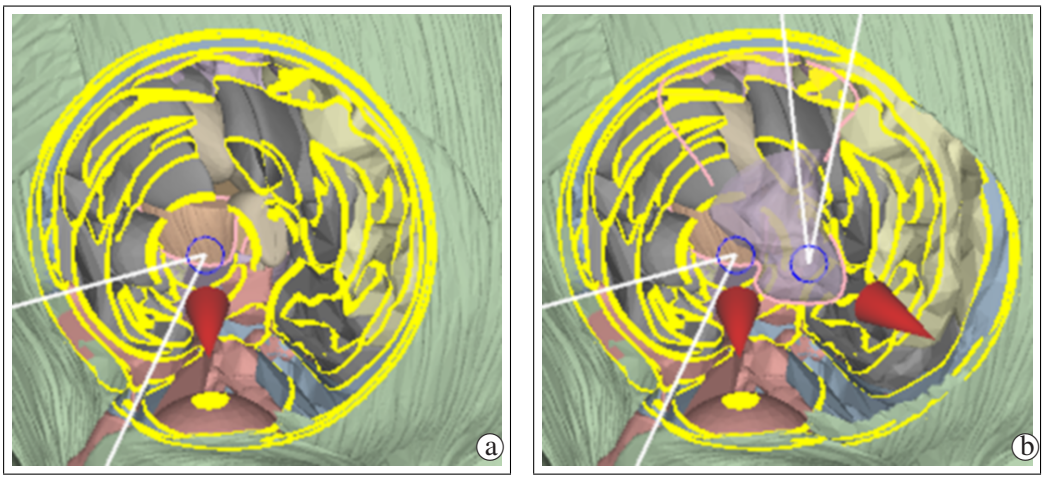


FIGURE 5.6: Exploring a brain. (a) A first Gimlens gets instantiated and delves deep inside the head, hiding several objects in the process. (b) A second Gimlens gets instantiated to look at an object that lies in the cut cone of the first lens, causing a conflict. The portion of this object that lies near the focus point of the second lens gets rendered translucently.

configurations users need depends on their goal. For instance, if users are looking for potential intersections between two objects, they would likely set one view from above the surface, with another one with a shallow angle relative to the surface to observe the interface between surfaces. Although both lenses will follow the focus point, when users proceed to the inspection, they will require readjustment of their orientations as surfaces change.

To spare users the burden of constantly readjusting each lens orientation, Gimlenses let users choose between several orientation constraints for each lens, that will readjust automatically the view angle during navigation. The choice of orientation constrain depends on the scenario. For the intersection inspection scenario introduced previously, users could choose the **Surface** constraint for both Gimlenses, which would automatically update the lens orientation so as to maintain the same relative view angle at the focus point with respect to the object's surface normal. With the **Surface** constraint, the first lens will maintain a view from above on the surface, and the second a shallow angle as users move the focus point on the surface.

As the focus point jumps from one polygon to the next, **Surface** orientation constraints might cause the view in the lens to change abruptly. To avoid this, we smoothly animate the transition over 100ms.

We define three orientation constrain that enable different behaviors. Figure 5.7

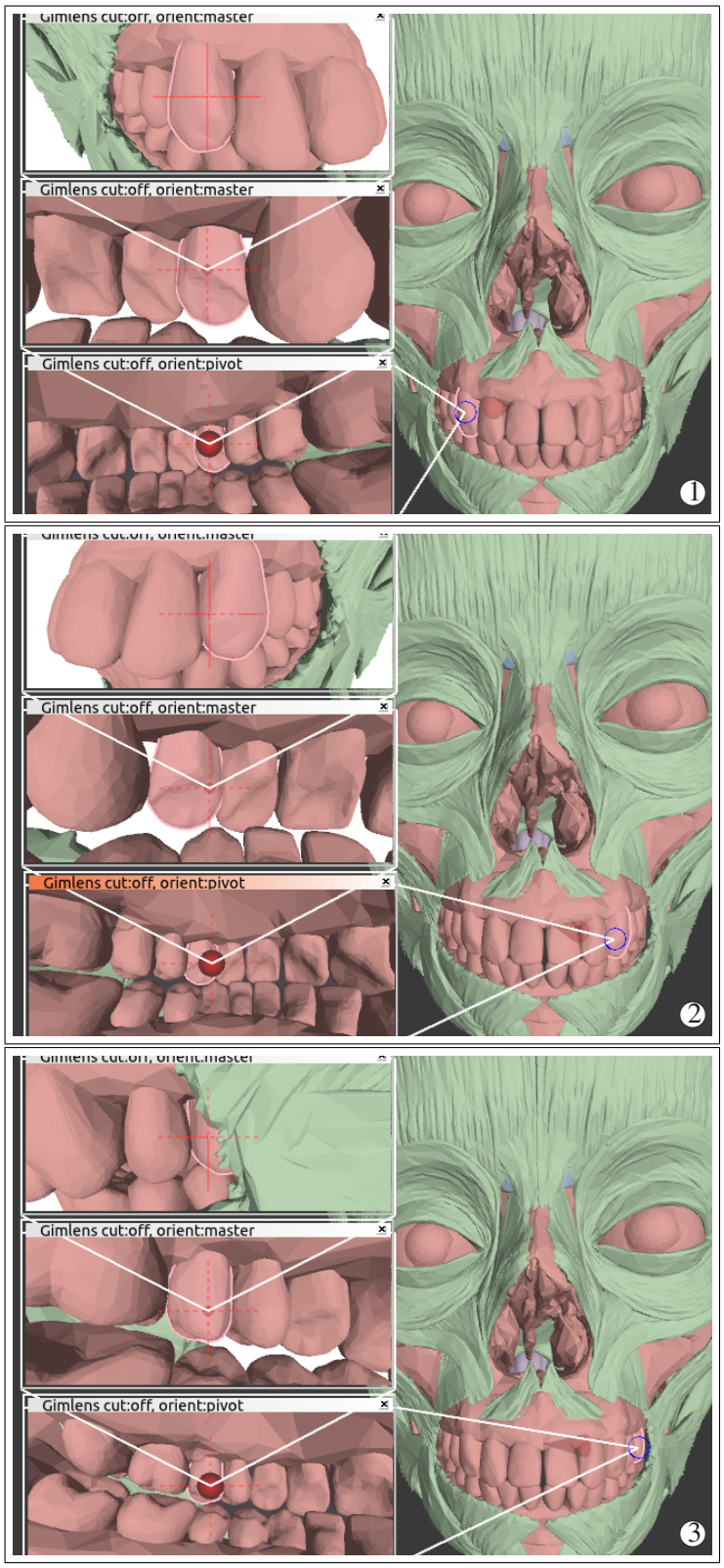


FIGURE 5.7: Three Gimlenses combined to explore the inside of the maxillary dental arcade of a patient using different orientation constraints. Dragging the single focus point that controls all three lenses updates all views automatically, providing complementary perspectives on the successive teeth.

illustrates the potential of this feature. The first lens is set up with a **Pivot** constraint, that maintains the camera position in-place as it pivots around to follow the focus point. This is particularly useful when exploring the inside of cavities (Figure 5.7-bottom lens). The two other lenses are set up with a **Parent** orientation constraint. This means that their orientations are defined relative to the one of their parent lens (the first lens). With these constraints set up, as we slide the focus point along the teeth from left to right, the lenses automatically move along and rotate, maintaining views from both inside and outside the patient's mouth without the need for users to make any adjustment themselves.

5.7 Discussions

We presented magnification lenses for navigating complex 3D scenes. As demonstrated throughout the chapter, the technique can prove useful in a variety of domains : for inspecting an industrial CAD model of a car engine, for exploring an anatomical representation of the human head, or walking through a 3D map of a plant.

Gimlenses are based on a content-aware design approach. The *cone-cut* 3D occlusion management technique adapts to the size of the parts to provide a better trade-off between visibility of the focused object and context preservation. The *drilling* navigation technique which allows quick selection of objects in dense scenes, adjusts the position of the focus point depending on the geometry of the underlying object. And the orientation management system allowing to update view angle of cascaded lenses also depends on geometry of focused objects.

The content-aware design approach allowed for better optimization of the interface to the content of the scene. However, it requires the data to be well segmented into distinct part. Some datasets, such as results from 3D scanners are made of unorganized triangles. These datasets typically do not support navigation with Gimlenses as is. They require first to be segmented. If automatic algorithms exist to segment 3D data, they are prone to making mistakes, and it would be interesting to see how Gimlens could help in monitoring such segmentation processes. Assigning each triangle a category typically requires detail-in-context interpretation, and Gimlens could be a valuable support for such decision making.

From an implementation point of view, we wrote a fully functional prototype of Gimlenses for desktop environments, using C/C++ with OpenGL running on the Qt framework. Some effects, such as semi-transparent spots for handling conflict between two *cone-cuts* were achieved with GLSL shaders. Besides using an AABB tree data structure to optimize the navigation technique which makes

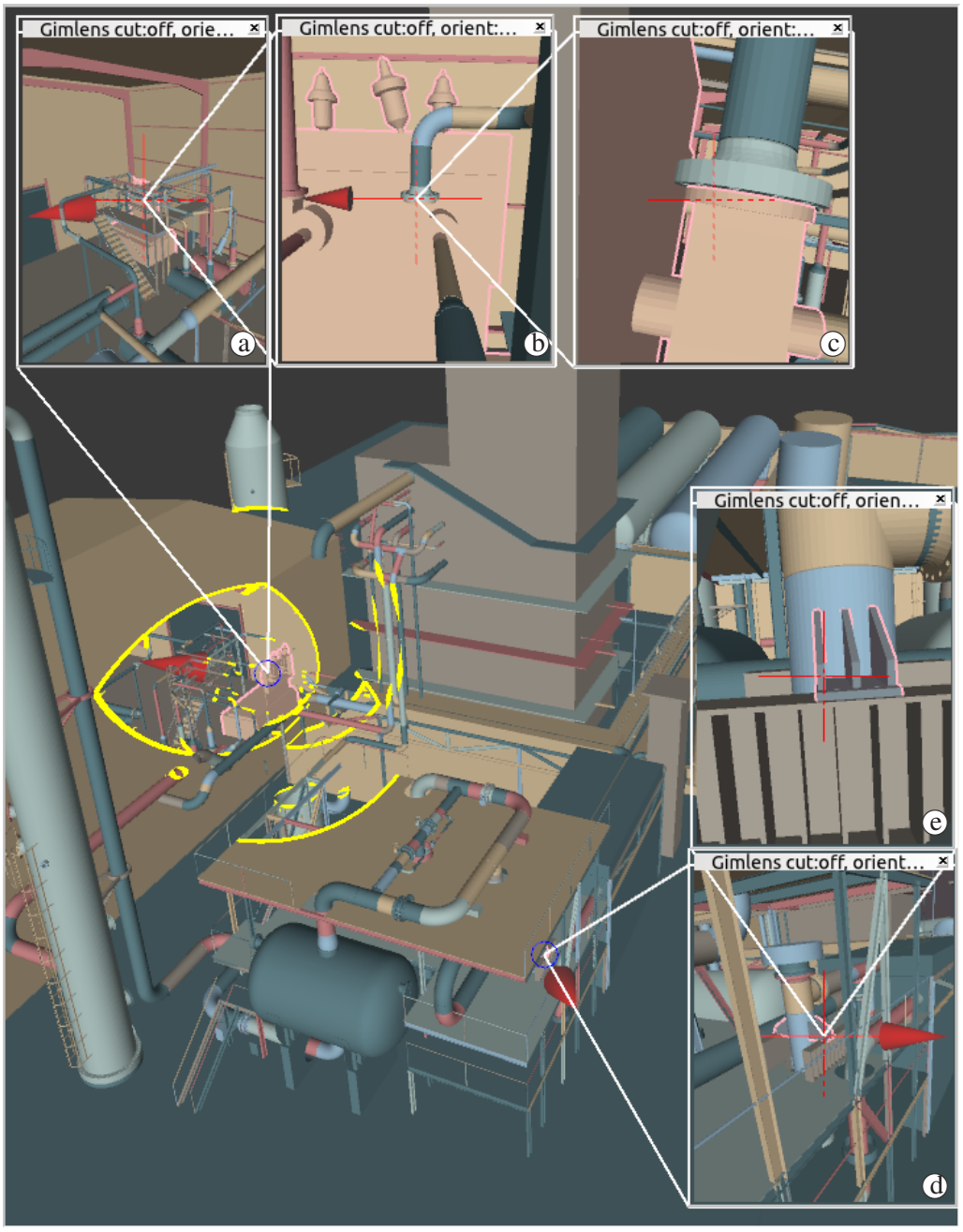


FIGURE 5.8: Exploring an air distillation plant. Three Gimlenses are cascaded to magnify a steal pipe. (a) shows a view from the left to better situate the focused part within the shed, (b) and (c) apply magnification successively. (d) and (e) magnify a distinct part of the plant.

intensive use of ray-casting into the model to find the projection of the *object selector* in the scene. We achieved interactive frame-rate rendering without any further optimization, on a high-end computer [1] for all our three models : the Air Plant, the car engine and the anatomical head, featuring respectively : 1600, 266, 58 Millions of triangles each. However implementation supporting the interactive rendering of more complex datasets, or on computers with less computing power, requires further optimization in the rendering pipeline, such as the one introduced in [9].

A venue for future work, would be to investigate how Gimlens could support collocated collaborative exploration in wall-sized display environments. The design of large 3D models require the collaboration of several designers, and yet their is no system that provides them with support for collaborative inspection of their work, looking for conflicts, understanding them and taking design decisions. Wall-sized displays running Gimlenses could offer the opportunity for such navigation. However, porting Gimlens to wall-sized displays raises the question of the mapping of Gimlens parameters to user interface controls and input channels. While the mapping to wheel-equipped mice or even a gesture-enabled trackpad for a laptop or desktop workstation is straightforward, the mapping to the more exotic input devices typically used in CAVEs or when interacting with wall-sized displays (motion tracking, mid-air pointing device, etc.) will require more thinking and empirical testing.

Another issue related to collaboration with Gimlenses is the definition of rules of ownership for the lenses. Each Gimlens generates occlusion that prevents exploration of some part of the scene by other users. Users could be tempted to adjust the content of the same Gimlens. And in such environments, it is not obvious who is interacting which lens. This requires some rules of ownership to prevent conflicts.

1. The computer was an HP Z-series 800 PC equipped with an nVidia Quadro 4000, and an Intel Xeon E5-1650 CPU running at 3.20GHz.

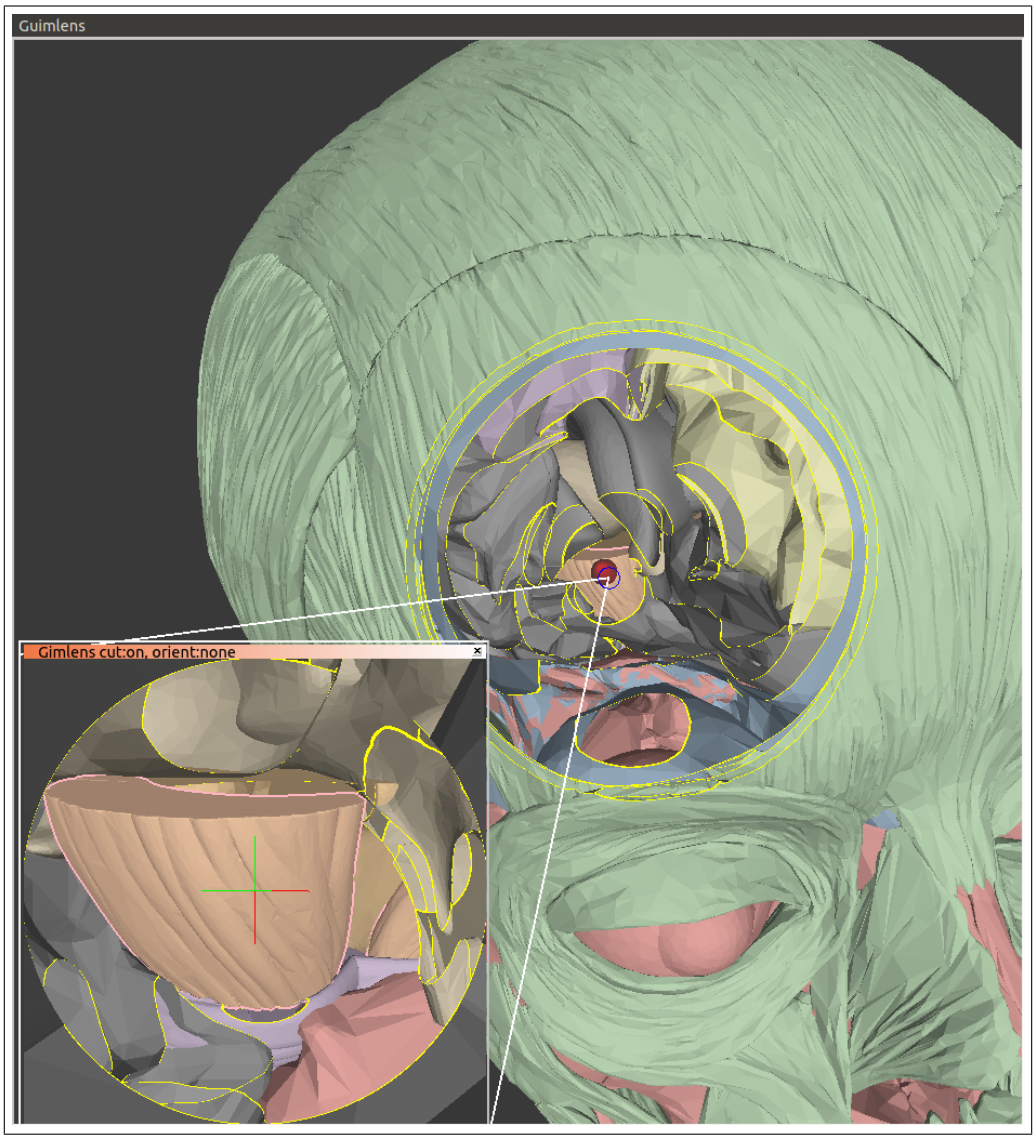


FIGURE 5.9: Exploring a brain with Gimlenses – Plate 1

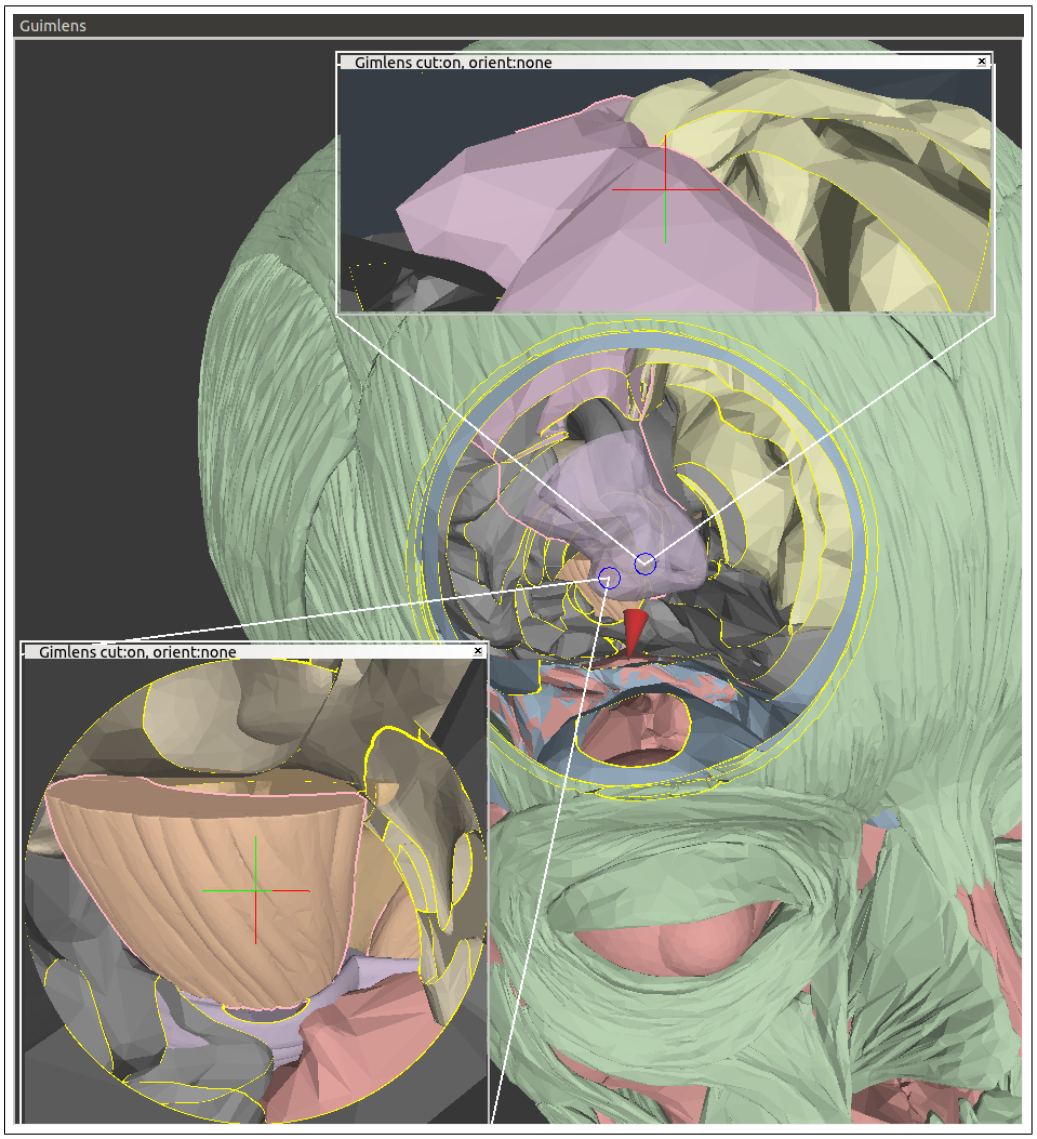


FIGURE 5.10: Exploring a brain with Gimlenses – Plate 2. Our technique handles conflict between lenses (see Section 5.6.1).

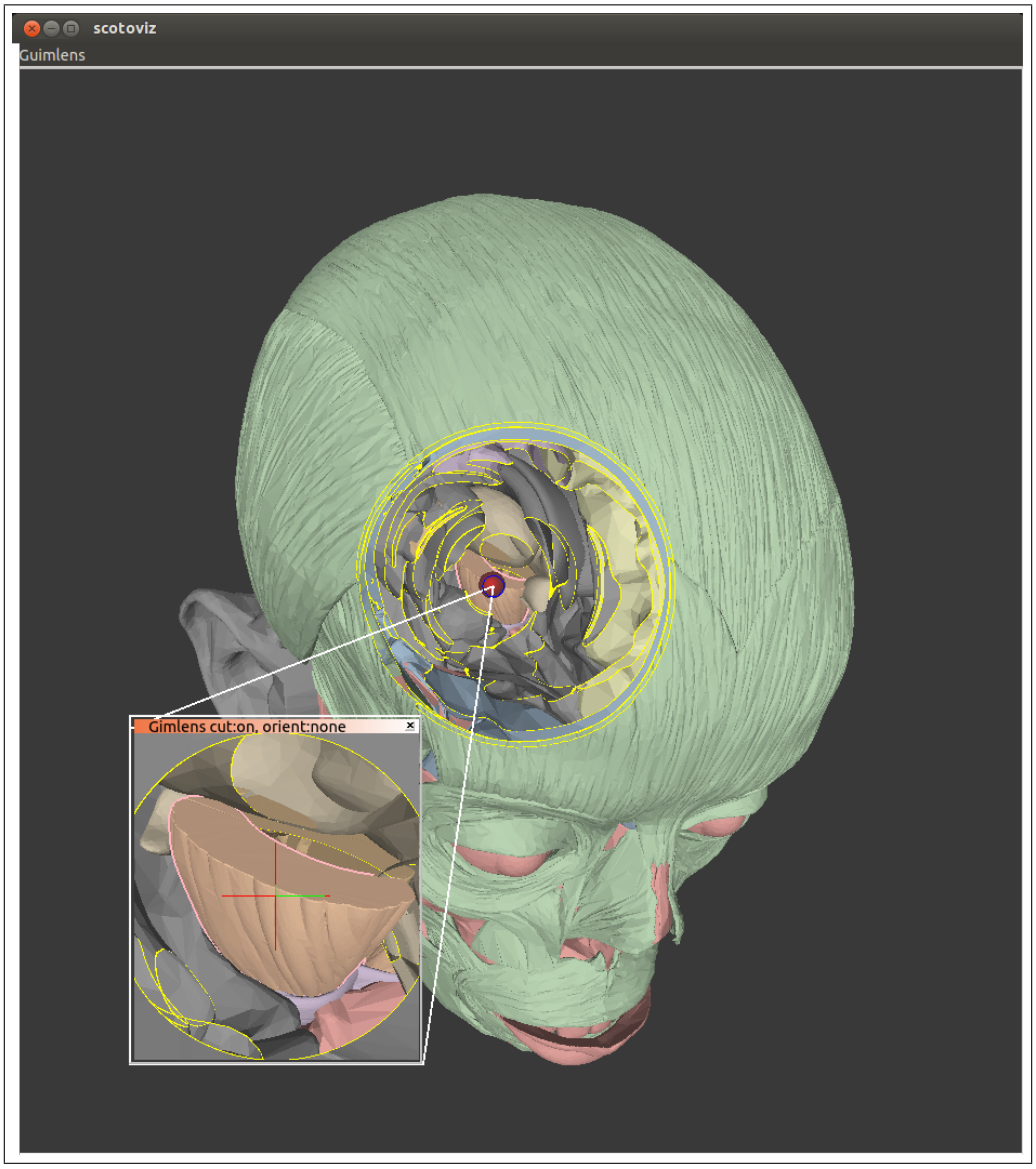


FIGURE 5.11: Exploring a brain with Gimlenses – Plate 3.

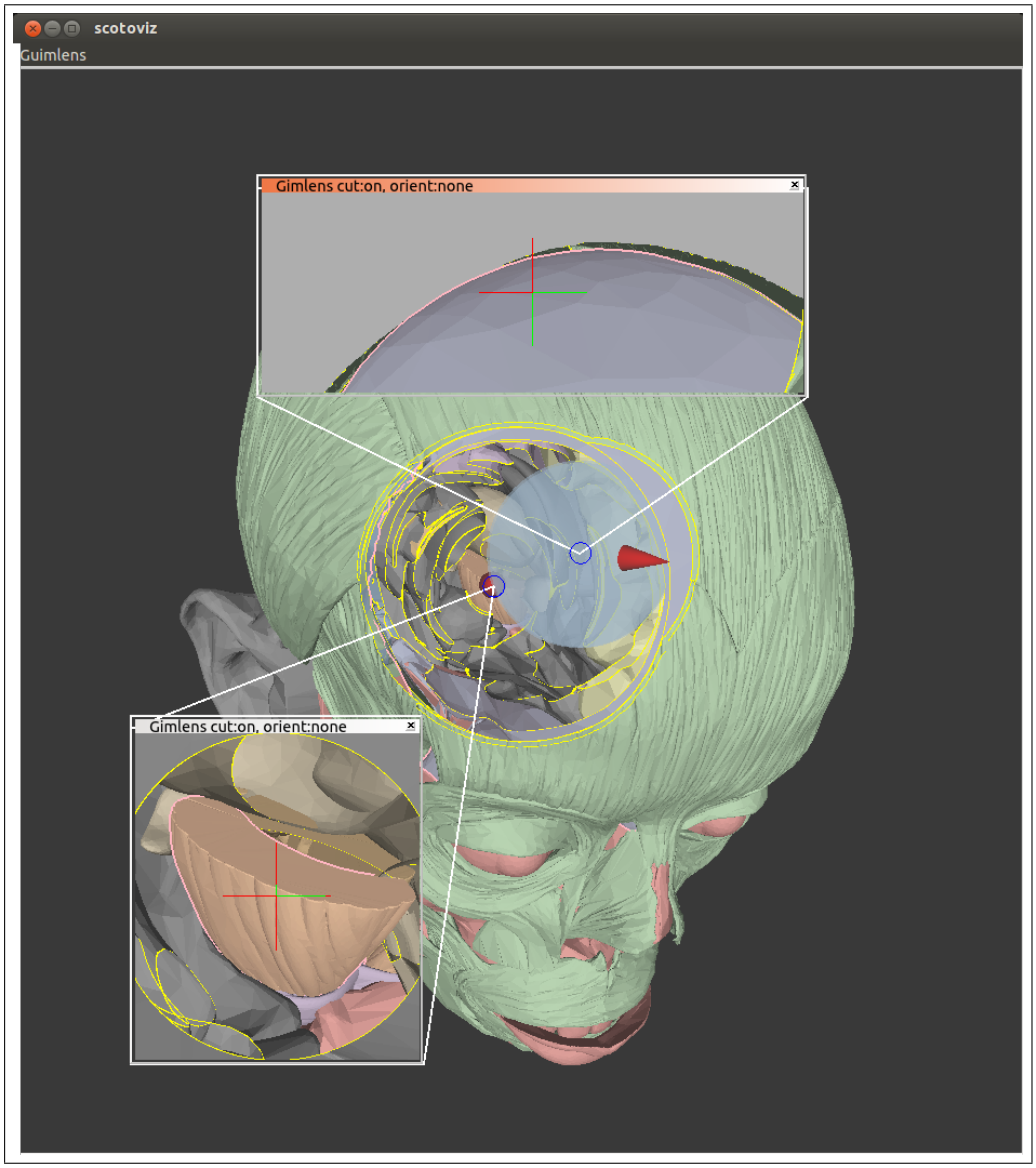


FIGURE 5.12: Exploring a brain with Gimlenses – Plate 4. Our technique handles conflict between lenses (see Section 5.6.1).

Chapitre 6

Conclusion and Future Research Directions

We introduced several techniques for multiscale navigation in large datasets, all based on a content-aware design approach. We have discussed their potential applications for supporting collaborative navigation on large displays. We have discussed how they may be integrated in a navigation pipeline. And we have also discussed several drawbacks and benefits related to this general approach, and began to discuss their implication for design. This chapter, after summarizing contributions of the thesis, outlines future research directions along all of these three paths.

6.1 Summary of Contributions

6.1.1 Design Approach to Multiscale Navigation

In this dissertation we introduced a new design approach to multiscale navigation. The content-aware design approach adapts the interface to the content of the scene, to provide visualizations of higher relevance, and ease the understanding of objects of interest. The techniques introduced in the dissertation showed that the approach can help design efficient navigation techniques. A controlled experiment showed increases in performance for the Arealens technique over a conventional fisheye lens on a multiscale search task.

Content-awareness complements two other design approaches to multiscale navigation that we identified in Section 1.4 : content-agnostic, that provides techniques allowing navigation in datasets ; and content-driven, that provides navi-

gation in more specific category of representations, thus optimizing the visualization. While the content-driven approach provides global optimization of the interface over a certain type of data and representations, the content-aware approach allows local optimization of the interface.

The design of content-aware navigation techniques involves adaptation of the interface that could potentially distract users. The design of Pathlens and Gimlens showed that relying on known metaphors helps solve this problem by improving the predictability of changes in the interface. The design of adaptive interfaces is a matter of finding a trade-off between visualization of higher relevance and the necessary animated changes caused by the adaption.

6.1.2 Multiscale Navigation Techniques

We have presented three novel multiscale navigation techniques for large datasets. Each technique is based on the content-aware design approach.

In this thesis, we break the usual behavior of fisheye lenses by proposing to dynamically change their geometry while preserving visual continuity between the focus and the context regions. In Chapters 2 and 3 we introduced two techniques, Pathlens and Arealens. Pathlens improves upon regular fisheye lenses by adapting the shape of the lens to the geometry of the underlying content, better accommodating distortion to better preserve the surrounding context. The adaptation of the lens is based on a novel water-drop metaphor, of which we describe an original implementation based on implicit surfaces. Arealens is a multi-foci fisheye lens that achieves interactive expansion of objects in dense fields while preserving continuity between magnified and context regions. The technique introduces a mapping algorithm for expanding objects in 2D fields while preventing objects from overlapping each other. An empirical evaluation, reported in Section 3.4 showed that Arealenses outperform regular fisheyes for a multiscale visual search task.

We introduced a library for creating distortion-oriented presentation techniques allowing to render fisheye lenses featuring possibly multiple flat tops of arbitrary shape. The library introduces a simple interface for creating lenses, that consists of defining geometry and mapping of flat tops. The library ensures continuity in-between Each flat-tops get integrated into a singles continuous presentation by the library that distorts the representation in-between to integrate them smoothly. The Jellylens library was used to design and implement both Arealens and Pathlens techniques, and proved useful for fast prototyping of rich distortion-oriented presentation technique.

The content-aware design approach was also instrumental for the design of Gimlens which are magnification lenses allowing detail-in-context navigation in large complex 3D models. We introduced a 3D occlusion management technique allowing to reveal objects of interest in dense scenes featuring numerous parts clustered closely together. Gimlens also introduces a new drilling technique allowing exploration of such scenes. Gimlens provides support for navigating a scene with multiple lenses, allowing to cascade and coordinate views, relieving users from intensive camera readjustments while navigating.

6.2 Principles of Content-Aware Design Approach

6.2.1 Understanding Adaptation

Content-aware navigation techniques involve two distinct types of adaptation. Either the adaptation concerns the interface, providing visualization of higher relevance (such as the lens shape of Arealens and Pathlens techniques, and the size of the aperture of the 3D occlusion management technique of Gimlens) ; or it concerns the interaction technique allowing users to display certain parts of the visualization easier (such as the *drilling* technique of Gimlenses). The kind of adaptation we consider in this section is the adaptation of the interface. The design of the various navigation techniques introduced in this thesis reveals new potential benefits and drawbacks of the content-aware design approach. We discuss them in this section and we propose future work to improve our understanding of the mechanisms involved in this design approach. As a general goal, we would like to set research directions that would help draw rules to guide the future implementation of content-aware techniques.

About adaptive interfaces, Shneiderman [109] reported that unpredictable changes in the interface may cause user anxiety : as they do not understand why the interfaces changes, users are anxious about potential further changes. Shneiderman was actually commenting interfaces based on an autonomous agent interaction model, which predicts users' interest and anticipates their needs by adapting the interface. However, such interaction models are very likely to make mistakes, and change the interface against users' immediate own interest. As a result, users do not understand why the interface changed, and they become anxious that the interface would change again. The content-aware design approach is different and less likely to suffer from this problem. Instead of adapting to predicted changes of users' interests, we adapt the interface depending on the content of regions of the representation users are focusing on. As users see changes in the scene, they better anticipate and understand changes of the interface. However,

the success of such interfaces depends on their ability to reflect changes of the scene. As shown by the design of Pathlens, the use of interaction metaphors can help design predictable adaptation.

The design of Arealens and Gimlens both pointed-out a second issue : adaptation of the interface could cause user distraction. Although the results of one particular controlled experiment (presented in Section 3.4) do not show significant impact on Arealens performance, users still reported being disturbed when the lens was being operated by someone else. Possible further work includes trying to measure at a perceptual level the impact on navigation tasks performance, of animated patterns moving at varying speeds, and at varying distances from users' focus of attention. Such controlled experiments could help better understand how animation in adaptive interfaces affect navigation, and provide guidelines to support future design of content-aware navigation interfaces.

The design of adaptive interfaces revealed potential advantages beyond providing visualizations of higher relevance. As reported by Jul and Furnas, disruptions in the physical environment (such as stairs, elevator, doors) are navigational cues that help people find their way in the physical world [67]. In the same spirit, we argue that changes in an interface also disturb virtual navigation and, could provide navigational cues that would help users remember specific locations. Of course empirical evaluations are needed to validate this hypothesis.

6.2.2 Interface Performance

The controlled experiment reported in Section 3.4 revealed that Arealenses outperform conventional fisheye lenses on a visual search task. Other recent navigation techniques have been shown to perform better than the technique they build upon [35], yet, they tend not to be adopted in end-user applications. This gap between the introduction of new navigation techniques and their adoption by software designers affects interaction techniques at large [10]. Possible causes may be design and implementation costs [10]. Designers need to learn new design practices. Developers may have to re-implement the new interaction techniques from scratch, while conventional techniques are available off the shelf in popular GUI frameworks.

This issue is further aggravated in the case of navigation techniques, as often those are designed for specific types o data and representations. This is the case for most of the graph navigation techniques, and all the techniques we reviewed as a presentation of the content-driven designed approach in Section 1.4.2. This case-by-case design approach, results in the introduction of new navigation tech-

nique for each data type. Which leads to non-reusable navigation techniques and extensive costs for both designers and developers, thus making integration more tedious and unlikely to happen.

From this perspective, the content-aware design approach seems a viable approach to improve integration of the resulting navigation techniques into end-user applications. Indeed, the techniques introduced in this thesis work with a wider range of data representations. Arealens and Pathlens could be used to navigate in many different sets of data including GUIs, graphs, street/transport maps. While Gimlens could navigate in very different 3D models such as CAD models of manufactured objects featuring highly regular shapes, to the very noisy surfaces of the scanned 3D models.

Pushing this comparison further would require an indicator to better quantify the wideness of the range of data types that suit a navigation technique. There are several ways to evaluate the performance of navigation technique [35]. However, very low attention was given to the evaluation of the versatility of navigation techniques, which plays an important role in their integration. Understanding how content-aware techniques could make those techniques more versatile could help better understand potential impact on adoption by users.

6.3 Objects Definition

The content-aware navigation techniques introduced in this thesis require some geometrical information about the objects of interest populating the scene. Indeed, to provide visualization of higher relevance, the techniques adapt some of their parameters to the geometry of the objects of interest. The type of this information depends on the nature of the representation : Arealens and Pathlens require the 2D contour of the regions of interest populating the scene, and Gimlens requires the 3D surface model of the objects.

As discussed earlier, the acquisition of this geometrical information is highly dependent on the nature of the graphics. Vector-based representations such as SVG files, HTML source of web pages or 3D CAD models, all make this data readily available. On the contrary, raw images (possibly resulting from captures of the world such as 3D scanned models, 2D pixel images or 3D volumes) provide complex highly unstructured datasets. For the latter category of datasets, we can apply some segmentation algorithms to extract the geometry information, that we provide, then, as input to the adaptation techniques. However, in both cases, the identification of objects of interest is highly task-dependent, and providing a predefined set of objects assumes that users know in advance what

they are looking for, and where those objects are. Which is very unlikely during exploratory visualization of data, when users refocus their attention, and readjust their navigation goals as new information is encountered. Enabling users to interact with objects of interest could improve exploration of large datasets with content-aware navigation techniques. This involves dynamically changing the set of objects identified as being of interest, switching between several predefined sets, or adjusting the objects' geometry. Those are challenging issues that require further investigation.

Sketching or gesture-based interaction techniques could let users draw shapes that could be used directly, without further processing, to define the geometry of the objects the navigation technique adapt to. Or, when information geometry is available, the sketched shapes could be matched against the shapes found in the scene (implementing techniques in the spirit of [45]) thus acting as a selection mechanism. This could be used to let users readjust the set of objects of interest while navigating. In cases where the geometry information is structured – such as in transportation or street maps that group objects by category : routes, sightseeing, water pond – such selection techniques could be further extended to select a whole category of objects at once.

In cases where no geometry is available, allowing users to define the shape of objects of interest while navigating provides an interesting alternative to automatic segmentation algorithms. Applying smart selection tools from image editing software, such as the Magic Wand or the Quick Selection Tool from Adobe Photoshop, would allow for interactive definition of regions of interest based on pixel colors and user input.

The Histomages technique [33] provides another promising approach for interactively defining geometries of objects of interest. Indeed, Histomages considers color histograms as spatial rearrangements of image pixels. It provides views allowing to choose between several histograms of pixels (red, green and blue channels, lightness, hue or saturation), along with simple selection tools (rectangle or lasso selection) to select pixels. Users can combine several views to achieve rich image selection, allowing to refine the selection according to localization of pixels and other attributes (see Figure 6.1).

Extending Histomages selection capabilities into an overall interactive image segmentation technique to define regions of interest require the identification of relevant attributes to support meaningful region selection. In particular, combined with Arealens or Pathlens, it could help define the geometry of the objects of interest on the fly while navigating large datasets.

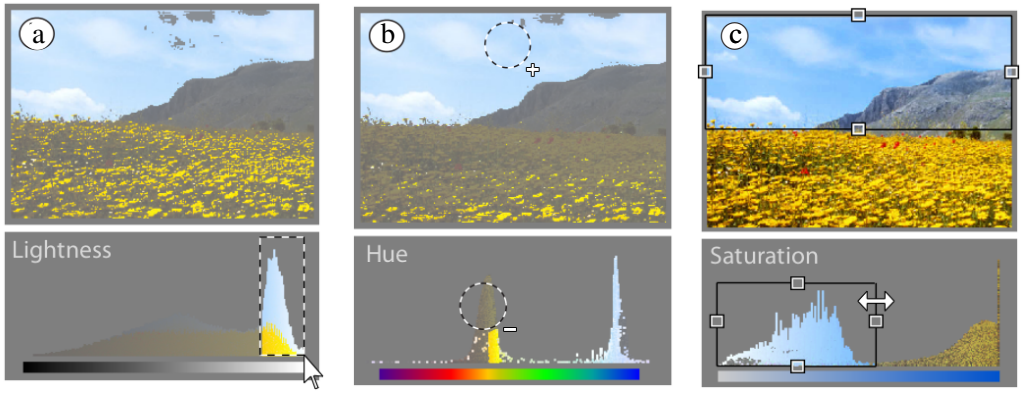


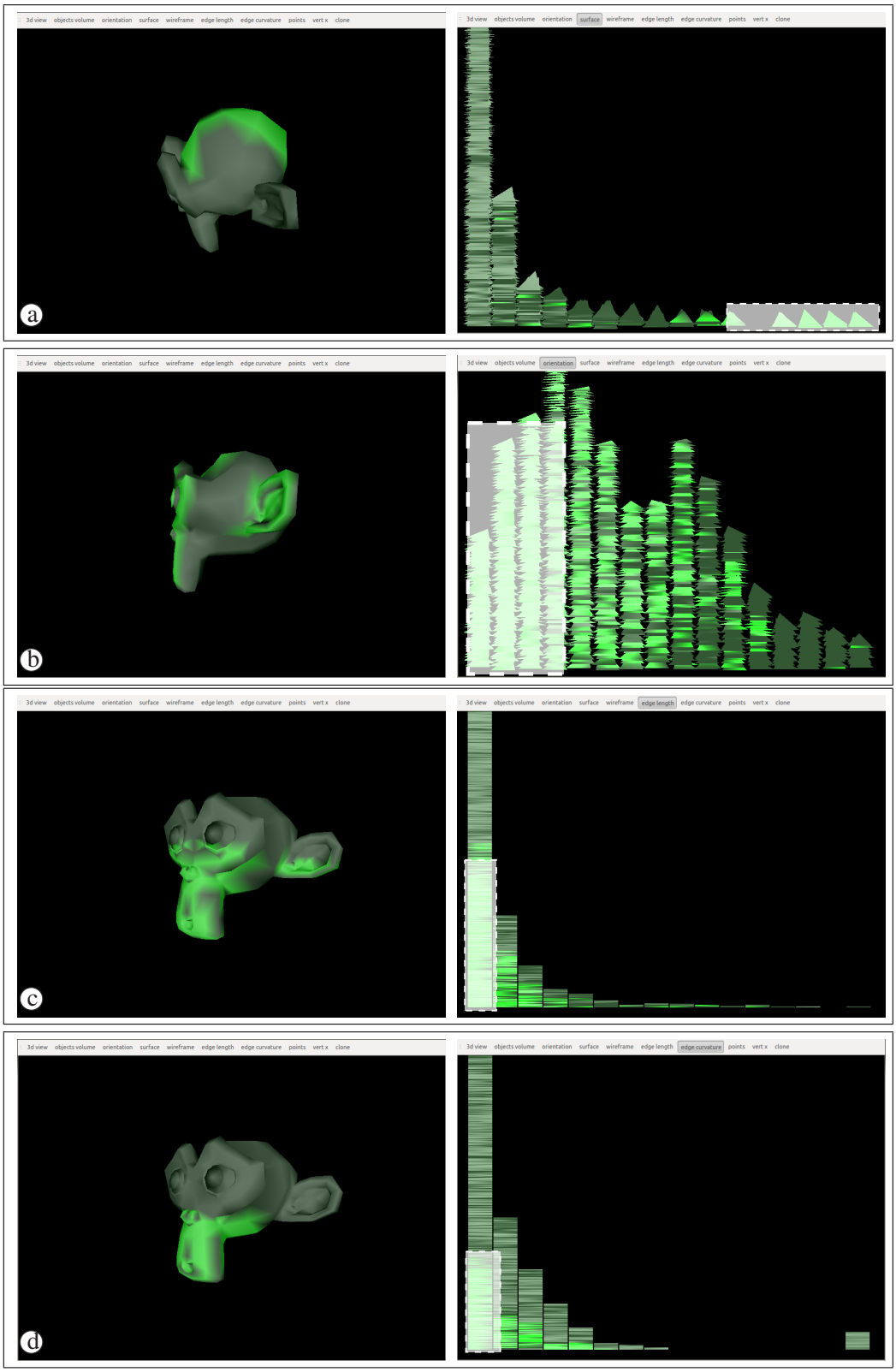
FIGURE 6.1: Histomages [33] considers color histograms as spatial rearrangements for achieving rich pixel selection : (a) pixels are rearranged into a lightness histogram ; (b) pixels are rearranged into a hue histogram and yellow pixels are filtered out with "subtract" selection brush (bottom). Missing pixels are added with the "add" selection brush on the image (top) ; (c) selecting and resizing blue pixels in the saturation histogram allow to enhance the sky.

Building on the Histomages approach, we are working on an interface providing support for 3D surface selection. The interface provides several histograms, rearranging each one of the graphics primitives (triangle, edge or point) according to one particular geometrical attribute. The view maintains visual continuity while switching histograms by animating graphics primitives from their current location to the final position in the histogram. While switching between two different graphics primitive representation, for instance from a triangle histograms to a edge histograms, animation fades out the former primitives while the new graphics primitives fades in. Allowing to coordinate selection across several combined views showing different histograms, the interface provides support for elaborate selections and refinement thereof.

Although we support various geometrical attributes for each one of the graphics primitives (area and orientation for triangles, length and curvature for edges, and coordinated for points) identification of the relevant properties to support meaningful selection of the 3D surface remains an open questions.

The integration of such approach into a multiscale interface such as Gimlens could provide an efficient selection tool, allowing interactive segmentation of raw 3D surfaces like scanned surfaces. This approach could be particularly relevant, as deciding what object each triangle belongs to is a matter of interpreting details (the triangle) within their surrounding context (the overall model).

Supporting interaction with the objects in the scene in the interface and com-


3d view objects volume orientation surface wireframe edge length edge curvature points vert x clone
a
b
c
d

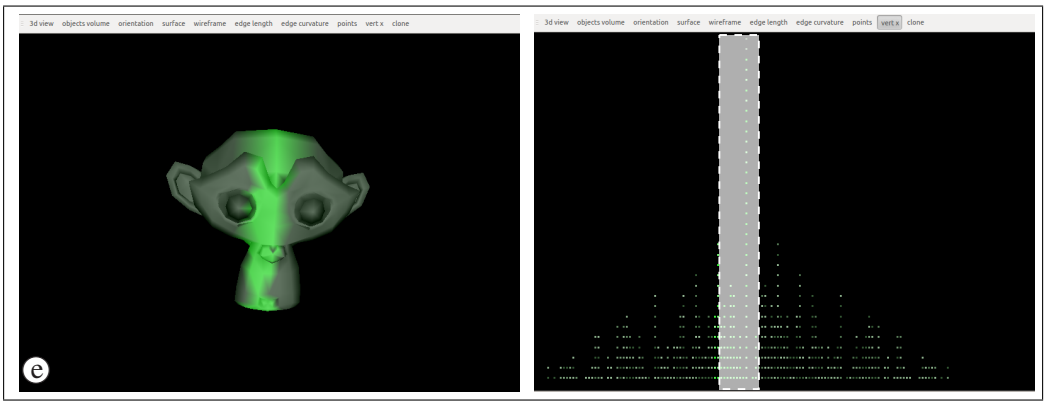


FIGURE 6.2: Grinder considers rearrangement of triangles, edges or points of 3D surfaces into histograms to support rich selection capabilities. Combining multiple views allows to perform a rectangle selection on a histogram (right) while the selected primitives are highlighted on a 3D view of the model (left). Triangles can be binned by their surface area (a) or orientation (b) ; edges can be binned by length (d) or surface curvature (e) ; points can be binned by x-coordinate (c).

bining it with interaction for navigation raises new questions. Users no longer simply explore a dataset, but also manipulate it, grouping the graphics primitive into meaningful entities, adjusting finely each groups' delimitation. The evaluation of interfaces resulting from this approach, are necessary to understand the high-level impact on navigation performance.

6.4 Application to Wall-Sized Displays

This thesis focused on detail-in-context magnification lens techniques. In addition to easing navigation and interpretation, they support simultaneous zoomed-in views to explore remote regions of the datasets concurrently. Such functionality is particularly relevant to support collaborative navigation on wall-sized displays. However it raises challenges for both interaction design and implementation.

Wall-sized displays feature rich input capabilities to visualize and manipulate very large datasets. They are usually driven by a cluster of computers. Porting graphics and visualization libraries, such as the Jellylens library presented in Chapter 4, raises several implementation issues discussed in Section 4.5. Distributing distortion-oriented rendering techniques and managing very large datasets among several cluster nodes is challenging. Especially given the comparatively low video memory resources provided by graphics hardware. Similar issues must be addressed to successfully port Gimlenses to such visualization platforms.

Mapping input to control actions to navigate and manipulate large visualizations on wall displays is complex, especially given the large variety of input devices. Hand-held input devices such as tablets or remote controllers and game pads embedding joystick, and all the mid-air interaction capabilities provided by 3D motion tracking systems, offer interaction designers a wide range of solutions, which can be confusing. Identifying good practices, interaction models and design guidelines to support the design of navigation techniques the effective exploration of large datasets on such platforms is a challenging but interesting venue for future work.

Multiple users navigating the same datasets with magnification lenses are very likely to interact with one another, or could be getting in each others' way. Interfaces supporting efficient collaborative navigation will have to manage such interactions, and provide support to deal with potential conflicts between multiple users, enabling the sharing and locking of views.

Refereed International Conferences

[1] Cyprien Pindat, Emmanuel Pietriga, Olivier Chapuis, and Claude Puech. Jellylens : content-aware adaptive lenses. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, pages 261–270, New York, NY, USA, 2012. ACM.

[2] Cyprien Pindat, Emmanuel Pietriga, Olivier Chapuis, and Claude Puech. Drilling into complex 3d models with gimlenses. In *Proceedings of the 2013 ACM symposium on Virtual reality software and technology*, VRST '13, pages 223–230, New York, New York, USA, 2013. ACM. *Best Paper Award*.

Bibliographie

[3] Caroline Appert, Olivier Chapuis, and Emmanuel Pietriga. High-precision magnification lenses. In *CHI '10 : Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 273–282. ACM, 2010.

[4] Caroline Appert and Jean-Daniel Fekete. OrthoZoom scroller : 1D multi-scale navigation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, page 21–30, New York, NY, USA, 2006. ACM.

[5] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. *ACM TOG*, 26(3), 2007.

[6] R. Bane and T. Hollerer. Interactive tools for virtual x-ray vision in mobile augmented reality. In *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, pages 231–239, 2004.

[7] Alan H. Barr. Global and local deformations of solid primitives. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '84, pages 21–30. ACM, 1984.

[8] Patrick Baudisch, Bongshin Lee, and Libby Hanna. Fishnet, a fisheye web browser with search term popouts : a comparative evaluation with overview and linear view. In *AVI '04 : Proceedings of the working conference on Advanced visual interfaces*, AVI '04, pages 133–140. ACM, 2004.

[9] William V. Baxter,III, Avneesh Sud, Naga K. Govindaraju, and Dinesh Manocha. GigaWalk : interactive walkthrough of complex environments. In *Proceedings of the 13th Eurographics workshop on Rendering*, EGRW '02, page 203–214, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[10] Michel Beaudouin-Lafon. Instrumental interaction : an interaction model for designing post-WIMP user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '00, page 446–453, New York, NY, USA, 2000. ACM.

[11] Benjamin B. Bederson. Fisheye menus. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, UIST '00, pages 217–225. ACM, 2000.

[12] Benjamin B Bederson, Aaron Clamage, Mary P Czerwinski, and George G Robertson. DateLens : A fisheye calendar interface for PDAs. 11(1) :90–119, 2004.

[13] Benjamin B Bederson, Jesse Grosjean, and Jon Meyer. Toolkit design for interactive structured graphics. *Software Engineering, IEEE Transactions on*, 30(8) :535–546, 2004.

[14] Benjamin B. Bederson and James D. Hollan. Pad++. In *Proceedings of the 7th annual ACM symposium on User interface software and technology - UIST '94*, page 17–26, New York, New York, USA, November 1994. ACM Press.

[15] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, pages 35–42. ACM, 1992.

[16] Anastasia Bezerianos, Petra Isenberg, Olivier Chapuis, and Wesley Willett. Perceptual affordances of wall-sized displays for visualization applications : Color. April 2013.

[17] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses : the see-through interface. In *SIGGRAPH '93 : Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 73–80. ACM, 1993.

[18] Arpan Biswas and Vadim Shapiro. Approximate distance fields with non-vanishing gradients. *Graphical Models*, 66(3) :133–159, 2004.

[19] R. Blanch and E. Lecolinet. Browsing zoomable treemaps : Structure-aware multi-scale navigation techniques. *IEEE Transactions on Visualization and Computer Graphics*, 13(6) :1248–1253, 2007.

[20] James F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1 :235–256, July 1982.

[21] Joachim Böttger, Ulrik Brandes, Oliver Deussen, and Hendrik Ziezold. Map warping for the annotation of metro maps. *IEEE Comput. Graph. Appl.*, 28 :56–65, 2008.

[22] Joachim Böttger, Martin Preiser, Michael Balzer, and Oliver Deussen. Detail-In-Context visualization for satellite imagery. *Computer Graphics Forum*, 27(2) :587–596, 2008.

[23] John Brosz, Sheelagh Carpendale, and Miguel Nacenta. The undistort lens. *Computer Graphics Forum*, 30 :881–890, 2011.

[24] John Brosz, Faramarz F. Samavati, S. Carpendale, and Mario Costa Sousa. Single camera flexible projection. In *NPAR '07 : Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering*, NPAR '07, pages 33–42. ACM, 2007.

[25] Stefan Bruckner, Sören Grimm, Armin Kanitsar, and M. Eduard Gröller. Illustrative context-preserving volume rendering. In *Proceedings of the Seventh Joint Eurographics / IEEE VGTC conference on Visualization*, EUROVIS'05, pages 69–76, Aire-la-Ville, Switzerland, Switzerland, 2005. EA.

[26] Michael Burns and Adam Finkelstein. Adaptive cutaways for comprehensible rendering of polygonal scenes. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 154 :1–154 :7. ACM, 2008.

[27] Nicholas Burtnyk, Azam Khan, George Fitzmaurice, Ravin Balakrishnan, and Gordon Kurtenbach. Stylecam : interactive stylized 3d navigation using integrated spatial & temporal controls. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, UIST '02, pages 101–110, New York, NY, USA, 2002. ACM.

[28] M. S. T. Carpendale, John Ligh, and Eric Pattison. Achieving higher magnification in context. In *UIST '04 : Proceedings of the ACM Symposium on User Interface Software and Technology*, UIST '04, pages 71–80. ACM, 2004.

[29] M. S. T. Carpendale and Catherine Montagnese. A framework for unifying presentation space. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 61–70, New York, NY, USA, 2001. ACM.

[30] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. 3-dimensional pliable surfaces : for the effective presentation of visual information. In *UIST '95 : Proceedings of the ACM Symposium on User Interface Software and Technology*, UIST '95, pages 217–226. ACM, 1995.

[31] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Extending distortion viewing from 2D to 3D. *IEEE Comput. Graph. Appl.*, 17(4) :42–51, July 1997.

[32] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Making distortions comprehensible. In *Proceedings of the 1997 IEEE Symposium on Visual Languages (VL '97)*, VL '97, pages 36–45. IEEE, 1997.

[33] Fanny Chevalier, Pierre Dragicevic, and Christophe Hurter. Histomages : fully synchronized views for image editing. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, page 281–286, New York, NY, USA, 2012. ACM.

[34] G. Cirio, P. Vangorp, E. Chapoulie, M. Marchal, A. Lecuyer, and G. Drettakis. Walking in a cube : Novel metaphors for safely navigating large virtual environments in restricted real workspaces. *IEEE Transactions on Visualization and Computer Graphics*, 18(4) :546–554, 2012.

[35] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM CSUR*, 41 :2 :1–2 :31, 2009.

[36] Andy Cockburn, Philip Quinn, Carl Gutwin, and Stephen Fitchett. Improving scrolling devices with document length dependent gain. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, page 267. ACM Press, 2012.

[37] C. Coffin and T. Hollerer. Interactive perspective cut-away views for general 3d scenes. In *3D User Interfaces, 2006. 3DUI 2006. IEEE Symposium on*, pages 25–28, 2006.

[38] C. Correa and Kwan-Liu Ma. The occlusion spectrum for volume classification and visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6) :1465–1472, 2009.

[39] Carlos D. Correa and Kwan-Liu Ma. Visibility histograms and visibility-driven transfer functions. *IEEE Transactions on Visualization and Computer Graphics*, 17 :192–204, 2011.

[40] Carlos D. Correa and Deborah Silver. Programmable shaders for deformation rendering. In *GH '07 : Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics hardware*, GH '07, pages 89–96. EA, 2007.

[41] Carolina Cruz-Neira, Daniel J. Sandin, Thomas A. DeFanti, Robert V. Kenyon, and John C. Hart. The cave : audio visual experience automatic virtual environment. *Commun. ACM*, 35(6) :64–72, June 1992.

[42] Rudolph P. Darken and John L. Sibert. Wayfinding strategies and behaviors in large virtual worlds. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '96, page 142–149, New York, NY, USA, 1996. ACM.

[43] Rodrigo A. de Almeida, Clément Pillias, Emmanuel Pietriga, and Pierre Cubaud. Looking behind bezels : french windows for wall displays. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, page 124–131, New York, NY, USA, 2012. ACM.

[44] A. Dietrich, A. Stephens, and I. Wald. Exploring a boeing 777 : Ray tracing large-scale cad data. *Computer Graphics and Applications, IEEE*, 27(6) :36–46, 2007.

[45] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Sketch-based 3D shape retrieval. In *ACM SIGGRAPH 2010 Talks*, SIGGRAPH '10, page 5 :1–5 :1, New York, NY, USA, 2010. ACM.

[46] N. Elmqvist and P. Tsigas. A taxonomy of 3D occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(5), October 2008.

[47] Niklas Elmqvist, Nathalie Henry, Yann Riche, and Jean-Daniel Fekete. Melange : space folding for multi-focus interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 1333–1342, New York, NY, USA, 2008. ACM.

[48] Steven K. Feiner and Dorée Duncan Seligmann. Cutaways and ghosting : satisfying visibility constraints in dynamic 3D illustrations. *The Visual Computer*, 8(5-6) :292–302, September 1992.

[49] Sarah F. Frisken and Ronald N. Perry. Designing with distance fields. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, pages 60–66. ACM, 2006.

[50] Sarah F Frisken, Ronald N Perry, Alyn P Rockwood, and Thouis R Jones. Adaptively sampled distance fields : a general representation of shape for computer graphics. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, page 249–254, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[51] G. W Furnas. Generalized fisheye views. In *ACM SIGCHI Bulletin*, volume 17, pages 16–23. ACM, 1986.

[52] George W. Furnas and Benjamin B. Bederson. Space-scale diagrams : understanding multiscale interfaces. In *CHI '95 : Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 234–241. ACM & Addison-Wesley, 1995.

[53] John F Gantz and Christopher Chute. The diverse and exploding digital universe : An updated forecast of worldwide information growth through 2011. IDC, 2008.

[54] James J. Gibson. *The Ecological approach to visual perception*. Lawrence Erlbaum Associates, new edition edition, September 1986. Published : Paperback.

[55] Jonathan Grudin. Partitioning digital worlds : focal and peripheral awareness in multiple monitor use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, page 458–465, New York, NY, USA, 2001. ACM.

[56] Carl Gutwin. Improving focus targeting in interactive fisheye views. In *CHI '02 : Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 267–274. ACM, 2002.

[57] Carl Gutwin and Chris Fedak. A comparison of fisheye lenses for interactive layout tasks. In *GI '04 : Proceedings of Graphics Interface 2004*, GI '04, pages 213–220, 2004.

[58] M. Hachet, F. Decle, S. Knodel, and P. Guitton. Navidget for easy 3D camera positioning from 2D inputs. In *Proceedings of the 2008 IEEE Symposium on 3D User Interfaces*, 3DUI '08, pages 83–89, Washington, DC, USA, 2008. IEEE.

[59] I. Herman, G. Melancon, and M.S. Marshall. Graph visualization and navigation in information visualization : A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1) :24–43, 2000.

[60] Kenneth E. Hoff, III, John Keyser, Ming Lin, Dinesh Manocha, and Tim Culver. Fast computation of generalized voronoi diagrams using graphics hardware. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 277–286. ACM & Addison-Wesley, 1999.

[61] Kenneth E. Hoff, III, Andrew Zaferakis, Ming Lin, and Dinesh Manocha. Fast and simple 2d geometric proximity queries using graphics hardware. In *I3D '01 : Proceedings of the 2001 symposium on Interactive 3D graphics*, I3D '01, pages 145–148. ACM, 2001.

[62] Kasper Hornbæk, Benjamin B. Bederson, and Catherine Plaisant. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM ToCHI*, 9(4) :362–389, 2002.

[63] Edward W. Ishak and Steven K. Feiner. Interacting with hidden content using content-aware free-space transparency. In *Proceedings of the 17th annual ACM symposium on User interface software and technology - UIST '04*, page 189, New York, New York, USA, October 2004. ACM Press.

[64] Edward W. Ishak and Steven K. Feiner. Content-aware scrolling. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pages 155–158, New York, NY, USA, 2006. ACM.

[65] Edward Waguih Ishak and Steven Feiner. Content-aware layout. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '07, page 2459–2464, New York, NY, USA, 2007. ACM.

[66] Waqas Javed, Sohaib Ghani, and Niklas Elmqvist. Polyzoom : multiscale and multifocus exploration in 2d visual spaces. In *Proc. CHI '12*, CHI '12, pages 287–296, New York, NY, USA, 2012. ACM.

[67] Susanne Jul and George W. Furnas. Navigation in electronic worlds - workshop report. *SIGCHI Bulletin*, 29 :44–49, 1997.

[68] N Kadmon and E Shlomi. "A polyfocal projection for statistical surfaces". *The Cartographic Journal, 15(1) :36-41, June 1978.*, 1978.

[69] T. Alan Keahey and Edward L. Robertson. Nonlinear magnification fields. In *INFOVIS '97 : Proceedings of the 1997 IEEE Symposium on Information Visualization*, InfoVis '97, pages 51–58. IEEE, 1997.

[70] Azam Khan, George Fitzmaurice, Don Almeida, Nicolas Burtnyk, and Gordon Kurtenbach. A remote control interface for large displays. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, UIST '04, page 127–136, New York, NY, USA, 2004. ACM.

[71] Azam Khan, Ben Komalo, Jos Stam, George Fitzmaurice, and Gordon Kurtenbach. HoverCam : interactive 3D navigation for proximal object inspection. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 73–80, New York, NY, USA, 2005. ACM.

[72] Jens Kruger, Jens Schneider, and Rudiger Westermann. Clearview : An interactive context preserving hotspot visualization technique. *IEEE TVCG*, 12(5) :941–948, September 2006.

[73] Yair Kurzion and Roni Yagel. Interactive space deformation with hardware-assisted rendering. *IEEE Computer Graphics and Applications*, 17(5) :66–77, 1997.

[74] Pierre-Yves Laffont, Jong Yun Jun, Christian Wolf, Yu-Wing Tai, Khalid Idrissi, George Drettakis, and Sung-eui Yoon. Interactive content-aware zooming. In *Proc. GI*, pages 79–87. CIPS, 2010.

[75] Eric Lamar, Bernd Hamann, and Kenneth I. Joy. A magnification lens for interactive volume visualization. In *PG '01 : Proceedings of the 9th Pacific conference on Computer graphics and Applications*, PG '01, page 223. IEEE, 2001.

[76] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI '95 : Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 401–408. ACM & Addison-Wesley, 1995.

[77] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM ToCHI*, 1 :126–160, 1994.

[78] Wilmot Li, Maneesh Agrawala, Brian Curless, and David Salesin. Automated generation of interactive 3D exploded view diagrams. *ACM TOG*, 27(3) :101 :1–101 :7, 2008.

[79] Wilmot Li, Lincoln Ritter, Maneesh Agrawala, Brian Curless, and David Salesin. Interactive cutaway illustrations of complex 3D models. *ACM TOG*, 26(3), 2007.

[80] Julian Looser, Mark Billinghurst, and Andy Cockburn. Through the looking glass : the use of lenses as an interface tool for augmented reality interfaces. In *Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '04, New York, NY, USA, 2004. ACM.

[81] William E. Lorensen and Harvey E. Cline. Marching cubes : A high resolution 3d surface construction algorithm. In *SIGGRAPH '87 : Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 163–169. ACM, 1987.

[82] Jock D Mackinlay, George G Robertson, and Stuart K Card. The perspective wall : detail and context smoothly integrated. In *Proceedings of the SIGCHI conference on Human factors in computing systems : Reaching through technology*, CHI '91, pages 173–176. ACM, 1991. ACM ID : 108870.

[83] Jock D. Mackinlay, George G. Robertson, and Robert DeLine. Developing calendar visualizers for the information visualizer. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, UIST '94, page 109–118, New York, NY, USA, 1994. ACM.

[84] Sylvain Malacria, Eric Lecolinet, and Yves Guiard. Clutch-free panning and integrated pan-zoom control on touch-sensitive surfaces : the cyclostar approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 2615–2624, New York, NY, USA, 2010. ACM.

[85] James McCrae, Igor Mordatch, Michael Glueck, and Azam Khan. Multiscale 3D navigation. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, page 7–14, New York, NY, USA, 2009. ACM.

[86] Michael McGuffin and Ravin Balakrishnan. Acquisition of expanding targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, page 57–64, New York, NY, USA, 2002. ACM.

[87] Michael J. McGuffin, Liviu Tancau, and Ravin Balakrishnan. Using deformations for browsing volumetric data. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 401–409, Washington, DC, USA, 2003. IEEE.

[88] Tomer Moscovich, Fanny Chevalier, Nathalie Henry, Emmanuel Pietriga, and Jean-Daniel Fekete. Topology-aware navigation in large networks. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, page 2319–2328, New York, NY, USA, 2009. ACM.

[89] Mathieu Nancel, Julie Wagner, Emmanuel Pietriga, Olivier Chapuis, and Wendy Mackay. Mid-air pan-and-zoom on wall-sized displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 177–186. ACM, 2011.

[90] Ken Perlin and David Fox. Pad : an alternative approach to the computer interface. In *SIGGRAPH '93 : Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 57–64. ACM, 1993.

[91] E. Pietriga. A toolkit for addressing HCI issues in visual language environments. In *2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 145–152, 2005.

[92] Emmanuel Pietriga, Caroline Appert, and Michel Beaudouin-Lafon. Pointing and beyond : an operationalization and preliminary evaluation of multi-scale searching. In *CHI '07 : Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 1215–1224. ACM, 2007.

[93] Emmanuel Pietriga, Olivier Bau, and Caroline Appert. Representation-independent in-place magnification with sigma lenses. *IEEE TVCG*, 16(03) :455–467, 2010.

[94] Matthew Plumlee and Colin Ware. An evaluation of methods for linking 3D views. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, I3D '03, page 193–201, New York, NY, USA, 2003. ACM.

[95] Matthew Plumlee and Colin Ware. Integrating multiple 3d views through frame-of-reference interaction. In *Proceedings of the conference on Coordinated and Multiple Views In Exploratory Visualization*, CMV '03, pages 34–, Washington, DC, USA, 2003. IEEE.

[96] Matthew D. Plumlee and Colin Ware. Zooming versus multiple window interfaces : Cognitive costs of visual comparisons. *ACM ToCHI*, 13(2) :179–209, 2006.

[97] Paul Rademacher. View-dependent geometry. In *SIGGRAPH '99 : Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 439–446. ACM & Addison-Wesley, 1999.

[98] Ramana Rao and Stuart K. Card. The table lens : merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, page 318–322. ACM, 1994.

[99] George G. Robertson and Jock D. Mackinlay. The document lens. In *UIST '93 : Proceedings of the ACM Symposium on User Interface Software and Technology*, UIST '93, pages 101–108. ACM, 1993.

[100] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees : animated 3D visualizations of hierarchical information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, page 189–194, New York, NY, USA, 1991. ACM.

[101] Timo Ropinski and Klaus Hinrichs. Real-time rendering of 3D magic lenses having arbitrary convex shapes. In *Journal of the International Winter School of Computer Graphics (WSCG04)*, pages 379–386. Science Press, 2004.

[102] Manojit Sarkar and Marc H Brown. Graphical fisheye views of graphs. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '92, page 83–91. ACM, 1992.

[103] Manojit Sarkar and Marc H. Brown. Graphical fisheye views. *CACM*, 37(12) :73–83, 1994.

[104] Manojit Sarkar, Scott S. Snibbe, Oren J. Tversky, and Steven P. Reiss. Stretching the rubber sheet : a metaphor for viewing large layouts on small screens. In *UIST '93 : Proceedings of the ACM Symposium on User Interface Software and Technology*, UIST '93, pages 81–91. ACM, 1993.

[105] Doug Schaffer, Saul Greenberg, Lyn Y N Bartram, John Dill, Zhengping Zuo, Shelli Dubs, and Mark Roseman. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Transactions on Computer-Human Interaction*, 3(2) :162–188, 1996.

[106] Sagi Schein, Eran Karpen, and Gershon Elber. Real-time geometric deformation displacement maps using programmable hardware. *The Visual Computer*, 21(8) :791–800, 2005.

[107] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 151–160. ACM, 1986.

[108] B. Shneiderman. The eyes have it : a task by data type taxonomy for information visualizations. In *, IEEE Symposium on Visual Languages, 1996. Proceedings*, pages 336–343, 1996.

[109] Ben Shneiderman. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, IUI '97, page 33–39, New York, NY, USA, 1997. ACM.

[110] R. Spence. *Information visualization*. ACM Press Bks. Addison-Wesley, 2001.

[111] ROBERT SPENCE. A framework for navigation. *International Journal of Human-Computer Studies*, 51(5) :919–945, 1999.

[112] Martin Spindler, Marco Bubke, Tobias Germer, and Thomas Strothotte. Camera textures. In *GRAPHITE '06 : Proceedings of the 4th international Annual Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, GRAPHITE '06, pages 295–302. ACM, 2006.

[113] Desney S Tan, George G Robertson, and Mary Czerwinski. Exploring 3D navigation : combining speed-coupled flying with orbiting. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '01, page 418–425, New York, NY, USA, 2001. ACM.

[114] Greg Turk and James F O'Brien. Shape transformation using variational implicit functions. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 1999. ACM.

[115] BARBARA TVERSKY, JULIE BAUER MORRISON, and MIREILLE BETRANCOURT. Animation : can it facilitate ? *International Journal of Human-Computer Studies*, 57(4) :247–262, October 2002.

[116] `http://freecadweb.org/`.

[117] `https://www.leapmotion.com/`.

[118] `http://www.autodesk.com/products/autodesk-maya/overview`.

[119] `http://www.blender.org/`.

[120] `http://www.openstreetmap.org/`.

[121] `http://www.shanghai-272-gigapixels.com/`.

[122] `http://www.tfl.gov.uk/assets/downloads/standard-tube-map.pdf`.

[123] `http://www.tfl.gov.uk/assets/downloads/standard-tube-map.pdf`.

[124] John Viega, Matthew J. Conway, George Williams, and Randy Pausch. 3D magic lenses. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, UIST '96, page 51–58. ACM, 1996.

[125] Ivan Viola and Meister E. Gröller. Smart visibility in visualization. In *Proceedings of the First Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*, Computational Aesthetics'05, pages 209–216, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[126] Ivan Viola, Armin Kanitsar, M Eduard Gröller, and M E Groller. Importance-driven feature enhancement in volume visualization. *IEEE TVCG*, 11(4) :408–18, 2005.

[127] Huamin Wang, Peter J. Mucha, and Greg Turk. Water drops on surfaces. *ACM Trans. Graph.*, 24 :921–929, July 2005.

[128] Lujin Wang, Ye Zhao, Klaus Mueller, and Arie Kaufman. The magic volume lens : An interactive focus+context technique for volume rendering. In *Visualization, 2005. VIS 05. IEEE*, VIS '05, pages 367–374. IEEE, 2005.

[129] Yu-Shuen Wang, Tong-Yee Lee, and Chiew-Lan Tai. Focus+Context visualization with distortion minimization. *IEEE TVCG*, 14(6) :1731–1738, 2008.

[130] Yu-Shuen Wang, Chaoli Wang, Tong-Yee Lee, and Kwan-Liu Ma. Feature-preserving volume data reduction and Focus+Context visualization. *IEEE TVCG*, 17(2) :171 –181, February 2011.

[131] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. Guidelines for using multiple views in information visualization. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '00, page 110–119, New York, NY, USA, 2000. ACM.

[132] Matthew Ward, Georges G Grinstein, and Daniel Keim. *Interactive data visualization : Foundations, techniques, and applications*. AK Peters, 2010.

[133] Colin Ware. *Information visualization : perception for design*. Morgan Kaufmann, 2012.

[134] Colin Ware, Marlon Lewis, and O F Creativity. The DragMag image magnifier. In *Conference companion on Human factors in computing systems*, CHI '95 EA, page 407–408, New York, NY, USA, May 1995. ACM.

[135] Brian Wyvill, Craig McPheeters, and Geoff Wyvill. Animatingsoft objects. *The Visual Computer*, 2(4) :235–242, August 1986.

[136] G. Wyvill. Data structure for soft objects. *The visual computer*, 2(4) :227–234, 1986.

[137] Yonggao Yang, Jim X. Chen, and Mohsen Beheshti. Nonlinear Perspective Projections and Magic Lenses : 3D View Deformation. *IEEE Comput. Graph. Appl.*, 25(1) :76–84, 2005.

[138] Shumin Zhai, Stéphane Conversy, Michel Beaudouin-Lafon, and Yves Guiard. Human on-line response to target expansion. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, page 177–184, New York, NY, USA, 2003. ACM.

[139] Xin Zhao, Wei Zeng, David Gu, Arie Kaufman, Wei Xu, and Klaus Mueller. Conformal magnifier : A focus+context technique with local shape preservation. *IEEE TVCG*, 2012. PrePrint.