JellyLens: Content-Aware Adaptive Lenses

Cyprien Pindat, Emmanuel Pietriga, Olivier Chapuis, Claude Puech

JellyLens: Content-Aware Adaptive Lenses

Cyprien Pindat[1,2] pindat@lri.fr
Emmanuel Pietriga[2,3,1] emmanuel.pietriga@inria.fr
Olivier Chapuis[1,2] chapuis@lri.fr
Claude Puech[3,2,1] claude.puech@u-psud.fr

[1]Univ Paris-Sud & CNRS (LRI) 91405 Orsay, France
[2]INRIA 91405 Orsay, France
[3]INRIA Chile (CIRIC) 7561211 Santiago, Chile

Figure 1. Magnifying the Lido in Venice. (a) 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; (b) 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; (c) a JellyLens automatically adapts its shape to the region of interest, magnifying as much relevant information in the focus region as (b) while better preserving the context: surrounding islands are left almost untouched from (a).

ABSTRACT

Focus+context lens-based techniques smoothly integrate two levels of detail using spatial distortion to connect the magnified region and the context. Distortion guarantees visual continuity, but causes problems of interpretation and focus targeting, partly due to the fact that most techniques are based on statically-defined, regular lens shapes, that result in far-from-optimal magnification and distortion. JellyLenses dynamically adapt to the shape of the objects of interest, providing detail-in-context visualizations of higher relevance by optimizing what regions fall into the focus, context and spatially-distorted transition regions. This both improves the visibility of content in the focus region and preserves a larger part of the context region. We describe the approach and its implementation, and report on a controlled experiment that evaluates the usability of JellyLenses compared to regular fisheye lenses, showing clear performance improvements with the new technique for a multi-scale visual search task.

ACM Classification Keywords

H.5.2 [Information interfaces and presentation]: User Interfaces – Graphical user interfaces.

Author Keywords

Multi-scale Interfaces, Focus+Context, Lenses

C. Pindat, E. Pietriga, O. Chapuis and C. Puech. JellyLens: Content-Aware Adaptive Lenses. In UIST '12: Proceedings of the 25th Symposium on User Interface Software and Technology, ACM, 2012.

INTRODUCTION

In many domains, datasets grow in size much more rapidly than computer displays in resolution. This includes maps produced by geographical information systems: rasterizing the OpenStreetMap world map at street level would require an $18 \cdot 10^{15}$ pixel bitmap; satellite or astronomical imagery: Spitzer's recent infrared survey of the inner part of our galaxy is made of thousands of frames stitched together to produce a 4.7 billion pixel image; images several dozens of gigapixels in size assembled from multiple frames taken by conventional SLR cameras; as well as numerous information visualizations such as complex networks represented as node-link diagrams. None of these datasets fit on computer screens, not even high-resolution wall displays. They require multi- scale navigation capabilities that will enable users to transition from high-level overviews to zoomed-in, highly-detailed representations of a region of interest, and conversely.

Multi-scale navigation in large information spaces is typically achieved using one of the following three interface schemes or combination thereof [15]: overview+detail, zooming and focus+context. Techniques based on the latter integrate a detailed representation of a region of interest directly in the surrounding context [17], which essentially corresponds to the original, unmagnified visualization showing more data at a lower scale. One of the main focus+context techniques is the magnification lens and its derivatives, that provides in-place magnification integrated into the context.

Smooth integration of the two levels of detail is generally achieved using spatial distortion to connect the magnified region and the context, as with the so-called fisheye lenses.

While distortion guarantees visual continuity, it also causes problems of interpretation [12, 13], focus targeting [1, 18, 30] and virtual navigation. Indeed, most techniques are based on lens shapes defined statically by distance functions obtained through $L(P)$-metrics [14], 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 1-a); or the lens is big, showing a larger portion of the region of interest at the expense of the context (Figure 1-b). Moreover, distortion impedes comprehension, even more so when the lens' and objects' shapes differ significantly.

We present interactive focus+context techniques called *JellyLenses*, that dynamically adapt to the geometry of object(s) of interest. JellyLenses 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 1-c). The first technique, *PathLens*, consists of a lens attached to the mouse cursor, that adapts its shape, circular by default, to the graphical objects considered of interest based on the cursor's location. 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 [38]. The second technique, *AreaLens*, consists in the dynamic relayout and resizing of all objects that fall in the lens' scope. The technique tries to preserve the original aspect of those objects as much as possible, magnifying those closest to the cursor while reducing those closer to the lens' periphery, and distorting the regions in-between. As discussed later, the two techniques are complementary, providing solutions to a wide range of tasks and visual configurations. Both work for arbitrary 2D datasets, ranging from networks and maps displayed as vector graphics (Figure 2) to documents and Web pages. They also apply to satellite imagery (Figure 1) and other pixel-oriented datasets, as detailed later in this paper.

After an overview of related work, we describe 1) a method for modeling lenses of arbitrary shape, used in all JellyLenses to enable rendering of magnified regions of interest at interactive frame rates, and 2) two methods for dynamically generating relevant lens shapes based on the geometry of regions of interest, that respectively correspond to the PathLens and AreaLens techniques. We then report on a controlled experiment that evaluated the usability of JellyLenses compared to regular fisheye lenses, showing clear performance improvements with the new technique.

RELATED WORK

Distortion-oriented visualizations often rely on metaphors inspired by the physical world: magnifying glasses [32, 18], stretchable rubber sheets [34] and, more generally, surface deformations [11]. Other techniques work with more fundamental concepts: hyperbolic projection [26], non-linear magnification fields [22] or complex logarithmic views [8].

Early systems made the distortion extend to the boundaries of the representation [26, 27, 32, 33, 34], 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 [19, 30]. Such lenses, usually termed *constrained lenses*, can be created using 3-dimensional pliable surfaces [11] and the framework for unifying presentation space [14, 13], non-linear magnification fields [22], conformal mapping [42], or the Sigma Lens framework [1, 30].

Magic Lens filters [6] 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 [11, 42], including the recent undistort lenses [9].

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 [11]. LaMar *et al.*'s magnification lenses [25] are based on homogeneous texture coordinates and special geometries. They can be applied to both 2D and 3D representations but are limited in the type of distortion and lens shapes they can model. Non-linear perspective projections [41] 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 [10], 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 [3, 36] to view-dependent geometry [31] and deformation based on hardware-accelerated displacement mapping [16, 35] and deflectors [23]. 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 [37] 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 [39] 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.* [7] 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 JellyLenses have conceptual similarities: they both make use of the geometry of particular objects of interest in the visualization to adapt the distortion.

Finally, JellyLenses are conceptually related to the numerous content-aware image resizing techniques that have emerged recently, from seminal work on seam carving [2] to Laffont *et al.*'s image zooming technique [24]. 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.

GENERAL APPROACH

JellyLenses magnify a subregion of the visualization and smoothly integrate it in the context by locally compressing the area immediately surrounding the region of interest. But as opposed to previous techniques, JellyLenses dynamically adapt to better match the geometry of the region of interest. Thus, they must be able to take arbitrary shapes (Figure 2).

Adapting a JellyLens to match the geometry of the region of interest is a three-step process. The first step consists of obtaining information about the geometry 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. The techniques to achieve this are readily available off-the-shelf, but this particular problem is heavily application-domain dependent and thus not discussed in detail in this paper. At this point we simply emphasize that a JellyLens 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 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 and our first contribution, defines how the lens behaves as it gets repositioned by the user in the visualization using the mouse. In Figure 2, the lens adapts its shape to match that of the portion of the object of interest (the itinerary) around the mouse cursor positioned in **P**. We present two methods of adaptation: PathLens, better suited to contours and paths at large, illustrated in Figure 2; and AreaLens, better suited to the magnification of filled shapes, illustrated in Figure 6.

The last step is the rendering of the region seen through the lens. This modeling and rendering method is used by both PathLens and AreaLens to adapt the lens in real-time at interactive frame rates. This second contribution is described in the next section, before the two adaptation methods.

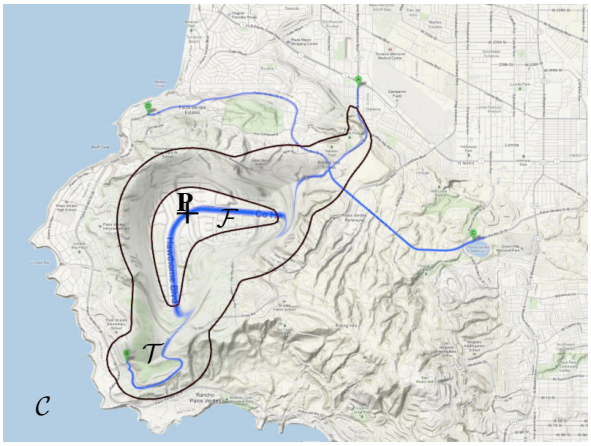


Figure 2. Regions defined by a constrained lens: focus region $\mathcal{F}$ magnifying the region of interest, context region $\mathcal{C}$, and smooth transition $\mathcal{T}$ between $\mathcal{F}$ and $\mathcal{C}$ achieved through distortion.

JELLYLENS MODELING AND RENDERING

Constrained lenses are defined by (Figure 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 $\mathbf{P}$ 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.

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:

$$\mathbf{x} = T_F(\mathbf{x}') = \mathbf{P} + \frac{\mathbf{x}' - \mathbf{P}}{\mu} \tag{1}$$

The transformation is formulated as a reverse mapping. For each pixel coordinates $\mathbf{x}$ 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:

$$T(\mathbf{x}) = \frac{w_F(\mathbf{x}) \cdot T_F^{-1}(\mathbf{x}) + w_C(\mathbf{x}) \cdot \mathbf{x}}{w_F(\mathbf{x}) + w_C(\mathbf{x})}, \forall \mathbf{x} \in \mathcal{T} \tag{2}$$

where weights are defined as follows:

$$w_F = \left(\frac{1}{dist_F(\mathbf{x})}\right)^{b_F}, w_C = \left(\frac{1}{dist_{\mathcal{C}}(\mathbf{x})}\right)^{b_C} \tag{3}$$

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 [14]: *"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.

The above method makes it possible to generate a wide variety of lens shapes. It can be further generalized to create lenses with multiple foci (flat-tops). Multiple foci can be useful when magnifying disjoint objects of interest that are in close proximity (Figure 5). They make it possible, with a single lens, to have disjoint yet undistorted magnified regions of interest, and are an important feature of AreaLenses.

Let $\mathcal{F}_i$ ($i \in [1, n]$) be n regions of interest. Each region is associated with a transformation T_i with its own $\mathbf{P}_i$ and μ_i. The general mapping function is then a weighted average over the contributions of all involved regions, conceptually similar to the weighted coordinate transformation of multiple pairs of lines in field morphing [5]:

$$T(\mathbf{x}) = \frac{\sum_{i=0}^{n} w_i \cdot T_i^{-1}(\mathbf{x})}{\sum_{i=0}^{n} w_i} \tag{4}$$

$$with \quad w_i = \left(\frac{1}{dist_i(\mathbf{x}) + a_i}\right)^{b_i} \tag{5}$$

where a_i are constants 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 6. 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. Each parameter b_i has an influence, on the corresponding $\mathcal{F}_i$ region, similar to that of the above parameters b_C and b_F on their respective regions.

We implemented this algorithm in C++ and OpenGL, delegating the spatial transformation to programmable graphics hardware. Mapping function T (Equation (4)) 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 [20], 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 [12].

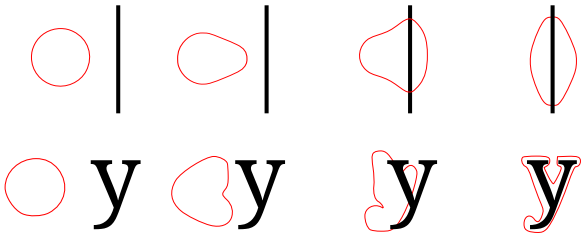


Figure 3. PathLens morphing effect.

ADAPTATION

We present two methods of adaptation, that both rely on the above modeling and rendering method. The first resulting interaction technique, PathLens, consists of a deformable lens, circular by default, that dynamically morphs its shape to roughly adopt that of the closest object of interest. The technique features a single flat-top, and is better suited to magnifying paths. The second technique, AreaLens, considers all regions of interest within a given *area of influence* around the lens' focal point (mouse cursor). Each region gets scaled piecewise to make room for the regions of higher importance, in an attempt to preserve the content of all regions from being significantly distorted.

PathLens

PathLenses assume a default circular shape that gets combined with implicit descriptions of the geometry of some objects in the scene based on distance fields. The actual shape of a PathLens at a given focal point $\mathbf{P}$ in the visualization depends on what objects of potential interest are in the vicinity. As illustrated in Figure 3, when far away from any object, a PathLens adopts the default circular shape, and progressively morphs, conserving the same area, as it approaches an object of interest. Not all objects in the scene are necessarily taken into account, and each application can define what objects the lens will adapt to. For instance, on Google Maps™, a PathLens could be made to adapt to interstate roads only, ignoring any highway, service road or other geographical landmark such as parks and water bodies.

We define the shape by an implicit function, $f(\mathbf{x})$, as the set of all points $\mathbf{x}$ such that $f(\mathbf{x}) = s$, s being an iso-level. Tesselation of implicit shapes is achieved using the marching square algorithm [28].

The cursor position and the geometry of nearby objects of interest both contribute to the definition of the lens' final shape. These influences are represented in terms of contribution functions: $lens(\mathbf{x})$ represents the contribution of the default lens shape to the final shape, $data(\mathbf{x})$ represents the contribution of proximal objects to the final shape. The shape is thus implicitly represented by:

$$f(\mathbf{x}) = lens(\mathbf{x} - \mathbf{c}) + data(\mathbf{x}) \tag{6}$$

where $\mathbf{c}$ is the cursor's coordinates. The $lens$ contribution is defined as:

$$lens(\mathbf{x}) = \frac{1}{||\mathbf{x}||} - ||\mathbf{x}|| \tag{7}$$

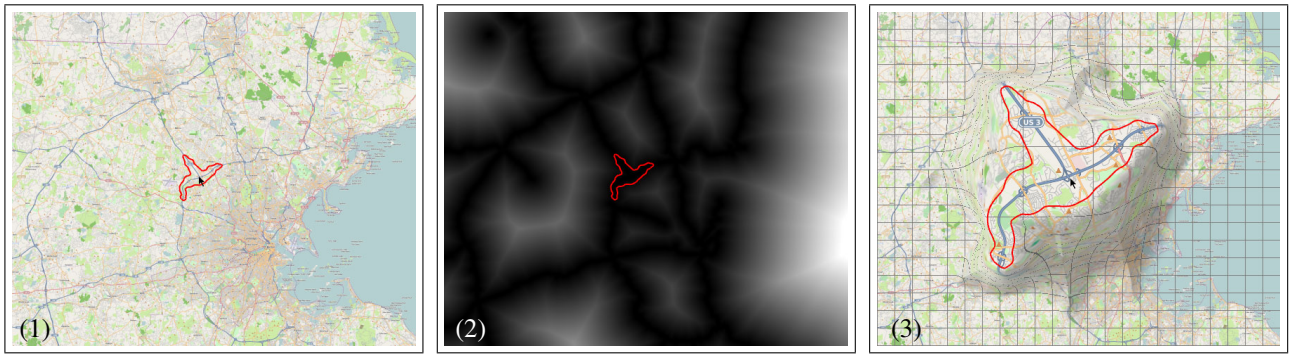


Figure 4. Illustration of the adaptation and rendering pipeline for PathLens: the region of interest is identified as a function of cursor position (1) and distance fields (2); the lens is then adapted to match this shape and rendered with a shading effect (3) – actual rendering displayed to the user, red contour excepted.

We use an adaptation of the function described in [40] to define $data$:

$$data(\mathbf{x}) = G\left(\frac{D(\mathbf{x})}{\beta}\right) \tag{8}$$

where $D(\mathbf{x})$ is the distance between $\mathbf{x}$ and the closest object of interest; G being defined as:

$$G(r) = \begin{cases} 2r^3 - 3r^2 + 1, & if \;\; r < 1; \\ 0 \;\; \text{otherwise} \end{cases} \tag{9}$$

and β being a parameter bounding the contribution of $data$ to a given radius of influence.

Moving away from the object, as $D(\mathbf{x})$ increases, the contribution of $data(\mathbf{x})$ drops smoothly to zero, the lens reverts to its default circular shape.

We can then define the following set for each cursor position $\mathbf{c}$ and each iso-level s composed of several connected components:

$$E(\mathbf{c}, s) = \{\mathbf{x} \mid lens(\mathbf{x} - \mathbf{c}) + data(\mathbf{x}) > s\} \tag{10}$$

Because a small region around the cursor always yields large values (Equation (7)), it is guaranteed that among the set of all connected components of $E(\mathbf{c}, s)$ there will always be one that encompasses the cursor.

Let $S(\mathbf{c}, s)$ be its boundary. Then $S(\mathbf{c}, s)$ is the shape implicitly defined by the field function of Equation (6) at iso-level s. As s decreases, the shape's area, denoted $area(S(\mathbf{c}, s))$, increases. As mentioned earlier, we want the lens to have a constant area A. This is achieved by means of a numerical minimization method applied to:

$$u(s) = (area(S(\mathbf{c}, s)) - A)^2 \tag{11}$$

We use a gradient descent method, with the gradient defined as follow, P_i ($i \in [1, n]$) being the n vertices of the tessellation of the shape:

$$u'(s) = -2 \times \sum_{i=0}^{n} \frac{||P_i - P_{i+1}||}{||\vec{\nabla} f(P)||} \times (area(S(\mathbf{c}, s)) - A) \tag{12}$$

The above method is used to compute both the focus and context regions' contours. Computation of the context region's contour must also take into account the focus region's magnification by a factor of μ and make the former large enough to fully encompass the latter, so as to avoid intersecting contours that would entail discontinuities in the final result.

As the user repositions the lens in the visualization, $\mathcal{E}(\mathbf{c}, s(\mathbf{c}))$ is likely to eventually merge with another connected component from $E(\mathbf{c}, s(\mathbf{c}))$. This can lead to unwanted discontinuities in the lens' shape between the frame where this happens and the next. This effect can be avoided thanks to parameter β introduced earlier. To understand the influence of this parameter, let us first consider the case when β tends towards infinity. The contribution of function $data$ is then flat, leading to $E(\mathbf{c}, s(\mathbf{c}))$ being composed of only one connected component. When β tends towards zero, the iso-level of the field function shrinks (Equation (6)), and tends to fit the object's geometry more accurately, but $E(\mathbf{c}, s(\mathbf{c}))$ is then composed of several connected components. 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 depend on thedistance from the cursor position to the object, $D(\mathbf{P})$,ranging from a minimum value as $D(\mathbf{P})$ tends towards 0 and a maximum value as $D(\mathbf{P})$ increases and reaches a maximum radius of influence.

AreaLens

While PathLenses only consider the closest object of interest during the adaptation process, AreaLenses consider all objects of interest within a given area of influence. This makes the second technique better suited to the magnification of filled shapes in dense scenes, while PathLenses are better suited to magnification of paths or filled shapes in sparse scenes (in terms of regions of interest). In the following, regions of interest are referred to as *objects* for the sake of brevity and to emphasize their piecewise processing.

An AreaLens affects each object that falls into its area of influence. Objects that fall outside of it correspond to the context and are left untouched. As illustrated in Figure 5, 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.

As the user moves the mouse cursor, objects of interest in the area of influence are either smoothly pulled towards the

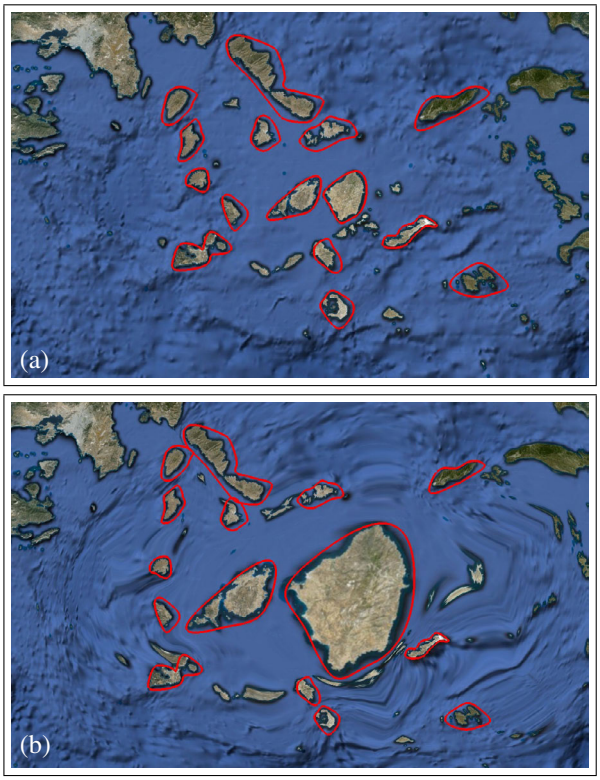


Figure 5. (a) Aegean islands (no lens applied). Red contours indicate regions of interest. (b) an AreaLens preserves the islands' appearance (piecewise magnification), putting most of the distortion in the ocean.

cursor as it approaches them, or pushed away as it gets away from them, eventually reverting to their original location and size when getting out of the area of influence. Achieving this behavior entails the following requirements. Magnified objects should not overlap each other (*R1*); objects should move in a smooth manner (*R2*); the topology of the information space should be preserved as much as possible, i.e., objects preserve their relative positions (*R3*); and finally, objects beyond the area of influence, i.e., that belong to the context region, should neither be moved nor scaled (*R4*).

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 (*R4*). We use two concurrent mapping algorithms to achieve the other three requirements: a *dispersion* mapping, and a *magnification* 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 *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 objects* from overlaping 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 overlap anymore.

The *magnification* mapping rescales objects by applying the following transformation to their initial position, where $\mathbf{P}$ is the focal point (cursor) and μ the lens' magnification factor: $T(\mathbf{x}) = \frac{\mathbf{x}-\mathbf{P}}{\mu} + \mathbf{P}$. The closer to the cursor, the more an object gets influenced by the *magnification* mapping. This is achieved through an iterative process. Initially, every object is set to its initial position in the scene, and is considered as a *movable object*. We then process objects one by one, according to their proximity to the cursor (closest one first). Each object is set to its final position: $w \times T(\mathbf{x}) + (1 - w) \times \mathbf{x'}$ with $w = 1 - \frac{dist}{radius}$ where $\mathbf{x}$ is the initial position, $\mathbf{x'}$ the current position, $radius$ a parameter specifying the lens' operating range and $dist$ its distance to the cursor. The object is then added to the *magnified objects* set, and the *dispersion* mapping is applied to all remaining *movable objects*. The iterative process stops when there are no more *movable objects* within range.

The mapping algorithm results in the displacement of each movable object. Each object being associated with a rigid transformation T_i, movable objects in their final position are then used as regions for distortion modeling and rendering.

The above method works well for scenes that contain both low and high numbers of objects, provided that they feature some empty space to absorb the distortion. When objects are densely packed, or even juxtaposed, the method can be adapted to allow for some distortion of movable objects, thanks to parameter a_i associated with each flat-top (introduced earlier). Figure 6 illustrates this case: objects correspond to the different neighborhoods in central Los Angeles. By setting a_i to 0 on the object directly under the mouse cursor (Hollywood), we make sure that this particular neighborhood gets fully magnified, without distortion. The surrounding regions get assigned a_i values greater than 0, allowing the lens to distort those regions if necessary.

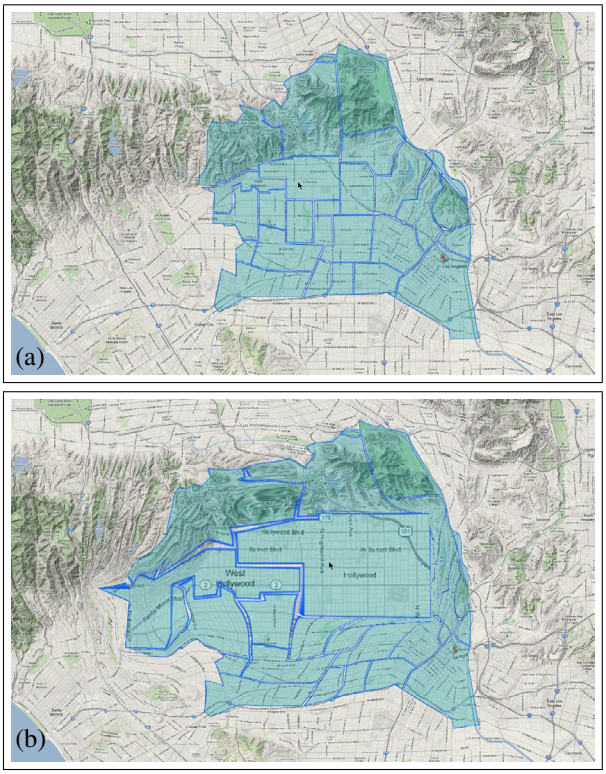


Figure 6. (a) Map of Los Angeles neighborhoods (no lens applied). (b) AreaLens adapting to Hollywood and part of West Hollywood.

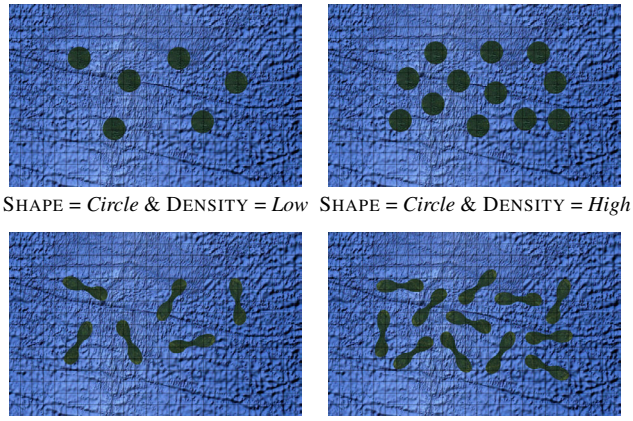
EVALUATION

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 [15, 21, 29]. Our primary goal was 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 number of objects.



SHAPE = *Circle* & DENSITY = *Low* SHAPE = *Circle* & DENSITY = *High*

SHAPE = *Blob* & DENSITY = *Low* SHAPE = *Blob* & DENSITY = *High*

Figure 7. Screenshots for the four SHAPE × DENSITY conditions.

Task and Procedure

In all trials, the information space was made of islands on an ocean (Figure 7). 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 7-top) or elongated blobs (*Blob*, Figure 7-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 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 7-left); and a *High*er density (Figure 7-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 [29]. 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.

Quantitative Results

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×SHAPE×DENSITY×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.

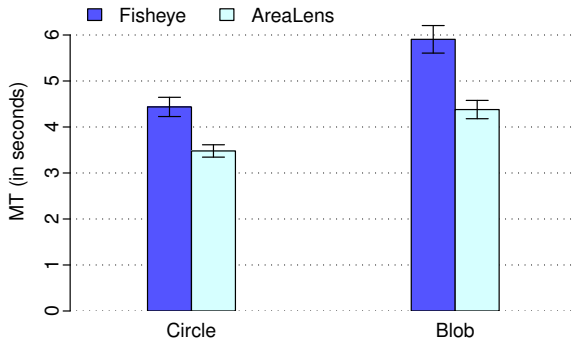


Figure 8. *MT* per TECH by SHAPE. Error bars show the 95% confidence limit.

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×SHAPE ($F_{1,11} = 13.3$, $p = 0.0039$) that can be observed in Figure 8. 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×DENSITY nor a TECH×SHAPE×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.

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 distract-

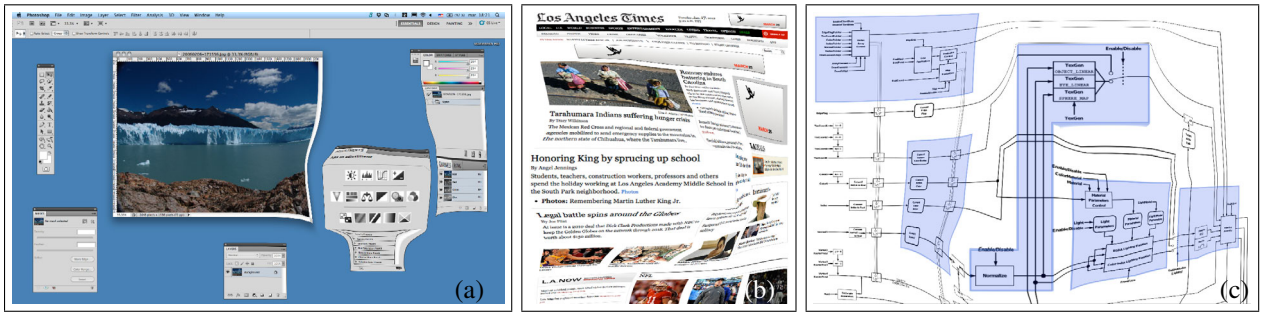


Figure 9. The AreaLens technique applied to: (a) magnifying a group of widgets in a user interface (Adobe Photoshop); (b) the Los Angeles Times Web homepage; and (c) a part of the OpenGL state machine block diagram.

ing; 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*.

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.

DISCUSSION AND FUTURE WORK

Our implementation of JellyLenses can be applied to arbitrary 2D datasets, provided that objects of interest can be identified in the data. 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 and 4 illustrate the use of JellyLenses on vector graphics, that make geometry information readily available. Figures 1-c and 5 show lenses applied 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.

JellyLenses 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 9-a), or to magnify portions of Web pages such as a newspaper homepage (Figure 9-b). The technique makes it possible to display an overview of the full page, and to browse it at a readable level, somewhat like Fishnet [4] 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 9-c 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.

In this paper we break the usual behavior of fisheye lenses by proposing to dynamically change their geometry while preserving visual continuity between the focus and context regions. Our empirical evaluation shows that this approach has strong potential, though this is of course only a first step. More evaluation and more case studies are needed to better understand the advantages and weakness of the approach and of each technique. For instance, it would be interesting to evaluate the PathLens technique in path tracing tasks, and to better understand possible visual distractions caused by the various types of shape (objects of interest) and layout of these shapes. This might lead to novel adaptation techniques that use transformations that go beyond magnification (e.g., rotation). Another avenue for future work is to investigate tech-

niques that would allow users to specify objects of interest on-the-fly, e.g., by sketching, involving them more explicitly in the adaptation process, and to couple adaptive lenses with the recently introduced undistort lenses [9].

REFERENCES

1. Appert, C., Chapuis, O., and Pietriga, E. High-precision magnification lenses. In *Proc. CHI '10*, ACM (2010), 273–282.
2. Avidan, S., and Shamir, A. Seam carving for content-aware image resizing. *ACM TOG 26*, 3 (2007).
3. Barr, A. H. Global and local deformations of solid primitives. In *Proc. SIGGRAPH '84*, ACM (1984), 21–30.
4. Baudisch, P., Lee, B., and Hanna, L. Fishnet, a fisheye web browser with search term popouts: a comparative evaluation with overview and linear view. In *Proc. AVI '04*, ACM (2004), 133–140.
5. Beier, T., and Neely, S. Feature-based image metamorphosis. In *Proc. SIGGRAPH '92*, ACM (1992), 35–42.
6. Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T. D. Toolglass and magic lenses: the see-through interface. In *Proc. SIGGRAPH '93*, ACM (1993), 73–80.
7. Böttger, J., Brandes, U., Deussen, O., and Ziezold, H. Map warping for the annotation of metro maps. *IEEE Comput. Graph. Appl. 28* (2008), 56–65.
8. Böttger, J., Preiser, M., Balzer, M., and Deussen, O. Detail-In-Context visualization for satellite imagery. *Computer Graphics Forum 27*, 2 (2008), 587–596.
9. Brosz, J., Carpendale, S., and Nacenta, M. The undistort lens. *Computer Graphics Forum 30* (2011), 881–890.
10. Brosz, J., Samavati, F. F., Carpendale, S., and Sousa, M. C. Single camera flexible projection. In *Proc. NPAR '07*, ACM (2007), 33–42.
11. Carpendale, M. S. T., Cowperthwaite, D. J., and Fracchia, F. D. 3-dimensional pliable surfaces: for the effective presentation of visual information. In *Proc. UIST '95*, ACM (1995), 217–226.
12. Carpendale, M. S. T., Cowperthwaite, D. J., and Fracchia, F. D. Making distortions comprehensible. In *Proc. VL '97*, IEEE (1997), 36–45.
13. Carpendale, M. S. T., Ligh, J., and Pattison, E. Achieving higher magnification in context. In *Proc. UIST '04*, ACM (2004), 71–80.
14. Carpendale, M. S. T., and Montagnese, C. A framework for unifying presentation space. In *Proc. UIST '01*, ACM (2001), 61–70.
15. Cockburn, A., Karlson, A., and Bederson, B. B. A review of overview+detail, zooming, and focus+context interfaces. *ACM CSUR 41* (2009), 2:1–2:31.
16. Correa, C. D., and Silver, D. Programmable shaders for deformation rendering. In *Proc. GH '07*, EA (2007), 89–96.
17. Furnas, G. W. Generalized fisheye views. In *ACM SIGCHI Bulletin*, vol. 17, ACM (1986), 16–23.
18. Gutwin, C. Improving focus targeting in interactive fisheye views. In *Proc. CHI '02*, ACM (2002), 267–274.
19. Gutwin, C., and Fedak, C. A comparison of fisheye lenses for interactive layout tasks. In *Proc. GI '04* (2004), 213–220.
20. Hoff, III, K. E., Zaferakis, A., Lin, M., and Manocha, D. Fast and simple 2d geometric proximity queries using graphics hardware. In *Proc. I3D '01*, ACM (2001), 145–148.
21. Hornbæk, K., Bederson, B. B., and Plaisant, C. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM ToCHI 9*, 4 (2002), 362–389.
22. Keahey, T. A., and Robertson, E. L. Nonlinear magnification fields. In *Proc. INFOVIS '97*, IEEE (1997), 51–58.
23. Kurzion, Y., and Yagel, R. Interactive space deformation with hardware-assisted rendering. *IEEE Computer Graphics and Applications 17*, 5 (1997), 66–77.
24. Laffont, P.-Y., Jun, J. Y., Wolf, C., Tai, Y.-W., Idrissi, K., Drettakis, G., and Yoon, S.-e. Interactive content-aware zooming. In *Proc. GI*, CIPS (2010), 79–87.
25. Lamar, E., Hamann, B., and Joy, K. I. A magnification lens for interactive volume visualization. In *Proc. PG '01*, IEEE (2001), 223.
26. Lamping, J., Rao, R., and Pirolli, P. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proc. CHI '95*, ACM & Addison-Wesley (1995), 401–408.
27. Leung, Y. K., and Apperley, M. D. A review and taxonomy of distortion-oriented presentation techniques. *ACM ToCHI 1* (1994), 126–160.
28. Lorensen, W. E., and Cline, H. E. Marching cubes: A high resolution 3d surface construction algorithm. In *Proc. SIGGRAPH '87*, ACM (1987), 163–169.
29. Pietriga, E., Appert, C., and Beaudouin-Lafon, M. Pointing and beyond: an operationalization and preliminary evaluation of multi-scale searching. In *Proc. CHI '07*, ACM (2007), 1215–1224.
30. Pietriga, E., Bau, O., and Appert, C. Representation-independent in-place magnification with sigma lenses. *IEEE TVCG 16*, 03 (2010), 455–467.
31. Rademacher, P. View-dependent geometry. In *Proc. SIGGRAPH '99*, ACM & Addison-Wesley (1999), 439–446.
32. Robertson, G. G., and Mackinlay, J. D. The document lens. In *Proc. UIST '93*, ACM (1993), 101–108.
33. Sarkar, M., and Brown, M. H. Graphical fisheye views. *CACM 37*, 12 (1994), 73–83.
34. Sarkar, M., Snibbe, S. S., Tversky, O. J., and Reiss, S. P. Stretching the rubber sheet: a metaphor for viewing large layouts on small screens. In *Proc. UIST '93*, ACM (1993), 81–91.
35. Schein, S., Karpen, E., and Elber, G. Real-time geometric deformation displacement maps using programmable hardware. *The Visual Computer 21*, 8 (2005), 791–800.
36. Sederberg, T. W., and Parry, S. R. Free-form deformation of solid geometric models. In *Proc. SIGGRAPH '86*, ACM (1986), 151–160.
37. Spindler, M., Bubke, M., Germer, T., and Strothotte, T. Camera textures. In *Proc. GRAPHITE '06*, ACM (2006), 295–302.
38. Wang, H., Mucha, P. J., and Turk, G. Water drops on surfaces. *ACM Trans. Graph. 24* (July 2005), 921–929.
39. Wang, Y.-S., Lee, T.-Y., and Tai, C.-L. Focus+context visualization with distortion minimization. *IEEE TVCG 14*, 6 (2008), 1731–1738.
40. Wyvill, G. Data structure for soft objects. *The visual computer 2*, 4 (1986), 227–234.
41. Yang, Y., Chen, J. X., and Beheshti, M. Nonlinear Perspective Projections and Magic Lenses: 3D View Deformation. *IEEE Comput. Graph. Appl. 25*, 1 (2005), 76–84.
42. Zhao, X., Zeng, W., Gu, D., Kaufman, A., Xu, W., and Mueller, K. Conformal magnifier: A focus+context technique with local shape preservation. *IEEE TVCG* (2012). PrePrint.