Browsing Large Image Datasets through Voronoi Diagrams

Paolo Brivio, Marco Tarini, and Paolo Cignoni

Fig. 1. An example of a thumbnail bar, shaped as a belt that partially encircles the title of this paper. It features irregular shaped thumbnails with gradually decreasing sizes, starting from the currently selected image (in the middle). Shown thumbnails represent a dynamic subset of a large dataset composed of a few thousands of images; the selected subset is adaptive, denser around the selected image and respectful of existing image hierarchies. When a new image is selected, the thumbnails dynamically rearrange around the new selection in a configuration preserving all the above characteristics and without breaking temporal continuity.

Abstract—Conventional browsing of image collections use mechanisms such as thumbnails arranged on a regular grid or on a line, often mounted over a scrollable panel. However, this approach does not scale well with the size of the datasets (number of images). In this paper, we propose a new thumbnail-based interface to browse large collections of images. Our approach is based on weighted centroidal anisotropic Voronoi diagrams.
A dynamically changing subset of images is represented by thumbnails and shown on the screen. Thumbnails are shaped like general polygons, to better cover screen space, while still reflecting the original aspect ratios or orientation of the represented images. During the browsing process, thumbnails are dynamically rearranged, reshaped and rescaled. The objective is to devote more screen space (more numerous and larger thumbnails) to the parts of the dataset closer to the current region of interest, and progressively lesser away from it, while still making the dataset visible as a whole. During the entire process, temporal coherence is always maintained. GPU implementation easily guarantees the frame rates needed for fully smooth interactivity.

Index Terms—Visualization System and Toolkit Design, Scalability Issues, User Interfaces, Zooming and Navigation Techniques.

1 Introduction

Images are a wide-spread medium of communication which has enormously grown with the advent of inexpensive digital cameras. Recently, very large datasets of pictures are made available (e.g. from Web, or targeted photographic campaigns). Browsing the images is a central step in all application scenarios. Common image browsers are based on equally-sized thumbnails, arranged on a regular grid or on a line and displayed over a scrollable panel. It is widely recognized that this approach, as is, does not scale well with the number of images [22, 2, 26, 21, 6, 32, 15, 28, 27, 10].

This paper introduces a novel image browsing mechanism in which thumbnails of images are dynamically packed inside a "thumbnail area", covering but a fraction of the screen. The arrangement of the thumbnails is based on weighted anisotropic Voronoi diagrams. This allows to closely fill a freely shaped thumbnail area with a large number of significant images, as shown in the teaser.

1.1 Application scenarios

Results of photographic campaigns in Cultural Heritage: time-stamp is a natural choice for the total ordering, but is not the only valid one. Massive external image calibration like [28], also freely available from Web-services [31, 20], can provide reliable external calibrations (camera roto-translation for each shot). The latter can be used to define a more meaningful ordering on the images than just time-stamp: e.g. based on a path connecting viewpoints [27], or on the photographed region [7]. Image calibrations also define image up-directions.

Large collection of images from the Web: like those provided by on-line picture management and sharing applications (e.g. Flickr). Natural total orderings in this case are derived from information on authors, tags, time-stamps, and so on (or combinations).

Results of Web-searches: the Web can be queried for its contents focusing on images (e.g. Google-image). Resulting image-datasets can be ordered by time-stamp, relevance, or by content-based or context-based analysis.

Personal image collections: combining time-stamp, original folder structure, tagging, and the like [2, 9] provides a natural ordering.

In each of the above cases, an importance for each image can be assessed by analyzing the images content or by evaluating their quality; for example, images can be clustered using any content-based or tag-based technique, and selecting important image representatives as centroids of those clusters.

Aspect ratio is often not uniform within the dataset. Even when pictures are taken from similar cameras, aspect ratio can vary within the dataset, due to "portrait/landscape" orientation differences (it can even change drastically, e.g. in presence of a few "panoramic" images).

1.2 Input datasets

The proposed approach is not limited to a specific application scenario as long as the image-dataset follows few characteristics:

Total ordering: one (or more) total ordering is assumed to be defined among images in the dataset. In other words, it is always defined which of any two images comes "before" the other. A total ordering also implicitly defines relative distance between images (two images are distant as much as the number of images that appear in the ordering between them). We say that a total ordering is meaningful in the sense that such distance is usually and roughly related to semantic similarity.

Varying representativeness: some images can be tagged as more representative than other images, i.e. some images can be used as exemplars of all the images in a small neighborhood. This means that

- *Paolo Brivio and Marco Tarini are with University of Insubria, Varese - Italy, E-mail: {paolo.brivio,marco.tarini}@uninsubria.it.*
- *Paolo Cignoni is with ISTI-CNR, Pisa - Italy, E-mail: cignoni@isti.cnr.it.*

Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

it is also possible to exploit any kind of previously existing hierarchical or clustering organization of the images.

Conversely, we do not make assumptions on the characteristics of the image-datasets, such as:

Size: large datasets (e.g. thousands of images) must be dealt with.

Aspect ratios: images aspect ratio can be either constant over the dataset, or different for each image. Optionally, the system is capable of exploiting further information, if available, such as per-image *up directions*. These can be specified for each image and dictate the 2D orientation along which the image has to be shown in the interface. This orientation has not to be necessarily a multiple of 90 degrees.

In this paper, we propose a new, Voronoi driven, dynamic browsing mechanism in which thumbnails represent images inside a "thumbnail area". Before sketching the idea in Sec. 3, we overview related work.

2 Related work

Conventional thumbnail-based image browsers (e.g. [9]) use a regular grid layout showing all images, often mounted over scrollable panels. However, this approach has several drawbacks: when there are too many images, entire sections of the dataset are far beyond sight, unless thumbnails are made excessively small. Browsing for an image, for example, can result wearing. This problem is aggravated when the thumbnail bar covers only a fraction of the screen, for example, when the main screen area displays the selected image at full resolution. Secondarily, a regular grid layout enforces uniform thumbnail aspect ratio: when the images aspect ratio is not uniform, this causes suboptimal thumbnail cropping, distortions, or uneven gaps among thumbnails, which wastes screen space and can be perceived as unpleasant.

The challenge of more efficiently browsing a large image dataset has generated a vast literature of different solutions and approaches. The problem can be faced in a number of ways – complete analysis of all the presented strategies, interfaces and techniques is beyond the scope of this paper.

Most of the literature presents solutions that are in some sense more specific or orthogonal to our Voronoi driven dynamic browsing scheme, describing techniques that either are specific to some application field, or providing algorithms for ordering pictures that can be exploited by our approach.

For example, the retrieval of a specific image, while connected to browsing, is a well separated task. Thus, our technique does not compete with methods which approach this problem by adding features like annotation and speech recognition [21, 6, 26, 32, 29]. Image processing techniques can also be used to enhance queries and tag-sharing among images [23]. Similar approaches could be plugged in our thumbnail browser; however, it is worth remembering that some user studies seem to indicate that generic users may prefer simpler GUI-based navigation systems which require less user feedback [24, 8].

Similarly, the task of organizing and building structures over existing sets of images should be considered a separate problem. For example, in PhotoSpread [15] tag information is used to organize images into an extended version of a spreadsheet and to support subsequent tag-queries and image comparisons; in PhotoScope [33] the authors consider also space and time relationships, to the specific issue of providing an overview system for construction management.

Specifically, the technique we propose can make effective use of a representativeness value for the images (see Sec. 3). Therefore, the vast literature on techniques that allow to build clusters of images [14] is relevant, as hierarchical structures can be employed to generate representativeness values to be plugged in our approach. In [22], images are firstly grouped by time of shot and color analysis. In PhotoMesa [2] images are clustered using treemaps, while its zooming interface allows the user to look for a particular image by searching inside each image group. This strategy has also been tested on PDAs [16], in which case the browser cannot scale well due to too small screen resolution. Even though most of the above schemes use also time and tags as ordering/clustering function, images generally carry metadata which can be used to compute different organizations: owner, approximate location, and other shot parameters [30].

Moreover, recent advances in Computer Vision techniques make viable a new category of ordering and clustering approaches, based on the possibility to reconstruct a rough 3D representation of the depicted scene from a set of photos. The original position and directions of shooting of all photos can be recovered too and used to help organization and navigation of the photos. For instance, [27] observe that shots (i.e. calibrated images) tend to cluster along certain paths, compute them from image calibration and use these paths as a guide during navigation. Using calibrated images, [7] suggest a novel way to cluster images based on scene semantics, introducing the notion of geo-relevance. For each calibrated image, they reproject its content into a volume and score each voxel as a function of the number of images in which it is visible. Then, images are organized into a tree in which the root corresponds to the whole represented scene, while in-depth paths focus on different parts of the scene.

Other approaches still exploit joint 2D-image / 3D-model (i.e. image-based) visualization as a way to enhance user experience [28, 20, 10, 13]. At each frame, they show only the locally interesting set of (2D-3D) data. In PhotoTourism [28] a 2D plan map also shows an overall scheme of the scene, but this is practical when the scene is not sparse. Otherwise, the image calibration cannot be performed. For instance, browsing the dataset of pictures of a week holiday could be uncomfortable even if the calibrations were provided.

A recent trend focuses on browsing using hand-held devices. In [3] the browsing interface is based on multiple navigation bars (one for images, one for "linearized" shots distance, and one for frequency). Dynamic changing the ordering scheme can also guide navigation, as proposed by [17]. Additionally, specific hardware features of recent mobile devices can be used as a mean to perform browsing [5]. In mobile devices context, however, bandwidth and hardware considerations limit the amount of data that can be downloaded and, thus, the number of contemporarily visible images and performed computations.

The above techniques are designed for some specific scenarios. Conversely, in this paper we consider a more general setting, abstracting from specific usages and presenting a general tool for efficient and compact browsing of a large number of images. Similarly to us, other approaches focus on general techniques for displaying thumbnails.

First of all, there are the already cited approaches that exploit hierarchical quadtree like approaches (like for example [2]). With respect to our technique, these are more constrained in terms of hierarchy-representation and do not have an explicit way of managing both a linear ordering and an importance tagging of the images. Moreover, they are designed for a browsing-centered interface, where the thumbnail area is significantly large and, in all the presented examples, of rectangular shape. On the other hand, our approach is much more flexible in terms of the thumbnail area shape. Another approach that has similar limitations was presented in PhotoHelix [11], where a bi-manual gesture-based system is introduced to time-order images, group them by continuous sequences, and display them in a spiral. Users are expected to interact with the spiral arrangement to browse images at different scales, eventually grouping them in different manners. Beside the previous points, this solution requires specialized hardware.

3 Approach overview

The proposed dynamic browsing mechanism is based on a *'thumbnail area'* featuring image thumbnails, whose number, shape, position, and size change dynamically and smoothly during the browsing process.

This thumbnail area is intended to be embedded as an element of a GUI, such as a bar confined at the bottom or at the top of the screen, or on a L-shaped area around a screen corner, etc. This leaves most the screen available for the rest of the application. For example, in a photo browser, the main area of the screen can be occupied by a full-sized representation of the currently selected photo. In applications where a 3D model is shown as well as 2D images, the main area can be devoted to the 3D rendering (in these applications, the currently selected photo is usually linked to the 3D view position of its shot [28, 20, 4]).

This thumbnail area can be arbitrarily shaped and sized (and even dynamically reshaped – allowing for "liquid" interfaces). Curved or irregular shaped thumbnail-areas can be used, and still all of the space

devoted to them is profitably used, making this a flexible tool in the design of a GUI.

In our browsing approach we assume that, as commonly happens, one specific image has the role of the main image. We will refer to that image as the current *'focus'*. For example, the focus can represent the selected image currently displayed at full resolution in the main area. Clicking on any thumbnail is a natural way to select that image as a new focus; another natural way offered by the system is by dragging thumbnails with a pointer device (see Sec. 5.2). Other application-dependent ways to select a new focus include: selecting an image by text search; reverting to a previously bookmarked image; navigating the dataset along its total ordering by means of a key-based interface (e.g. via "next image" and "previous image" buttons); and so on.

Since the total ordering of the image is assumed to be meaningful, at any given moment it is reflected by the images spatial ordering in the thumbnail area (see Sec. 4.4). For example, in a horizontal thumbnail area the focus will probably be placed in the center, images preceding the focus (in the total order) will be always displayed on its left, and the ones following it on its right. In the same example, we do not regard the vertical positioning as meaningful (in other words, vertical positioning has no associated semantic and is freely optimized by the system to achieve good arrangements).

Clearly, when large image datasets are visualized, only a subset of all the images can be effectively presented as thumbnails to the end user. We term *'active'* images those being shown with a thumbnail on the screen. Images immediately around the current focus (w.r.t. the total ordering) should always be active, but away from it progressively fewer and fewer images should be displayed. The concept is that, in large datasets, exhaustively displaying of all images is only important around the current center of interest: away from the focus a progressively larger number of dataset images are not directly shown as thumbnails, but rather implicitly represented by some other image.

Another important element of our browsing scheme is that images closer to the focus deserve more space in the thumbnail area, so that more of their internal detail is visible. Conversely, the farther an image is from the focus, the smaller space its thumbnail gets. Also, coherently with the spatial ordering, thumbnails representing images closer to the focus are in a more central part of the thumbnail area, while images far from it are relegated to peripheral areas.

A first challenge we need to face is how to achieve a good arrangement of the thumbnails for active images, in a way that the above desiderata on position, size, activation status and so on are fulfilled.

The second challenge is related to temporal coherence. When a new focus is selected, applying the set of rules above determines a general rearrangement which potentially affects every thumbnail on screen. For each image, in fact, the distance from the focus changes and, consequently, so do its activation status, prescribed position and size. We want to switch to the new configuration with a fairly quick but continuous animation, without breaking temporal coherence (which would clearly harm usability).

Only when the new focus is extremely far from the previous one, so that the subset of active images is almost completely different, or when there are alternative total orderings in the dataset and user switches from one to another, then temporal continuity becomes meaningless and should explicitly be broken; in that case, reinitializing the entire thumbnail area directly to a good configuration is a simple solution.

In our approach, the key element to achieve temporal coherence and the rest of the objectives is to resort to a custom variation of Voronoi diagrams and Lloyd relaxation. We use Voronoi diagrams to partition the thumbnail area into *'regions'*, and each thumbnail corresponds to a region. When a region moves and changes size during relaxation, the associated thumbnail follows it. The diagram as a whole is steered (e.g. driving shapes, sizes and positions of the regions) by carefully changing a set of control parameters of the diagram (as region weights).

Paper structure: in the rest of this section we describe how, given a current focus, a set of active images is selected (Sec. 3.1), and how their ideal thumbnail sizes are prescribed (Sec. 3.2). The best answer to these questions varies greatly depending on the applicative context. Rather than proposing a specific solution, we show how this framework can accommodate different choices of an interface designer.

Then, Sec. 4 details the characteristics of a Voronoi diagram (and its relaxation) which we use to enforce requirements in a temporally coherent way; after that, Sec. 5 shows how thumbnails are fitted inside the resulting regions. Finally, Sec. 6 sketches a few details about the implementation, which must be enough efficient to ensure full interactivity. A conclusion and result Section closes the paper.

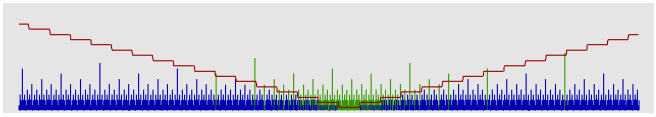

Fig. 2. Visual illustration of the criterion used to determine active images. Each image is shown as a vertical blue or green bar, displayed from left to right according to the total ordering defined on the image dataset. Each bar length is proportional to the representativeness assigned to that image. The red curve represents the selecting function g, centered on the current focus and sliding with its position. Images intersected with the curve (in green) are selected as active. The farther away images are from the focus, the more sparse active images are.

Fig. 3. Illustration of how thumbnails are arranged in the thumbnail area. Top: thumbnails are color-coded by how many missing images they currently "represent" (see Fig. 2), plus themselves. For illustration, each thumbnail is numbered in its center with its rank in the total ordering, so missing (non active) images are visible as gaps in the numeration.

3.1 Determining which images are active

As already stated, more images must be active near the current focus. Another needed constraint is monotonicity, needed to avoid unnecessary switches of the activation status: an active image can only become non active if the focus moves farther away, never if it gets closer.

To address this problem by assume that, for each image, we have a static value denoting its *'representativeness'*: a high value means that the image will be active even when the focus is far. In this case it will stand for other images nearby that are not active for the time being.

Next, we use a function $g : \mathbb{N} \rightarrow \mathbb{N}$ that dictates the minimal representativeness that an image must posses in order to become active, as a function of the distance of its index from the focus index. Thumbnails are tested against g starting from the focus and expanding simultaneously in both directions: when a predefined upper limit on the number of active images is met, remaining images are set as not active.

We used a $g(i)$ which is simply linear with i, and specifically $g(i) = |2 \cdot K \cdot log(N) \cdot i/N|$, N being the total number of images. Any other function can be used, as long as it is monotonically increasing, and $g(0) = 0$ and $g(1) = 0$ (so that an image is shown at least when it is the focus or when it is next to the focus). This function customizes how many images should be discarded away from the focus. For example, setting g as a constant 0, a solid interval of images around the focus is shown, skipping none, as in standard browsing interfaces. If g is set to be 1 for $i \geq 2$, it means that only even images are shown, except for the two adjacent to the current focus.

Fig. 2 presents a visual representation of this algorithm and Fig. 3 the arrangement of thumbnails resulting from the subset of active images resulting from its application. Note that this mechanism allows for images to be activated (or deactivated) between two images that are already active, and not only at the extrema of the active image set. Depending on the choice of g, this can be the most common situation.

It is worth nothing that representativeness values are not necessarily linked in any way to the actual content of the images; the only important thing is that they are appropriately scattered over the dataset. In fact, if no further a priori information is available, it is appropriate to assign well distributed and entirely arbitrary representativeness values as follows: if an image is the i-th image of the dataset according to a zero indexed total ordering, you can assign it a representativeness value equal to the number of trailing zeros of the binary representation of $i+1$. This rule generates a simple power-of-two distribution of representativeness values like the one depicted in Fig. 2.

On the other hand, this mechanism is quite general and it can accommodate, within limits, the information that might be available on the dataset about the meaningfulness of each image. In fact, we would like to underline that every clustering technique and approach for building automatic hierarchical organization of images can be used to generate valid representativeness values by simply using the distance from the farthest descendant leaf.

3.2 Prescribing thumbnail sizes

In order to prescribe the ideal size of each thumbnail, we dynamically assign a varying *'importance'* measure to each active image, as a scalar value expressed in an arbitrary measure that is intended to be directly proportional to the on-screen area of the thumbnail. The actual prescribed area of a region for an active image with importance a is given by the total thumbnail area multiplied by the ratio of a over the summed importance of all active images.

Importance for active image i is determined by the number k_i of currently active images separating, in the total image ordering, that image from the focus. The relationship between k_i and the importance value is tabled, and can be an arbitrary function. This should be considered a parameter in hand of the interface designer; this is analogous to the choice of thumbnail size in a standard thumbnail based browser, and it likewise depends on many factors, including the shape of the thumbnail area, the screen resolution and the typical image content, as well as the application dependent scenario.

For example, importance can be set to be linearly decreasing with k (so that many small images are visible far from the center, and fewer bigger ones nearer to the focus); else, it can be larger for $k = 0$ and constant for all other k (focus is larger and all other thumbnails are equally sized); and so on. A possibility is to award a maximal importance to all $k \leq M$ and a much smaller one to any other k, so that the first few M images on either side of the focus are as large as it, for example to let the user easily compare consecutive images (see for example bottom two images of Fig. 7, where $M = 1$ and 2 respectively).

4 Adapted Voronoi diagrams

A Voronoi diagram [1] is defined for set of points, called sites, and consists of a partition of a planar area into one region for each site, so that every point inside a region is closest to the site of that region than to any other site.

A Voronoi diagram can be evolved by iteratively moving each site in the barycenter of its region and then recomputing the partition for the new sites, a process named Lloyd relaxation. Lloyd relaxation is known to converge to a configuration where the regions are roughly equal sized and well distributed over the area. Away from the borders, the final partition tends to resemble regular tilings, while closer to the borders, it nicely adapts to the shape of the contours.

In the following subsections, we show how the definition of Voronoi diagram and Lloyd relaxation are adapted to our purposes, with a combination of small, targeted modifications, so that the result can be used to efficaciously drive the movements of thumbnails. Specifically, we will discuss the mechanism we use to: add appropriate weights so to ensure that the area prescriptions are met (4.1); insert and remove regions without breaking temporal coherence (4.2); add anisotropy so to achieve regions with shapes and proportions better matching those of the corresponding images (4.3); enforce coherent spatial ordering of the sites at all times (4.4); improve convergence configurations and reach them faster (4.5); robustly avoid oscillating behaviors that can arise due to a few of these modifications (4.6); optionally, allow regions to bulge a little out of the thumbnail area, if deemed appropriate in the context of the designed GUI (4.7).

4.1 Weight balancing

In order to enforce regions with different areas (see Sec. 3.2), each region is *weighted* differently. Weighted Voronoi diagrams are well understood and can be defined in several alternative ways (refer to [1]). We choose to use a *power diagram*, i.e. a Voronoi diagram where the distance of a point p to a site s_i of weight w_i is defined as $|p - s_i|^2 - w_i$. This choice has the advantage of producing convex regions and straight borders between regions (barring anisotropy, see Sec. 4.3).

While a region area is guaranteed to monotonically increase when its weight is increased, it would be difficult to explicitly predict which weight must be used to result in the prescribed area for that region. In order to bypass this problem, the actual region areas, resulting from current weightings, are explicitly measured (during Lloyd relaxation): weights are adjusted for the next iteration, according to detected area excess or defect. The amount of change in weight adjustment is made proportional to the area excess/defect, and bounded with a max-resize-speed parameter M_s (we used M_s as five percent of the total thumbnail-area size per second).

This way the system quickly converges to the weights that locally produce the sizes prescribed by the varying importance function. This mechanism also guarantees time-coherency: each region importance never changes abruptly, while the region shrinks/enlarges to the new size with continuity (and with controlled speed).

4.2 Inserting and removing regions

What is described above also applies to the case when a previously non active image is made active and its importance is set to a positive value, as a result of a focus change (according to what is described in Sec. 3.1). A new region for that image is added to the diagram, but its initial weight is set to zero, guaranteeing that the region will grow in place progressively pushing away neighbor regions.

We also need to determine an initial position for the site. This choice is not too critical, as the new region will quickly find an appropriate spot, thanks to Lloyd relaxation, regardless of the initial position. However, a good strategy is to average the current site positions of the regions corresponding to the active images immediately following and preceding the new region in the total order. When these two regions happen to be adjacent, as is usually the case, this means to spawn the new region in or near the border separating them. Note that this initial position is also coherent with the spatial ordering imposed to the region (see Sec. 4.4). In the relatively rare cases where either the following or the preceding active image is missing, we spawn the new site at the appropriate extreme of the thumbnail-area.

Likewise, when a previously active image is made no longer active, the corresponding region is not directly removed from the diagram. Instead, the image importance is zeroed, so that the region weight will start decreasing and neighboring regions can *progressively* expand over it. Only when its detected area is negligible, it is removed.

4.3 Regions anisotropy

Images which are (for example) longer than tall should be represented by a similarly shaped thumbnail. Also, in datasets where 3D external calibration of images is available, it is convenient that the thumbnails are rotated in 2D so that the world-space up direction matches the screen-space up direction (i.e. so that the skies inside the images are always on top, all buildings are always vertical, etc, regardless of how the camera was oriented when the shot was taken).

Therefore, thumbnails should be accommodated inside a Voronoi region with a roughly matching shape and orientation. This way

thumbnails allow for a better usage of screen space, diminishing the magnitude of needed cropping/resizing of the thumbnails (see Sec. 5.1). It also serves as a useful visual indication, for the end user, of the original image orientation and aspect ratio.

To achieve this result we resort to a custom anisotropic version of Voronoi diagrams (similarly to [18]).

Formally, we redefine the distance functions used in the Voronoi diagram so that it is different for each region: a distance function d_i is associated to each region i (and a point p belongs to the region of site s_i if and only if, for any other site s_j, $d_i(p,s_i) < d_j(p,s_j)$).

Each region i is beforehand associated to two orthogonal 2D vectors $\vec{h}_i$ and $\vec{v}_i$, respectively describing horizontal and vertical directions of the thumbnail in screen space. Vectors modules are such that $|\vec{h}_i|/|\vec{v}_i|$ is the x/y aspect ratio of the corresponding images, and $|\vec{h}_i| \cdot |\vec{v}_i| = 1$.

Distance of a point p from site i is given by

$$d_i(p) = \left| \begin{pmatrix} \vec{v}_i \\ \vec{h}_i \end{pmatrix} (p - s_i) \right|^2 - w_i$$

The net effect is that regions tend to be elongated in the prescribed direction by the prescribed amounts. Three scenarios can then take place, depending on the dataset:

1. all images of the datasets share the same aspect ratio and orientation: then the same $\vec{h}_i$ and $\vec{v}_i$ are used for every i;
2. images of the dataset differ in aspect ratios, like the case of datasets with identical sized images that are either "portrait" or "landscape" oriented, or containing panoramas; in this case, vectors $\vec{h}_i$ and $\vec{v}_i$ are all respectively horizontal and vertical in screen space, but their norms are swapped for regions with portrait thumbnails;
3. each thumbnail must be rotated differently by an arbitrary angle, (and aspect ratios can be different): then $\vec{h}_i$ have arbitrary directions.

In all three cases, the relaxation tends, in practice, to result in good arrangements of differently shaped regions (see Fig. 4), even if this is not strictly guaranteed to always be the case. Note that in cases 2 and 3 the boundaries between the regions can become slightly curved. We consider this a small and acceptable drawback.

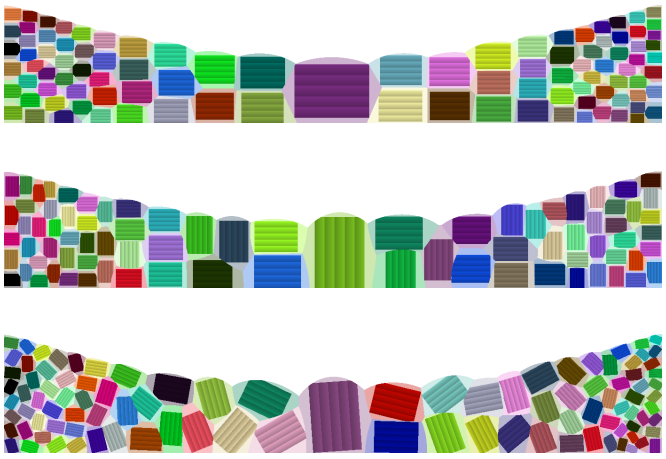

Fig. 4. Thanks to anisotropic Voronoi diagram relaxation, thumbnail arrangement can adapt to images aspect ratios and up orientations. Top: a dataset consisting of images with uniform aspect ratio and orientation. Middle: a dataset consisting of equally sized images with mixed portrait/landscape orientations. Bottom: a dataset where world space up direction information is available and used to rotate each region. For illustration purposes, image content inside thumbnails has been replaced by random colors, striped (horizontally in image space) to emphasize original image orientation, and the corresponding Voronoi regions are shown in a lighter shade of the same color (refer to Sec. 5.1 for info about how thumbnails are fitted inside regions).

4.4 Enforcing spatial ordering

Thumbnails must be spatially ordered inside the thumbnail area in a way that reflects the total ordering of the image dataset.

For example, consider a thumbnail-area shaped as a horizontal bar. Whenever image A precedes image B in the total ordering, the thumbnail for A is expected to always be on the left of thumbnail B (whereas in the vertical direction we choose not to constrain positions).

The order enforcing mechanism is simple: all active thumbnails are scanned following the total image ordering, and whenever the x coordinate of two consecutive thumbnails is not coherently oriented, they are swapped (leaving them at the same y). This operation is applied once after each relaxation step. Mis-ordered thumbnail couples are detected early so the jump is small, guaranteeing temporal coherence.

To generalize this mechanism for general shapes, thumbnail-areas are parametrized, so that a main direction is defined. When a planar region A is defined as shape for the thumbnail area, we also define a corresponding parametrization function $\phi : (u,v) \to (x,y) \in A$, and its inverse ϕ^{-1}, with $\phi^{-1}(x,y) = (\phi_u^{-1}(x,y), \phi_v^{-1}(x,y))$ (see Fig. 5). Function ϕ^{-1} maps A into a parametric planar region where the horizontal coordinate (that is, u) defines the intended visual ordering direction. In other words, two consecutive thumbnails, inside regions of sites (x_0,y_0) and (x_1,y_1) respectively, must be such that $\phi_u^{-1}(x_1,y_1) > \phi_u^{-1}(x_0,y_0)$. If that is not the case, the u parametric coordinates are swapped, leaving the v coordinates unaffected:

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \leftarrow \phi \begin{pmatrix} \phi_u^{-1}(x_1,y_1) \\ \phi_v^{-1}(x_0,y_0) \end{pmatrix}$$

(and vice-versa for (x_1,y_1)).

In the horizontal bar example, ϕ is just the identity. If a vertical bar is to be used with top to bottom ordering, then ϕ simply swaps coordinates, and $\phi(u,v) = (v,u)$. It is easy to define custom shapes for the thumbnail-areas together with the corresponding ϕ parametrization functions. In fact, most shapes allow for trivial parametrization and many interesting shapes can be designed by simply deforming a rectangle, so that the parametrization is available by construction.

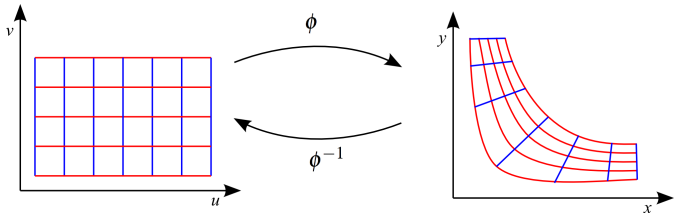

Fig. 5. An L-shaped thumbnail-area (right), and its parameterization.

4.5 Improving convergence

Regions tend to assume the appropriate positions inside the thumbnail-area as an indirect consequence of the almost regular arrangements induced by the Lloyd relaxation. For example, in a symmetrical setup on a horizontal bar, the *currently selected* region will almost invariably move to the center, because in the final configuration it will have as many regions on its right as on its left (due to enforced ordering), and each region on the left will have a corresponding one on the right of the same "importance" and therefore size.

While this works in the vast majority of cases, occasionally that expectation is not met (due to discretization errors, see Sec. 6, or to other factors). Other times the final configuration is met but only after too many iterations.

To solve both problems, we introduce an extra force per region that pushes each region site toward the intended final position on the x axis (or, in non horizontal thumbnail bar, in the u parametric direction, see Sec. 4.4). In our experience, it is enough to apply this force to the central region only. The final position of the $i-th$ active thumbnail is computed beforehand as a function of i from the importance distribution function (see Sec. 4.1) and thumbnail area shape.

Regions that must disappear on either one end of the thumbnail-area (see Sec. 4.2) are pushed toward the respective border.

4.6 System stabilization

Weight adaptations and ordering enforcements (Sections 4.1 and 4.4) can occasionally cause the system to oscillate near convergence configurations. Any of the configurations is close enough to convergence to be acceptable, but the oscillations themselves can be distracting. This can be solved in many practical ways. In our prototype we adopt a simple but robust heuristic: oscillations are detected and quickly stopped by progressively inhibiting further movements. For each region i we keep track of the time averaged (vectorial) displacement movement $\vec{m}_i$ of its site and the (scalar) traveled distance t_i, as follows. Before each relaxation iteration, a region i with a site which was moved from its previous position by a displacement $\vec{d}$ is updated with:

$$\vec{m}_i = k\vec{d} + (1-k)\vec{m}_i \qquad\qquad t_i = k|\vec{d}| + (1-k)t_i$$

where parameter k dictates how fast oscillations must be stopped (e.g. $k = 0.1$). Small values of ratio $|\vec{m}_i|/t_i \in [0,1]$ signal that the region is undergoing oscillations. That ratio is used as a multiplicative factor reducing (or stopping) site movements.

4.7 Soft thumbnail-area boundaries

As noted, the thumbnail-area can be of any custom shape, and for example it can have curved as well as straight boundary lines. In a regular Voronoi diagram, these boundary lines are strictly maintained, and points external to the thumbnail-area are never assigned to any region. Alternatively, we can choose to weaken this condition to let regions expand a little outside the area. Let p be a point external to the area, at a distance d_A from its border and at a distance d_i from the closest site s_i. Point p is assigned to region i if $d_i < f_b(d_A)$, and to no region otherwise. The border function f_b defines the shape of the border. We use $f_b(d) = K_0 - d/K_1$, with two positive parameters K_0 and K_1. Setting K_0 to 0 means that regions are kept strictly inside the thumbnail area. Otherwise, regions are allowed to "bulge out" of it a little in response to a region being pressed against a boundary. Increasing K_0 and K_1 augments this effect: their product bounds the maximal linear distance at which a region can extend outside the original thumbnail-area; K_1 controls how much curved the part which bulges out is (see Fig. 6).

This mechanism offers an additional degree of customizability to the proposed tool.

5 Browsing with adapted Voronoi diagrams

5.1 Filling regions with thumbnails

During the rendering of the thumbnail-area, actual thumbnails (smaller versions of the original images) must be fitted inside regions defined by the evolving Voronoi diagram.

By construction, each site is positioned roughly in the center of the corresponding region, so a natural choice is to place the thumbnail central point in that location. If up-directions are defined, thumbnails are always rotated accordingly, around their center.

The zoom factor used for each thumbnail is linked to the current area of the corresponding region. It is computed as the squared root of the ratio between the current area of the corresponding region and the (fixed) extension of the thumbnail (here we consider thumbnails as having unit area). This way, when a region increases its size (e.g. during its creation, or as a consequence of its increased importance), the thumbnail inside it automatically grows larger, and vice-versa.

In regular thumbnail bars, thumbnails are typically obtained by a combination of resizing and cropping of the original images: depending on the application scenario, different amounts of cropping can be used; for example, if the compositions of the pictures as a whole are important, images should be only resized, whereas if the the region of interest is usually found on the center of the images, it is beneficial to balance resizing and cropping, so that more details are left visible.

In our scenario, this consideration assumes a different meaning. All thumbnail zoom factors are further multiplied by a total factor S, which can be set between 0.5 and 1.50. Remember thumbnails are constrained to only cover their irregular shaped region: the parts of the thumbnails falling outside the region are just discarded. Therefore, using larger values for S means to effectively crop the images with an irregular (and time-varying) mask. This tends to cut out the boundaries of the image, and especially its corners. Reducing the parameter S, thumbnails progressively shrink inside a region, leaving gaps between neighbors. In short, parameter S designates the desired balance between cropping (with irregular masks) ad downsizing of images. Fig. 8 shows a small gallery of results obtained varying the S parameter.

Using small values of S, thumbnails are well separated by gaps, the entire content of each picture is visible, and the original image shape is gave away the thumbnail shape. In this set-up, the Voronoi diagram serves to determine dynamic thumbnail positions and zooming factors, rather than to partition the plane.

Using large values of S more screen area is used to show image content, as less space is wasted in gaps, and details of central part of the images are more visible. Non rectangular thumbnails are also appealing in sight of the recent advances in content based generation of thumbnails [25] where the portion (and shape) of the image chosen to be part of the thumbnail is driven by saliency. That can be ideal in scenarios where important parts of image tend to lie in the middle of the image rather than in its peripheral parts, and even less in corners.

With largest values, thumbnails form an irregular tiling fully covering the designed area (even then, tiles tend to have an aspect ratio that still reflects the one of the original image - see Sec. 4.3). The problem of losing a clear separation between thumbnails can be fixed: tiny lines can be added to separate them, or thumbnail areas can be "shaded" near region borders cropping them with a custom color like black or white (see Fig. 8, bottom two images). This separation shading can be enlarged and made stronger until thumbnails become elliptical shaped, which can be a good compromise as they look well separated, while still achieving good packing (see Fig. 8, bottom).

It should be noted that, due to irregularities of the relaxing Voronoi diagrams, even small values of S are not guaranteed to prevent that thumbnails occasionally touch or collide (when this happens, the consequence is just that small corners are cut). Similarly, larger values of S do not guarantee that no gaps arise between thumbnails (if required, these small gaps can be easily filled extending the thumbnails by replicating its border colors).

5.2 Basic Pointer Based Interactions

As mentioned, *clicking* on a thumbnail causes it to become the new focus (thus triggering the implied thumbnails activation status, position rearrangements, size changes, etc); *dragging* a thumbnail around with the pointer device triggers two concurrent effects (see attached video).

First, during the drag action, the Voronoi site corresponding to the dragged thumbnail is simply constrained to the current pointer position. This causes appropriate repositioning of neighboring regions (due to the ongoing Lloyd relaxation) which can well persist after the action. This mechanism allows the user to quickly rearrange (non permanently) the local disposition of thumbnails, by means of smaller drag actions. For example this is a mean to move two similar thumbnails closer so to compare them.

Second, whenever the drag action extent is larger than a threshold, the focus image changes accordingly. When the dragging action begins, we record current focus index i_F and the current horizontal position x_i of all active images i (assuming an horizontal thumbnail bar). When the horizontal position of the dragged thumbnail j is closer to x_h than to any other site, the new focus switches to $(i_F - h + j)$. This mechanism allows a user to scroll the thumbnail bar by means of slightly wider drag actions, to select the current focus simply by dragging a thumbnail into the current focus position, and so on.

6 Implementation details

6.1 Computing Lloyd relaxation and Voronoi diagrams

In order for this technique to be usable, Voronoi diagrams and Lloyd relaxation must execute in real time (depending on the context, they must leave enough computational power to the rest of the application).

Fig. 6. The effect of three different choices of parameters K_0 and K_1 determinining the behavior of regions at the boundary of the thumbnail area (a detail is shown). Left: with $K_0 = 0$ regions are strictly confined inside the thumbnail area boundary; middle: $K_0 = 0.25, K_1 = 0.5$, regions adjacent to boundary are allowed to "bulge out" a bit; right: $K_0 = 0.5, K_1 = 1$ the effect is more evident (in the three cases, arrangement of interior regions is also slightly different, because, as border regions invade some of the area outside the thumbnail area, they free space inside it).

After [12], Voronoi diagrams are computed leveraging the GPU: each region is rendered as a 3D paraboloid (a rotation of a convex parabola around its axis), with apex corresponding to sites and axis parallel to the depth direction, truncated at a maximal depth. The standard depth-test mechanism results in the correct region assignment of each screen pixel. Pixels are assigned a color ID identifying the region.

Additive weightings (see Sec. 4.1) are obtained by lifting the paraboloid up or down along the depth direction. Region anisotropy (see Sec. 4.3) is achieved by reshaping the paraboloid base as ellipses elongated in the prescribed direction. Thumbnail-area borders, either soft or strict (see Sec. 4.7), are obtained by pre-computing once the values of $K_0 - d/K_1$ for each pixel external to the thumbnail-area and reinitializing the depth buffer to these values at each iteration (depth buffer values inside the thumbnail-area are initialized to max depth).

Lloyd relaxation is computed from Voronoi diagrams which are created with off-screen renderings. We compute five relaxation steps before each final rendering shown on screen, when RGB textures representing thumbnail images are accessed (pre-filtered with MIP-mapping) and blended with border color if necessary (see Sec. 5.1).

The off-screen renderings are performed on a FrameBuffer Object which, for sake of speed, is a sub-sampled version of the final screen area. We experimentally determined that a 4×4 sub-sampling introduces acceptable approximation errors (remember that the off-screen renderings are only used to compute Lloyd relaxation, while the Voronoi diagram actually shown is rendered on full resolution).

As a speedup, paraboloids can be rasterized as a single quad which covers its base. For each generated fragment depth is computed according to the squared distance from its site and the region weight. However, we found out that tessellating paraboloids with triangles and quads emanating from the apex actually leads to a slight performance improvement, probably due to the loss of the Hierarchical Z-Buffer optimization implied by any depth-displacing fragment program.

Lloyd relaxation and weight adjustment procedures (Sec. 4 and Sec. 4.1) respectively require to compute current barycenters and areas of each region. In principle, the two tasks can be performed together entirely on the GPU, avoiding read-back of large buffers from video card memory. A Vertex Buffer Object stores a set of 2D vertexes, one for each texel of the FBO. A second FBO texture is then used to count and cumulate the vertexes positions according to the queried region ID in the former FBO. Finally, the latter FBO is read back to CPU where new site positions are computed. However, this approach requires to perform rendering over a 32-bit precision floating point texture, which, on the platforms we used, is not HW-supported and causes huge performance downgrades. Therefore, Lloyd relaxation was implemented in CPU, reading back the off-screen buffer.

Even so, a thumbnail area covering a 1000×300 pixels buffer performs always above 20 frames per second (and thus at 100 relaxation steps per second), resulting in fluid and fast thumbnail movements (on a PC with a 2,6GHz dual core processor, 3GB of RAM and GeForce 130M graphic card). We predict that, avoiding the read-back as discussed above, the same or better results will be easily achieved by leaving more computational resources untapped.

6.2 Out-of-core thumbnail storage

As described in Sec. 3, the proposed approach uses a representativeness-prioritized access pattern (the most representative images are used more often). This is useful when managing huge datasets, like those surpassing a million images: thumbnails cannot be all stored in video memory at the same time, or even in main volatile memory. The adopted access pattern fits well with any hierarchical out-of-core representation of the image dataset and allows for rather straightforward implementations of out-of-core mechanisms that prefetch images and thumbnails over a cascading cache levels system.

7 Results and conclusions

Examples of results obtained by the technique are visible in all images throughout the paper (except for 5), which are all actual screen-shots of the implementation. However, still images are clearly only partially representative of the results. Animated examples can be seen on the attached videos (which are captured on real time).

Note that Lloyd relaxation for getting centroidal Voronoi regions is often used in literature for the good properties of the final configurations it eventually converges to. In our case, we are interested in the process that leads to convergence as much as in the final configurations. Incidentally, that procedure naturally exhibits a convergence speed that corresponds to an appealing *ease out* animation curve [19]: thumbnails start fast and slow down as they come to the end of their motion, increasing the feeling of responsiveness and stability of the browsing mechanism.

7.1 Customizability

The proposed mechanism can be embedded into GUIs in a variety of contexts, thanks to its scalability and adaptivity (e.g. with non-uniform aspect ratios). Also, it is highly customizable, in terms of:

- the shape of the thumbnail area, which is arbitrary as long as it is parameterized (also defining a spatial ordering – see Sec. 4.4);
- the choice of function g determining which and how many images have to be active (see Sec. 3.1);
- the relative size of the thumbnails and how it should vary with the distance from the current focus (see Sec. 3.2);
- whether thumbnails are allowed to "bulge out" a little from the thumbnail area, and if so by how much (see Sec. 4.7);
- the mechanism to fill regions with thumbnails, balancing cropping and downsizing (see Sec. 5.1).

Clearly, the right setting for these parameters strongly depends on the context. Items in the above list represent as many tools in the hand of an interface designer. If needed, they can also be exposed to the end user (e.g. in "settings" panels), as they all allow for real-time tuning: changing them triggers smooth animated transitions between the resulting configurations.

Even though we are not presenting a specific interface for browsing large image datasets, we have tested our mechanism customizability by implementing them in a test environment, in which different practical cases can be reproduced.

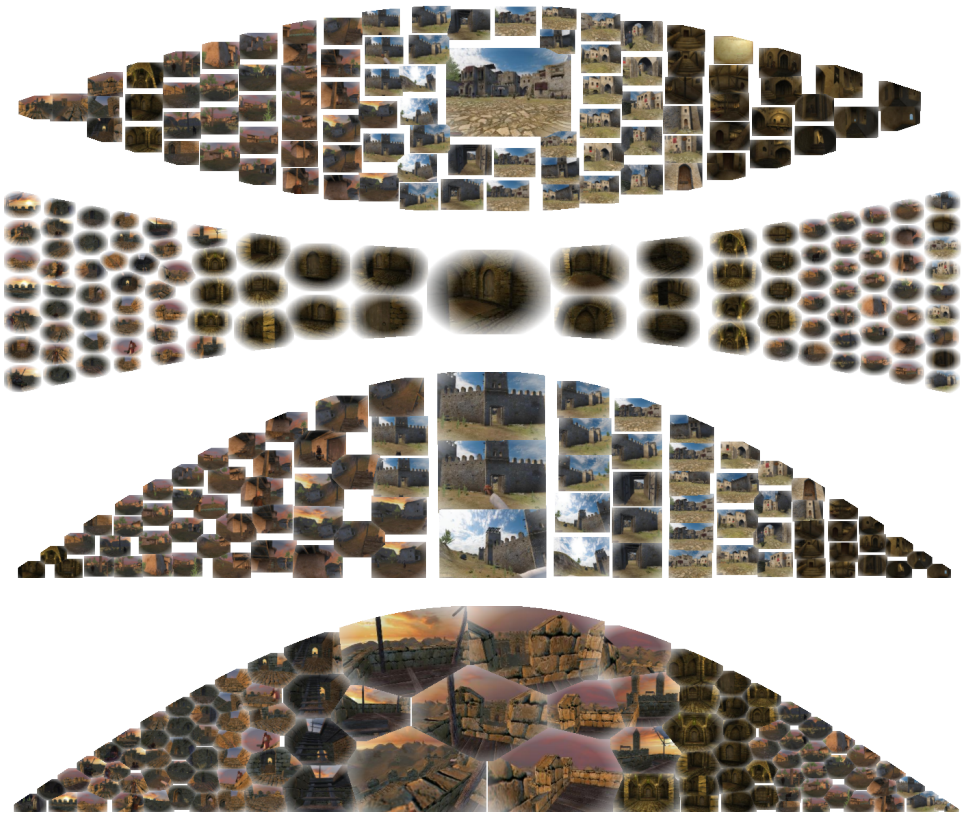

Fig. 7. Examples of results (on the same dataset) featuring various shapes of thumbnail areas and other choices of parameters listed in Sec. 7.1.

7.2 Usability

Here we briefly summarize the results of a preliminary, informal usability study that was conduced on the tool (for space and time reasons, a full scale usability study can only be demanded to future work). A group of 5 graduate students, not familiar of this tool nor CG experts, was asked to browse a medium-large dataset of images (a folder with 728 images featuring a virtual tour of a medieval city, see Fig. 7 and Fig. 8), using alternatively the proposed tool and the publicly available image browser "Picasa 3" by Google.At each run, the testee was asked to seek four particular images, after being given a written textual description of their expected content and those of their neighboring images. We recorded the time used to perform the task.

Picasa provides two alternative interfaces. In the first one, most of the screen is covered by the current focus image, displayed in full resolution; the thumbnail bar is fixed and very tiny (around 2 K-pixels) showing 6 extra-thumbnails around the current focus, arranged on a line. In the second interface, a regular grid of equally sized thumbnails covers most of the screen and no image is shown at full resolution. Even if the first modality represents the targeted scenario of our method, a test against it would have been unfair due to the disparity of screen space devoted to the bar. Instead, we tested against the second modality, conceding 0.6 Megapixels (the minimal size) to Picasa thumbnail bar against only 0.3 Megapixels of ours. Notwithstanding that, timings with the proposed tool were, on average, only 0.67 times the timings taken those of Picasa. Testees reported finding the proposed tool intuitive. Most appreciated its customizability and reported finding it pleasing to navigate with (but see Sec. 7.4).

These results cannot be regarded as conclusive evidence, but they seem to indicate that the proposed tool is an advancement over standard grid based browsers, at least in these setups.

7.3 Conclusions

We presented a novel dynamic image browsing mechanism based on Voronoi diagrams and Lloyd relaxation, adapted for our purposes in a number of different ways.

The system delivers good thumbnail packings (ensuring good screen-space usage) inside a designated area of freely customizable shape. The visualization system scales well with the dataset size: a subset of the dataset currently under direct inspection is displayed in its entirety, while a more succinct representation of the rest of the dataset is offered by showing a progressively sparser subsequence of the images: only the most representative thumbnails are displayed for those images far from the current focus. This is the only approach we are aware of where thumbnails size smoothly varies with the distance from the currently selected image. The system works well with datasets presenting non-uniform aspect ratios, or with images individually rotated, including non-multiples of 90 degrees. During the browsing process, thumbnails are dynamically rearranged, and the above characteristics are preserved during and after the resulting animation.

7.4 Limitations

The proposed system also presents drawbacks. The system is tailored on the single image selection browsing paradigm. This is not necessarily an intrinsic limit, but strategies to accommodate multiple selection of images are still to be investigated. Also, the approach does not provide a good spot for textual labels. While long labels can be substituted by pop-out widgets, there can be applications that require to present them constantly. Another problem relates to visual memory, as moving back to a previously selected focus usually does not result in the same configuration. Lastly, as it often happens, subjective judgement on overall pleasingness is not unanimous, with the occasional user reporting to find the produced dynamic arrangements "untidy looking".

ACKNOWLEDGMENTS

The research leading to these results has received funding from the Tuscany Region (POR CREO FESR 2007-2013) in the framework of the project "VISITO Tuscany", 2009-2011.

Project page: `http://vcg.isti.cnr.it/voronoiBrowser/`

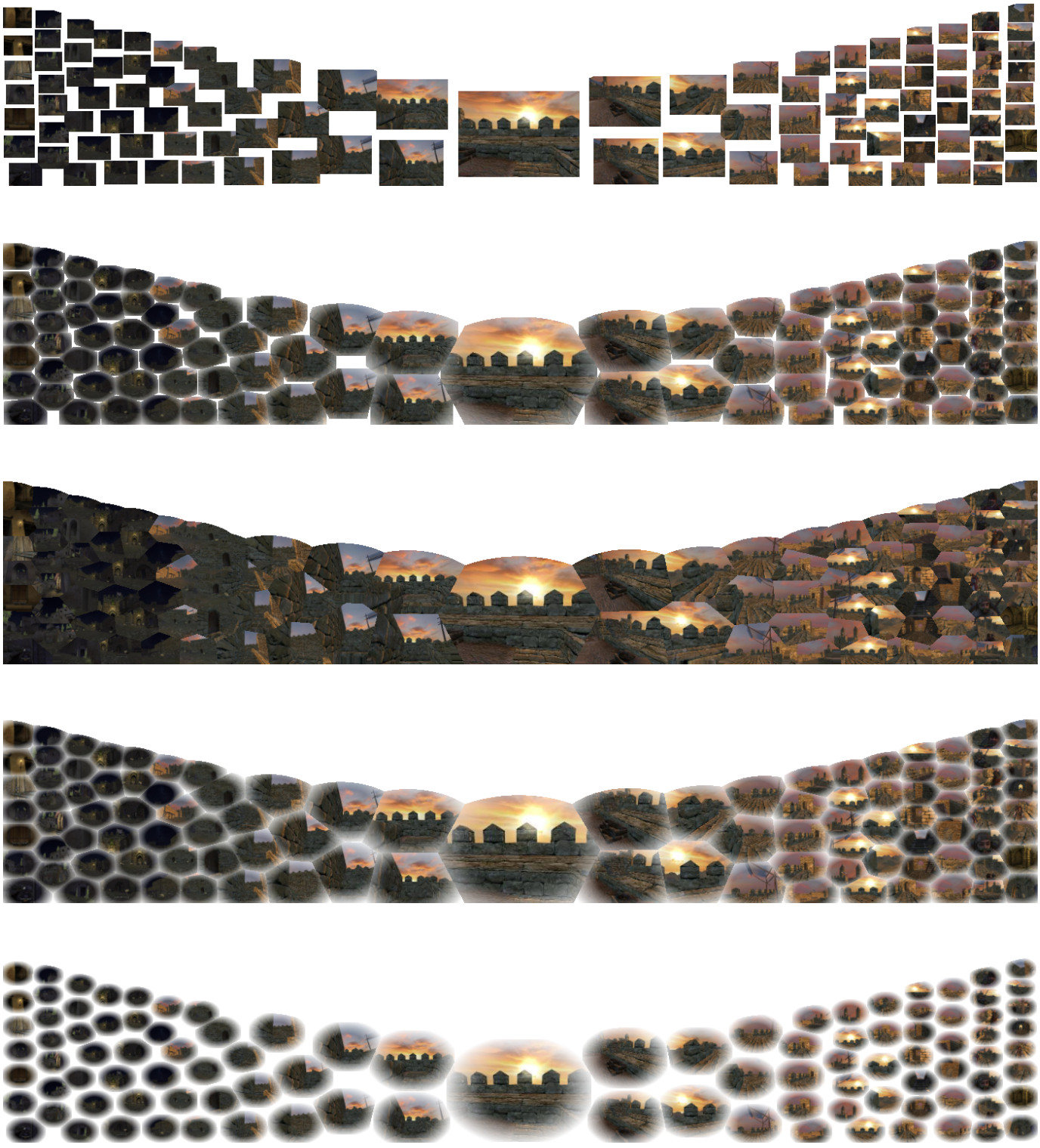

Fig. 8. Four examples of different sets of parameters used to fill the regions with thumbnails. The same Voronoi diagram is used in all examples, but different alternative strategies are used to fill its regions. From top to bottom: one: using a small value for parameter S, images are shrunk more but are well separated. Two: with larger values of S, gaps between thumbnails are reduced, and each thumbnail gets cropped at angles. Three: with larger values yet, thumbnails are cropped more, and end up fully partitioning the plane. Details inside them gets bigger and more visible, but it becomes difficult to pull them away. Four: a white smooth border is added inside cells to visually separate them. Five: the white border is enlarged inside each region, resulting in ellipsoidal shaped regions.

REFERENCES

[1] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991.

[2] B. B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 71–80, New York, NY, USA, 2001. ACM.

[3] G. Bieber, C. Tominski, and B. Urban. Tidi browser: a novel photo browsing technique for mobile devices. volume 6507, page 65070O. SPIE, 2007.

[4] P. Brivio and M. Tarini. Picture-driven procedural modelling - building an animated model of ghirla watermill (18th cen.). In *Eurographics '09 Italian Chapter Conference*, October 2009.

[5] S.-J. Cho, R. Murray-Smith, and Y.-B. Kim. Multi-context photo browsing on mobile devices based on tilt dynamics. In *MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, pages 190–197, New York, NY, USA, 2007. ACM.

[6] S. M. Drucker, C. Wong, A. Roseway, S. Glenner, and S. De Mar. Mediabrowser: reclaiming the shoebox. In *AVI '04: Proceedings of the working conference on Advanced Visual Interfaces*, pages 433–436, New York, NY, USA, 2004. ACM.

[7] B. Epshtein, E. Ofek, Y. Wexler, and P. Zhang. Hierarchical photo organization using geo-relevance. In *GIS '07: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, pages 1–7, New York, NY, USA, 2007. ACM.

[8] D. Frohlich, A. Kuchinsky, C. Pering, A. Don, and S. Ariss. Requirements for photoware. In *CSCW '02: Proceedings of the 2002 ACM conference on Computer Supported Cooperative Work*, pages 166–175, New York, NY, USA, 2002. ACM.

[9] Google. Picasa. `http://picasa.google.com/`, 2004.

[10] Google. Streetview. `http://www.google.com/streetview/`, 2007.

[11] O. Hilliges, D. Baur, and A. Butz. Photohelix: Browsing, sorting and sharing digital photo collections. *TABLETOP 2007: Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, 0:87–94, 2007.

[12] K. E. Hoff, III, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 375–376, New York, NY, USA, 2000. ACM.

[13] C.-C. Hsieh, W.-H. Cheng, C.-H. Chang, Y.-Y. Chuang, and J.-L. Wu. Photo navigator. In *MULTIMEDIA '08: Proceeding of the 16th ACM international conference on Multimedia*, pages 419–428, New York, NY, USA, 2008. ACM.

[14] D. F. Huynh, S. M. Drucker, P. Baudisch, and C. Wong. Time quilt: scaling up zoomable photo browsers for large, unstructured photo collections. In *CHI '05: Conference on Human Factors in Computing Systems '05 extended abstracts on Human factors in computing systems*, pages 1937–1940, New York, NY, USA, 2005. ACM.

[15] S. Kandel, A. Paepcke, M. Theobald, H. Garcia-Molina, and E. Abelson. Photospread: a spreadsheet for managing photos. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1749–1758, New York, NY, USA, 2008. ACM.

[16] A. Khella and B. B. Bederson. Pocket photomesa: a zoomable image browser for pdas. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 19–24, New York, NY, USA, 2004. ACM.

[17] J.-W. L. R. H. Kim, S.-K. Lee and M. E.-H. Chung. User-friendly personal photo browsing for mobile devices. *ETRI JOURNAL*, 30(3):432–440, 2008.

[18] F. Labelle and J. R. Shewchuk. Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 191–200, New York, NY, USA, 2003. ACM.

[19] J. Lasseter. Principles of traditional animation applied to 3D computer animation. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 35–44. ACM, 1987.

[20] Microsoft. Photosynth. `http://photosynth.net`, 2007.

[21] T. J. Mills, D. Pye, D. Sinclair, and K. R. Wood. Shoebox: A digital photo management system. Technical report, AT&T Research, 2000.

[22] J. C. Platt, M. Czerwinski, and B. A. Field. Phototoc: Automatic clustering for browsing personal photographs. Technical Report MSR-TR-2002-17, Microsoft Research, 2002.

[23] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen. Image browsing, processing, and clustering for participatory sensing: lessons from a dietsense prototype. In *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*, pages 13–17, New York, NY, USA, 2007. ACM.

[24] K. Rodden and K. R. Wood. How do people manage their digital photographs? In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 409–416, New York, NY, USA, 2003. ACM.

[25] L. Shi, J. Wang, L. Xu, H. Lu, and C. Xu. Context saliency based image summarization. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 270 –273, june 2009.

[26] B. Shneiderman and H. Kang. Direct annotation: A drag-and-drop strategy for labeling photos. *Information Visualisation, International Conference on*, 0:88, 2000.

[27] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. In *SIGGRAPH '08: ACM SIGGRAPH 2008 Papers*, pages 1–11, New York, NY, USA, 2008. ACM.

[28] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 835–846, New York, NY, USA, 2006. ACM.

[29] K. Thys, R. Thys, K. Luyten, and K. Coninx. Photofoaf: A community building service driven by socially-aware mobile imaging. In *Proceedings of International Workshop on Semantic Media Adaptation and Personalization*, 2006.

[30] K. Toyama, R. Logan, and A. Roseway. Geographic location tags on digital images. In *MULTIMEDIA '03: Proceedings of the 11th ACM international conference on Multimedia*, pages 156–166, New York, NY, USA, 2003. ACM.

[31] M. Vergauwen and L. Van Gool. Web-based 3d reconstruction service. *Machine Vision and Applications*, 17(6):411–426, 2006.

[32] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, New York, NY, USA, 2004. ACM.

[33] F. Wu and M. Tory. Photoscope: visualizing spatiotemporal coverage of photos for construction management. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1103–1112, New York, NY, USA, 2009. ACM.