The 13rd International Symposium on Virtual Reality, Archaeology and Cultural Heritage VAST
(2012)
D. Arnold, J. Kaminski, F. Niccolucci and A. Stork (Editors)

PileBars: Scalable Dynamic Thumbnail Bars

Paolo Brivio[1], Marco Tarini[1,2], Federico Ponchio[2], Paolo Cignoni[2], Roberto Scopigno[2]

[1]Università degli Studi dell'Insubria, Varese, Italy
[2] Istituto di Scienza e Tecnologie dell'Informazione (ISTI), CNR, Pisa, Italy

Abstract

We introduce PileBars, a new class of animated thumbnail-bars supporting browsing of large image datasets (hundreds or thousands of images). Since the bar is meant to be just one element of a GUI, it covers only a small portion of the screen; yet it provides a global view of the entire dataset, without any scrolling panel. Instead, thumbnails are dynamically rearranged, resized and reclustered into adaptive layouts during the entire browsing process. The objective is to enable the user both to accurately pinpoint a specific image (even among semantically close ones), and to jump anywhere to "distant" parts of the dataset. The thumbnail layouts proposed maximize also the temporal coherence, thus allowing for smooth animations from one layout to the next. The system is very general: it can be driven by any application-specific image-to-image semantic distance function, and respects any user-defined total ordering of the images; the ordering can be either inferred from the semantic or be chosen independently from it, depending on the application. The applicability of the resulting system is tested in a number of practical applications and fits very well the issues in management of Cultural Heritage image collections.

1. Introduction

In the aftermath of the explosion of digital imagery, large image datasets are becoming ubiquitous. The management of groups of semantically-related images (hundreds up to tens of thousands of digital images) is a common need of several application domains. An immediate example are the extensive photographic acquisition campaigns in Cultural Heritage (CH) applications (documentation of architectures or of archaeology excavations or sites, conservation projects, etc.), but also others such as thematic picture collections harvested from the web with image-based web-searches or personal picture collections. Invariably, CH users demand practical and intuitive ways to profitably explore and browse these images, possibly integrated in virtual 3D environments [BBT*12], both for scientific purposes and for presenting their results in museums.

Dealing with image datasets of that size is not trivial. Even plain browsing becomes a difficult task, e.g. how to let a user locate one image, or how to visualize the entire dataset as a whole, or how to navigate throughout the dataset. In this context, thumbnail-bars are a very common browsing GUI mechanism. In a thumbnail-bar each image is represented by a small thumbnail (a downsized and/or cropped version of the original image). Usually the thumbnail-bar is but one element of a more complex interface, and covers only a subset of the screen space.
Conventional thumbnail-bars display thumbnails over a panel, laid in a line or in a regular grid. When the panel size exceeds the screen size devoted to the thumbnail-bar, the panel is made scrollable and scrolling mechanisms are provided to the user to navigate the dataset. As the number of images increase over a few hundreds, this approach quickly becomes inadequate: either thumbnails are excessively downsized to be meaningful, or the amount of scrolling required becomes unfeasible.

We propose a new class of thumbnail-bars, termed *'PileBars'* (see Fig. 1), that are *focus-plus-context* oriented in order to achieve a better scalability. Images come often enriched with meta-data, ranging from tagging (e.g. from "semantic web"), to time of shot/creation, to per-image extrinsic camera calibration (position/orientation of the camera), and others yet can be extracted with various kinds of content analysis. We take advantage of that meta-data by using it either as ordering functions or as semantic distance functions defined over the dataset. The main idea is to reducing the cluttering on the screen by a dynamic and automatic grouping mechanisms. The challenge lies primarily in the conception of effective, new info-visualization and in-

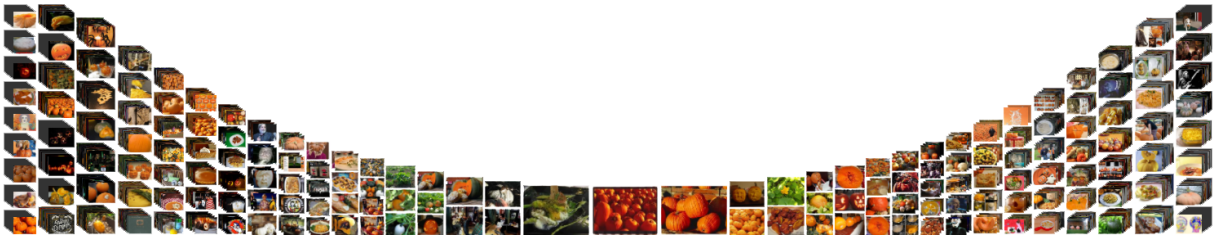

Figure 1: *A dataset of a thousand images visualized globally in a PileBar. Near the center, each individual image is visible as a larger thumbnail. At the side, images far from the area of interest are piled together into increasingly taller stacks. As the area of interest changes, the thumbnail bar dynamically rearranges accordingly with a smooth animation.*

teraction mechanisms, but it also embeds a system design constraint: images in the dataset can be just too many to host their thumbnails simultaneously in RAM (or in texture RAM); also in cases where the dataset is hosted on a remote server, e.g. in web applications, it would be impractical to download the entire thumbnail dataset before browsing it.

2. Related work

Image browsers are very common. Every modern Operating System embeds at least one, and with the ubiquitous diffusion of small digital cameras we can find browsers even in mobile phones. A conventional solution that image browsers adopt is to present images on a linear thumbnail-bar [Goo04] or arranged in a simple grid layout. As discussed, in many CH scenarios these simple approaches do not scale up well enough with the number of images.

Zoomable interfaces (i.e. dynamically reducing and enlarging thumbnails size) can aid the browsing to some extent [CB99], but images cannot be reduced indefinitely, and some sort of semantic zooming [HDBW05] or grouping [Bed01] must be introduced. Also, image arrangement has significant impact on the browsing, and different users might prefer different arrangements for the same dataset [RBSW01]. Users generally prefer to see many thumbnails in layouts which reflect some meaningful order (e.g. the time of shot), while browsing is usually performed interactively rather than by queries [RW03].

A trend consists in clustering images hierarchically, according to some kind of image semantic, like combining tags [KS05], time [GGMPW02], time and space [RCC10], color [PCF02], spatial image-distances [EOWZ07, JYC09], or a mixture of them [MFGJ08] to automatically, or interactively [GSW*09, CdOKRV09] compute image-clusters. Most of these works strive to identify good clusterings for images, rather than good way to dynamically present and explore the clustered dataset. Our approach is orthogonal to these, and we can adopt many of these clustering techniques. Our approach is similarly motivated to a [BTC10], which is based on an extended Voronoi relaxation scheme. [BTC10] supports arbitrary shaped thumbnail-bars, with irregularly shaped thumbnails whose sizes and locations are dynamically rearranged in liquid, optimized compositions. Thanks to the irregular thumbnail shapes, that approach copes better with the differences in aspect ratios of input images. In exchange, PileBars is superior under the following aspects:

- it is lighter on computational resources (and also easier to implement);
- thumbnail layouts are more regular, are thus more intuitive to scan [RBSW01]; they are also more stable during the animations;
- the approximative number of images clustered together is made visible in the interface;
- experiments indicates a better overall scalability with the number of images

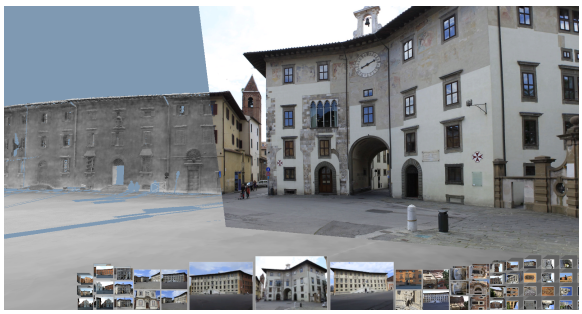

Figure 2: *A screenshot of PhotoCloud, an application using PileBars and presenting here the Cavalieri Square dataset. The main area of the screen presents a 3D scene, that the user is able to navigate. Selecting a focus image on the PileBar presented below causes the viewpoint to jump on the corresponding virtual camera location.*

3. PileBars overview

In thumbnail-bars (including PileBars), at any given time, one image of the dataset is the current *'focus'*, selected by the user. The focus image can be regarded as the current

"position" where the user sits within the dataset. The exact meaning of selecting a focus depends on the application: in an image viewer, the focus can be the image being shown at full resolution in the main area of the screen; in a 3D navigation application [Mic07, BBT*12], where images are linked to viewpoints (the physical camera positions of the shot), the focus image can drive the setup of the virtual camera used to render the 3D scene inside the main screen area (see Fig. 2).

In a PileBar, thumbnails dynamically change size and position during the browsing, and can be laid on top of each other forming *'piles'* of varying heights. Piling is a natural visual way of clustering similar images together. During the browsing, piles dynamically split into smaller piles or are merged into larger piles. In a pile, only the top thumbnail is fully visible to the user. Therefore, it must be a good representative of the images behind it.

The current focus determines the layout of the thumbnail-bar, i.e. the position, size and piling of every thumbnail in the dataset (see Sec. 4.2 and Fig. 4). A browsing session consists in a sequence of selections of new focuses, and consequently the animations from each thumbnail layout to the next.

In line with *focus-plus-context* principle, images semantically closer to the current focus are displayed as larger thumbnails: this constitutes the "focus" part of the interface, where the user must be able to see, and tell apart, images close to its current "position" in the dataset. Conversely, images farther from the focus will be represented by thumbnails which are smaller and stacked in taller piles: this is the "context" part of the interface, where the user is given a global vision of the entire dataset.

There is a continuum, both spatial and temporal, going from the "focus" to the "context" part. Spatially, the change of thumbnail sizes and piling is progressive, from the central focus thumbnail (large, isolated thumbnails) to the peripheral parts of the bar (small, piled thumbnails). Temporally, the transitions from a thumbnail disposition to the next (triggered by any change of the current "focus") is made smooth: we strive to keep the changes in size, position and piling of thumbnails as small as possible; changes are further masked by means of smooth animations (Sec. 4.3).

4. PileBars

Thumbnail layouts in a PileBar change dynamically by following some assumptions and objectives:

Total ordering. At any given moment, a specific total ordering is defined over the images; i.e. images in the datasets are assumed to be (implicitly) numbered from the first I_0 to the last I_{N-1}. This is a central choice of our approach, as navigating a linearized dataset is intrinsically easier and more intuitive; this is especially meaningful for thumbnail-bars whose shapes have a single dominant dimension. Alternative orderings can be selected dynamically by the user. The total ordering helps the user orient in the dataset: it is always reflected in the left-to-right spatial disposition of the thumbnails.

Semantic distance function. PileBars *can* take advantage of any meaningful image-to-image distance function $\mathcal{F}(I_a, I_b)$ available in the context of the dataset, and use this information as a way to drive thumbnail piling. Images which are mutually closer (according to this function) are piled together more often than images that differ much from each other. This way, the image visible on top of the pile will be a good representative of the pile. Images that are detected as different can still be piled together, but only when they are sufficiently farther away from the current focus.

The definition of image-to-image distance function(s) and total ordering(s) are two ways to taylor the behavior of a PileBar around the needs of a specific application scenario. Several possible choices (both general and application-specific) are summarized in Sec. 6.1 and Sec. 6 respectively.

4.1. Defining a PileBar shape

A standard thumbnail-bar is customizable by few spatial parameters, e.g. overall size, thumbnail size, and spacing between thumbnails. A PileBar has more degrees of freedom and it is defined mainly by a configuration of *'slots'*, which partition the 2D region of the screen it covers (see Fig. 3). A slot is a fixed rectangular region that will be occupied by a single pile of thumbnails.

Each slot S_i comes with a set of predefined fixed attributes, which stay constant during the browsing (barring resizing events, see Sec. 4.4). There are: a 2D position p_i, a size d_i, and a *'stacking value'* k_i affecting the height of the piles that will occupy it: slot S_i is expected to host piles of around 2^{k_i} thumbnails. The special value $k_i = -1$ means that slot S_i will host piles consisting of *strictly* one thumbnail.

There is a total of $2n+1$ slots, indexed from S_{-n} to S_n. Slot S_0, termed the *'central'* slot, is designated to contain the focus thumbnail. That slot is maximal in size, central in position, and carries a pile of only one thumbnail (i.e. $k_0 = -1$). Slots S_i, $i \in \{-n, ..., -1\}$ are placed on the left of S_0, and slots S_i, $i \in \{1, ..., n\}$ are placed symmetrically on its right. Slots become progressively smaller in size and have bigger values of k_i as $|i|$ increases. On either sides, sets of consecutive slots are grouped into a number of side-to-side *'columns'*; each is composed of a number (usually 1 to 5) of vertically aligned slots of the same size.

Different arrangements of slots following the above schema, like the ones depicted in Fig. 3 and Fig. 4, can be easily constructed procedurally, by selecting a few parameters which determine the number of slots per column and the fixed attributes of each slots S_i, as a function of $|i|$.

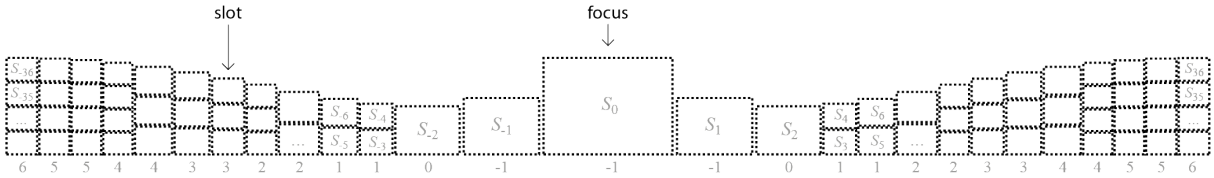

Figure 3: *An example of the arrangement of slots for a PileBar. Below each column of slots we report the value of k for all the slots in that column. Each slot will host a pile of, on average, 2^k thumbnails (or exactly one when $k = -1$).*

4.2. Arranging and piling thumbnails into slots

In a preprocessing stage, each image I_i in the dataset, $i \in \{0..N-1\}$, is assigned to a integer value σ_i, which represents its 1D position into a 'linear semantic space'. The values are strictly monotonically increasing and are such that, for two consecutive images in the dataset (in the total ordering), the difference of their σ is proportional to their image-to-image distance (up to the rounding). The following pseudo-code computes values of σ_i ($\mathcal{F}$ is the image distance function, and K is a parameter):

```
σ_0 ← 0
foreach i in {0..N−2}:
  σ_{i+1} ← σ_i + ⌈K · F(I_i, I_{i+1})⌉
```

At runtime, each thumbnail T_i of image I_i is assigned to a slot S_j. The linear semantic space is subdivided into chunks of increasing size, and thumbnails of consecutive images with σ values falling into the same chunk are piled into the same slot. Images are processed from the focus image I_f, whose thumbnail T_f is assigned to the central slot S_0, to the last image. Every time an image is found to belong to a different chunk of semantic space than the previous image, the next slot gets used. The size of the chunk depends on the clustering factor k_j of the currently slot S_j. Consecutive images falling in the same cluster are placed in the same chunk:

```
procedure Arrange:
j ← 0
c_last ← Cluster(σ_f, k_j)
foreach i in {f..N−1}:
    c_now ← Cluster(σ_i, k_j)
    if c_now ≠ c_last then j ← j+1
    Assign T_i to S_j
    c_last ← c_now
```

where Cluster is a function returning an unique index of a chunk of semantic space, given a position in linear semantic space σ and the clustering factor k, as described below. The process is repeated backwards to assign all thumbnails from T_f to T_0 to left-side slots.

```
function Cluster(σ,k):
    if k=−1 return σ
    else return σ >> (norm+k)
```

Figure 4: *Another example of layout of slots (as in Fig 3), this time for a PileBar with a rectangular shape.*

where $>>$ is the right shift operation in base 2. In the $k = -1$ case, a cluster will span exactly one value of the linear semantic space, as intended. The integer value *norm* is a normalization factor: $norm = \log_2(\sigma_{N-1}/N)$, rounded to the closest integer. That value is selected so that when $k = 0$ there is approximately one cluster of semantic space per image in the dataset: for any $k \neq 0$, the semantic space is clustered into approximatively $N/2^k$ chunks, resulting on an average of around 2^k images per slot. Note that a slot with $k = 0$ usually contains one image, but can occasionally contain multiple semantically close ones.

The above functions are designed to ensure that, if image T_i is on top of a stack in a given layout, it will continue to do so if the focus is moved closer. This is necessary to ensure good continuous transitions between layouts.

Sort over *y* axis: in contrast to the ordering of thumbnails on the *x* screen axis, which reflects the total ordering of the images, the distribution of thumbnails on the *y* screen axis is not associated to any semantic and is exploited to maximize temporal coherence. To achieve this, we keep track of the *y* position that each thumbnail had in the last frame. Within each column, thumbnail piles are re-assigned to the slot in the same order (from bottommost to topmost) of the *y* values of last frame. This simple strategy ensures that, when a new focus is selected, piles (whether they are dividing, merging, or just moving) are rearranged in a way that minimizes the total magnitude of movements (see the attached videos).

Determining size of thumbnails: in a slot ending up with just one thumbnail, that thumbnail covers the entire slot. When a pile of $n_p > 1$ thumbnails is assigned to a slot, a little of the space is devoted to make the stacking visible to the user: thumbnails are slightly reduced in size, by a factor r which increases with n_p ($r = 1 - r_0 \log_2(n_p)$, with $r_0 = 0.3$). Topmost thumbnail is pushed to the bottom left border of the

Figure 5: *Example of thumbnails piled inside a slot (shown dashed). From left to right: a pile of 1, 2, 3, 4, 12, 100 thumbnails.*

slot (bottom right, on the left-sided slots), leaving a margin where the other thumbnails are equally spaced (see Fig. 5).

4.3. Transitions

Change in thumbnails disposition induced by choice of a new focus is performed through an animation which smoothly interpolates between the old and the new layout. This is trivial to do, because the layout consists in a *status* per each thumbnail, and *statuses* can be linearly interpolated by linearly interpolating all their attributes.

4.4. Liquid PileBar shapes

Even if the layouts and the attributes of the slots are kept fixed during navigation, they can be easily changed dynamically to adapt to changed conditions. The values of slot attributes k can be made adaptive: if empty slot columns are found on either side, the values of k of a few slots are automatically decreased so that the piles will partially unfold and spread into more slots, covering the empty space. Conversely, whenever slots are not enough to hold the piles, the values of k are increased to produce taller and fewer piles.

In an interface, the user can be allowed to reshape the PileBar. Changing its aspect ratio has a potentially useful and intuitive side effect: when the PileBar is reduced only vertically, and columns are kept the same number of slots, then the slots size will reduce so that they can fit vertically in the new shape. This will make new columns of slots appear at the sides, which, being empty, trigger lowering of k values. The net effect is that, making the PileBar thinner will reduce the thumbnails, but will result in less piling, thus revealing more images. Conversely, widening the bar produces larger thumbnails, but more extreme clustering (see attached Video 5).

5. Interacting with the PileBar

PileBar comes with an array of natural interaction mechanisms. A class of mechanisms requires the user to pick a thumbnail with the pointer (e.g. mouse). Note that, thanks to the way thumbnails are arranged into piles (see Fig. 5), not only the top thumbnail of each pile can be picked, but also to the ones behind it, by pointing at the small border which is left visible.

Focus change. Modifications of the current focus can be triggered by events like:

- *pointing and clicking* on a thumbnail anywhere in the PileBar: the selected thumbnail becomes the new focus;
- *dragging* a thumbnail with the pointer over the x axis into a new position: the focus changes, so that the user selected x position is matched by the layout induced by the new focus. This is the user action corresponding to a scroll action in a conventional thumbnail-bar.
- *next* and *prev* events (e.g. linked to buttons, keys, or mouse wheel, etc.): the focus changes by one or several images in either direction.

The dragging action is very general: bringing a thumbnail in place of the current focus will make it the new focus; dragging a thumbnail away in the peripheral region of the PileBar will cause the focus to change drastically, because of the exponential piling factor. Conversely, the *next* and *prev* keys support fine tuning of the focus. Additional application-specific mechanisms (e.g. returning to a previously bookmarked thumbnail, image searches, etc.) can be added to change the current focus.

Whatever is the reason that triggered the change of focus, the effect is that, due exponential effect of the piling nature, the small piles close to the focus will move very fast, while the taller piles at the borders will move at a much lower speed; all piles will progressively split into smaller piles as they get closer to the focus, or merge into taller piles as they get farther from the focus. Importantly, small changes of focus will cause correspondingly small layout changes.

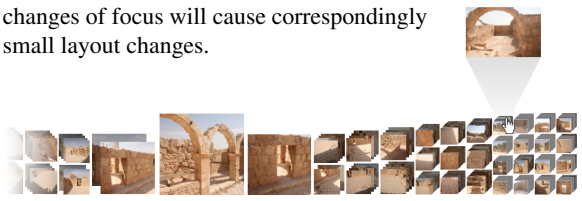

Figure 6: *A screenshot of the effect of "peeking" at a thumbnail inside a pile.*

Peeking. The user can *peek* at thumbnails, even if they are far away from the focus, by positioning the pointer over them, while (for example) pressing the right mouse button. The peeked thumbnail image will be shown above the PileBar fully unoccluded and at maximal thumbnail size, regardless of its position and size (see Fig. 6 and Video 3). This can be useful to enlarge small thumbnails at the PileBar periphery, and to reveal thumbnails partially hidden behind others inside piles.

Vertical drags. The effect of dragging a thumbnail solely along the *y* screen direction is simply to change the position of the moved pile from a slot to another in the same column.

Reordering. When more than one total ordering among images is available and meaningful (Sec. 6.1), the user can be allowed to switch dynamically from an ordering to another, e.g. via a button, similarly to what happens in a traditional image or file browser. The new ordering will automatically induce a new disposition of thumbnails inside the bar (position, sizing, piling of thumbnails); temporal coherence is ameliorated, as normal, by animations (see Sec. 4.3). The net effect is that the focus thumbnail remains fixed, and the other thumbnails redispose around it.

Resizing. As mentioned in Sec. 4.4, the PileBar is *liquid* in the sense that it is capable of gracefully adapting to different sizes and aspect ratios, with useful side effects.

6. Image Ordering and Image Distance Functions

As mentioned, an image-to-image distance function is used to cluster thumbnails relative to similar images into piles. Any distance function defined between images can be profitably employed for this purpose, regardless of the range, the scale, and the distribution of the returned values. Computational time is not critical here also because the distance will be pre-computed only for pairs of images that are consecutive in the total ordering, which is feasible even for the largest image datasets.

There are plenty of image distance metrics readily available. These include:

1. time (the distance in seconds between times of shot);
2. shot positions (the distance between shot positions);
3. shot directions (the angle between two shot directions — note that, conversely, the rotation of the camera around the shot direction axis is usually not meaningful);
4. view frustum content similarity;
5. average color (distance between images average colors);
6. color distribution (distance between normalized color histograms);
7. color spatial distribution (averaged per-pixel distance between severely downsized versions of the two images);
8. tagging distance (distance in tag space).

Clearly, not every measure is available in every dataset. Most photographic collections (including most personal image collections) come with *time-of-shot* data (provided by the file system or *exif* metadata). Time distances can range from a few milliseconds to several months, within the same dataset. This is not a problem, thanks to the normalization described in Sec. 4.2. Time distances are often a very natural choice, as it results in images being piled (and dynamically re-piled) per hour, day, week, month, etc. (according to the distance from the focus). *Shot-positions* and *shot-directions* are available when we have per-image camera calibration data, as in datasets resulting from photographic campaigns processed with 3D photo reconstruction techniques [VVG06, Mic07]. The *color-based* metrics (distances 5, 6, 7) are the simplest content-based distance functions, and they are usually good indicators of image semantic similarity. They can be useful, for example, to pile together images from a movie belonging to the same scene.
Regardless of its origin, each distance function results in a scalar positive quantity returned for an image pair. Therefore, different distance functions can be interpolated to form new *mixed* distance functions.

6.1. Examples of image ordering functions

As mentioned, a total ordering is assumed to be defined over the images. Many applicative contexts come with one (or more) "natural" image orderings: e.g. an order for a photographic campaign can be induced by the time of shot; web search results can be ordered by relevance; tagged images can be ordered alphabetically by tags, etc.

Useful orderings can also be inferred from any of the semantic functions described above (including the mixed ones): the image dataset can be considered a weighted fully connected graph, and the (approximatively) minimal Hamiltonian path will define a linear ordering of the images (found by means of heuristics).

7. Implementation issues

Since the focus of PileBars is scalability with dataset size, implementation efficiency is crucial. Real time performance is needed, but system resources must be left untapped for the rest of the application in which the PileBar is embedded in.

Arranging thumbnails according to a new focus (Sec. 4.2) is very efficient and must be performed once per focus change. Animating the thumbnail layout (Sec. 4.3) must be done once per frame, but is an even less demanding operation, boiling down to interpolation of a few scalar parameters per image. Tests show that both tasks present no performance hit even for datasets with 10^5 images. More care must be used with rendering performance, and memory/bandwidth requirements.

7.1. Rendering system

PileBars can take advantage of basic rendering mechanisms hardwired (and heavily optimized) in any GPU card. A thumbnail is simply a textured quad with a low resolution texture (not higher than the slot largest size, in pixels). However, plain rendering of several thousands of textured quads can still downgrade system performance. Technically, the problem is tied to the extreme depth complexity reached in piles holding hundreds of images, which tend to make the application fill limited (even if the per-fragment workload is very small). Both issues are solved by checking, for each

thumbnail T_i not on top of its pile, the relative position of thumbnail T_j directly above it in the current layout. During rasterization, the two positions differ by an integer number p of pixels (the maximal one at the four sides). Whenever p is zero (which happens if T_j is aligned with T_i up to pixel precision), the rendering of T_i is skipped. If p is one, only a row and a column of pixels of T_j is rendered (with line primitives). Else, the thumbnail is fully rendered. In very tall piles, static or moving, most renderings are skipped; a minority will be rendered producing few fragments; only one will be fully rendered.

On average, in a PileBar of 10^5 thumbnails laid out as in Fig. 1, 90.0% of the thumbnails are skipped, 9.8% are rendered as lines, and 0.2% are fully rendered.

7.2. Cache system

Large dataset visualization requires keeping in memory a large number of thumbnails. Even considering that they are much smaller than the real images they represent, the memory requirement easily surpasses texture RAM capacities (e.g. 10^5 images, with thumbnails resolution 256^2, require $\sim$ 20 GBytes of raw pixel data in texture memory; using MIP-maps, this requirement rises to 27 GBytes). Texture compression alone does not solve the problem.

To address this issue, we adopted a strictly inclusive, multilevel, multithreaded cache system, similarly to [vW07], consisting of four layers: (1) remote server (in remote applications); (2) local disk; (3) RAM; (4) texture RAM, on board of the graphic card. Each stage of the cache is executed in its own thread to avoid stalling the system on load/download.

In layers (1) and (2) thumbnails are stored using the compact JPG format. In layers (3) and (4) they are stored in the less compact but GPU friendly DXT1 compression. When passing objects from (2) to (3), fast JPG compression (libjpeg_simd) and fast DXT recompression [RJL07] libraries are used; both performs at around 100 MPixels/sec (meaning that computational resources are left untapped).

The total ordering of the thumbnails is used to assign a priority to each thumbnail equal to the image pixels actually shown on screen. The cache dynamically manages the available memory so to keep the highest priority thumbnails as close as possible to the last stage. Thumbnails not ready in GPU RAM are shown as rectangles colored as their average color, which is precomputed.

8. Application scenario examples

The PileBars approach has been used in the PhotoCloud system (Fig. 2, [BBT*12]) and we extensively tested it on several application scenarios:

1. Cavalieri square, Archeo01, Arche02: calibrated images resulting from CH photographic campaigns;
2. Pumpkin: images resulting from a Google search;
3. Toy: frames from a movie;
4. Animals: tagged images featuring different animals.

Tab. 1 presents for each scenario: the ordering functions used, the metric distance function used, and links to the images in the paper and the attached videos. Fig. 7 shows some thumbnail arrangements produced with PileBars. However, a still image is inadequate to show its effect and the reader is invited to see the attached videos. Performance-wise, the system is able to run at 60 fps on off the shelf computers, leaving RAM, CPU and texture memory untapped.

9. Conclusions

We have presented PileBar, a scalable tool to effectively and efficiently browse very large image datasets with a focus-plus-context approach. Its features include clarity of dataset visualization, ease of use, temporal coherence, customizatibility, richness of interaction mechanisms, ease of implementation, and low performance impact. Thanks to its features, PileBar is a GUI component for image browsing which can be beneficially integrated within a wide variety of software applications. The main limitation is that image aspect ratio is assumed to be constant in the dataset (cropping must be employed when this is not the case, e.g. [AS07]).

A user study, strongly indicating that PileBar outperforms conventional image browsers, is available at the project web page: `http://vcg.isti.cnr.it/pilebars/`

Acknowledgements This work received funding from the EC 7th FP (FP7/2007 2013) under grant agreement No. 231809 (IP *3DCOFORM*).

References

[AS07] AVIDAN S., SHAMIR A.: Seam carving for content-aware image resizing. *ACM Trans. Graph. 26* (2007). 7

[BBT*12] BRIVIO P., BENEDETTI L., TARINI M., PONCHIO F., CIGNONI P., SCOPIGNO R.: PhotoCloud: realtime web-based interactive exploration of large mixed 2D-3D datasets. *IEEE Computer Graphics and Applications 32* (2012), in press. 1, 3, 7

[Bed01] BEDERSON B. B.: 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* (2001), ACM, pp. 71–80. 2

[BTC10] BRIVIO P., TARINI M., CIGNONI P.: Browsing large image datasets through voronoi diagrams. *IEEE Trans. on Visualization and Computer Graphics 16*, 6 (2010), 1261–1270. 2

[CB99] COMBS T. T. A., BEDERSON B. B.: Does zooming improve image browsing? In *Proc. of the 4th ACM conference on Digital libraries* (1999), ACM, pp. 130–137. 2

[CdOKRV09] CRAMPES M., DE OLIVEIRA-KUMAR J., RANWEZ S., VILLERD J.: Visualizing social photos on a hasse diagram for eliciting relations and indexing new photos. *IEEE Trans. on Visualiz. and Comp. Graph. 15* (2009), 985–992. 2

[EOWZ07] EPSHTEIN B., OFEK E., WEXLER Y., ZHANG P.: Hierarchical photo organization using geo-relevance. In *GIS '07: Proc. of the 15th ACM Int. Symp. on Advances in Geographic Information Systems* (2007), ACM, pp. 1–7. 2

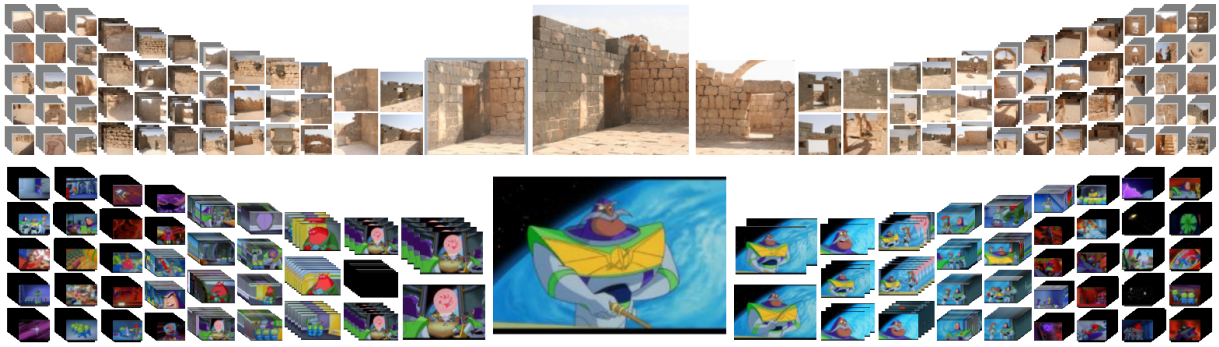

Figure 7: *Screenshots of PileBars in action on different datasets. Also refer to the attached videos.*

DATASET	IMAGE NO.	ORDERING FUNCTION(S)	DISTANCE FUNCTION(S)	FIG.	VIDEO
Cavalieri Square	461	shot position + orientation	time of shot	2	
Archeo01	430	time of shot, custom metric	time of shot + constant shot position	6	1,2
Archeo02	333	time of shot, shot position	constant, images spatial colour layout	7 (top)	3,5
Pumpkin	992	relevance, images spatial colour layout	images spatial colour layout, constant	1	4
Toy	10049	sequence of frames, images color, images spatial colour layout	images color, images spatial colour layout	7 (bottom)	6
Animals	3784	images metatag + images color + images spatial colour layout	images color + images spatial colour layout		7

Table 1: *Images used and relative data*

[GGMPW02] GRAHAM A., GARCIA-MOLINA H., PAEPCKE A., WINOGRAD T.: Time as essence for photo browsing through personal digital libraries. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries* (New York, NY, USA, 2002), ACM, pp. 326–335. 2

[Goo04] GOOGLE: Picasa. `http://picasa.google.com/`, 2004. 2

[GSW*09] GIRGENSOHN A., SHIPMAN F., WILCOX L., TURNER T., COOPER M.: Mediaglow: organizing photos in a graph-based workspace. In *Proc. of the 13th Int. Conf. on Intelligent User Interfaces* (2009), ACM, pp. 419–424. 2

[HDBW05] HUYNH D. F., DRUCKER S. M., BAUDISCH P., WONG C.: Time quilt: scaling up zoomable photo browsers for large, unstructured photo collections. In *CHI '05: Conference on Human Factors in Computing Systems* (New York, NY, USA, 2005), ACM, pp. 1937–1940. 2

[JYC09] JANG C., YOON T., CHO H.-G.: A smart clustering algorithm for photo set obtained from multiple digital cameras. In *Proceedings of the 2009 ACM symposium on Applied Computing* (New York, NY, USA, 2009), ACM, pp. 1784–1791. 2

[KS05] KUSTANOWITZ J., SHNEIDERMAN B.: Meaningful presentations of photo libraries: rationale and applications of bi-level radial quantum layouts. In *Proc. of the 5th ACM/IEEE-CS Conf. on Digital libraries* (2005), ACM, pp. 188–196. 2

[MFGJ08] MOTA J. A., FONSECA M. J., GONÇALVES D., JORGE J. A.: Agrafo: a visual interface for grouping and browsing digital photos. In *Proc.of the Conf. on Advanced Visual Interfaces* (2008), ACM, pp. 494–495. 2

[Mic07] MICROSOFT: Photosynth. `http://photosynth.net`, 2007. 3, 6

[PCF02] PLATT J. C., CZERWINSKI M., FIELD B. A.: *PhotoTOC: Automatic Clustering for Browsing Personal Photographs*. Tech. Rep. MSR-TR-2002-17, Microsoft Research, 2002. 2

[RBSW01] RODDEN K., BASALAJ W., SINCLAIR D., WOOD K.: Does organisation by similarity assist image browsing? In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems* (2001), ACM, pp. 190–197. 2

[RCC10] RYU D.-S., CHUNG W.-K., CHO H.-G.: Photoland: a new image layout system using spatio-temporal information in digital photos. In *Proc. of the 2010 ACM Symp. on Applied Computing* (2010), ACM, pp. 1884–1891. 2

[RJL07] RENAMBOT L., JEONG B., LEIGH J.: Real-time compression for high-resolution content. In *Proceedings of the Access Grid Retreat 2007* (Chicago, IL, USA, 2007). 7

[RW03] RODDEN K., WOOD K. R.: How do people manage their digital photographs? In *Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2003), ACM, pp. 409–416. 2

[VVG06] VERGAUWEN M., VAN GOOL L.: Web-based 3d reconstruction service. *Mach. Vision Appl. 17*, 6 (2006), 411–426. 6

[vW07] VAN WAVEREN J. P.: *Real-Time Texture Streaming and Decompression*. Tech. rep., Intel Software Network, 2007. 7