PhotoCloud:

Interactive Remote Exploration of Joint 2D and 3D Datasets

Paolo Brivio ■ *Università degli Studi dell'Insubria*

Luca Benedetti ■ *ISTI-CNR Visual Computing Lab*

Marco Tarini ■ *Università degli Studi dell'Insubria*

Federico Ponchio, Paolo Cignoni, and Roberto Scopigno ■ *ISTI-CNR Visual Computing Lab*

PhotoCloud is a real-time client-server system for interactive exploration of large datasets comprising high-complexity 3D models and up to several thousand photographs calibrated over the 3D data. PhotoCloud supports any 3D models that can be rendered in a depth-coherent way (point clouds, triangle soups, and indexed triangle meshes) and arbitrary photo collections. It tolerates 2D-to-2D and 2D-to-3D misalignments. It provides scalable visualization of generic integrated 2D and 3D datasets, exploiting data duality. A set of effective 3D navigation controls, tightly integrated with innovative thumbnail bars, enhance user navigation of the data.

PhotoCloud's scope is wide because the need to manage integrated 2D and 3D sampling arises in many domains: industrial plant inspection, city management, decision-support systems for crisis management, and so on. A particularly important application context is cultural heritage, which often requires efficient, easy browsing of photograph collections, often referenced over complex 3D models. PhotoCloud effectively supports the exploration of virtual monuments, museums, archaeological sites, streets, plazas, and entire cities.

PhotoCloud provides interactive visualization and exploration of large datasets comprising thousands of calibrated 2D photographs and a complex 3D description of the scene. The system isn't tailored to any specific data acquisition process; it aims at generality and flexibility.

Dealing with Hybrid Datasets

Images and 3D models are often complementary. Images usually feature details or objects not in the related model—for example, people, cars, and trees—or depict buildings as they once were. On the other hand, models describe objects and details in a view-independent way. Such joint datasets can be generated by multiple sources and are becoming increasingly common.

Active vs. Passive Acquisition

With the spread of active 3D digitization technology, high-resolution 3D models representing large artworks or entire complexes are also becoming increasingly available. The 3D acquisition campaign is often paired with an intensive photographic sampling, which is then explicitly aligned to the 3D model.

Another possible source of data is passive acquisition, which extracts the 3D model from collections of calibrated images. Images are aligned to the model by construction. Thanks to recent advances

 Published by the IEEE Computer Society

in structure-from-motion and multiview-stereo techniques, affordable, medium-quality, colored point clouds can be extracted from collections of numerous and spatially dense photographs, even if shot under uncontrolled conditions.[1]

Visualization Challenges

Designing a hybrid visualization tool such as PhotoCloud involves several challenges.

Exploring a collection of thousands or more images poses a challenge by itself. Traditional image browser mechanisms don't scale well with the number of images. Tasks such as identifying images featuring the desired view or visualizing the whole dataset quickly become impractical. Similarly, interactive remote navigation of a complex 3D model requires effective user interfaces to aid the selection of appropriate points of view.

As we'll show, a joint visualization of 2D and 3D data, with a joint interface, offers new opportunities to address the previous two challenges. That, however, also poses challenges of its own. The data is heterogeneous, incomplete, and nonuniformly sampled and presents discrepancies between the 2D and 3D parts.

Other problems involve the data's size. For example, the system must avoid excessive lag times in a client-server context. It must also be able to manage data potentially too large to fit in graphics card RAM or even central RAM. Finally, efficient rendering techniques are necessary to cope with the size of the 3D models.

Another requirement is to avoid assumptions regarding the specific 3D acquisition technology (for example, range scanning or photogrammetry) or image acquisition methodology. To this end, it's important to support many different kinds of 3D models and registered image sets.

To face these challenges, PhotoCloud combines, adapts, and improves on several state-of-the-art techniques.[2–5]

System Overview

PhotoCloud supports the simultaneous browsing of a digital image collection I and the navigation and visualization of a digital 3D scene M. Both I and M can be large—I in terms of the number and resolution of images (up to several gigapixels), and M in terms of geometric complexity (tens of millions of 3D points or triangles). We assume that I and M have been acquired and processed in a preliminary phase that includes calibration, denoising, cleaning, and so on.

Figure 1 shows the PhotoCloud pipeline. First, the data (M + I) undergoes preprocessing. The preprocessed data is stored in a server and accessed through a client. An efficient multilevel, GPU-friendly cache system manages both I and the nodes of the multiresolution structure for M.

The client's application window (see Figure 2) comprises two areas that can be dynamically resized.

The main 3D area integrates a 3D rendering of M, the selected image in I, and *framelets*—glyph representations of other images from I that are pertinent at that moment of navigation. The framelets roughly indicate the image contents and serve as an interface for image browsing.

At the bottom of the window, the thumbnail bar shows all the images in I. As we'll show, users can select images in the thumbnail bar to project on the 3D model.

PhotoCloud Modules

PhotoCloud comprises the following tightly integrated modules.

Preprocessing 2D and 3D Datasets

The preprocessing includes constructing the thumbnails from images, recompressing high-resolution images, and zipping the files. We also precompute the average image-space depth of each image's content. We store that depth by rendering a depth buffer of the 3D model from the image viewpoint defined in the camera calibration.

Image descriptors, distances, and orderings. We associate an image with various high-level descriptors used for defining total orderings over the images and for computing distances between consecutive images for each ordering.

Our preprocessor extracts and uses these image descriptors:

- the time of the shot from the exif (exchangeable image file);
- the shot's 3D position and 2D orientation, as given by the calibration;
- the color distribution, as a 16-entry histogram of colors; and
- the spatial color layout, as a 4 × 4 downsampled image (color computations are in the LAB color space).

A descriptor D implicitly defines an image-to-image distance metric $d_D(I_a, I_b)$. We can define new metrics by linear combinations of these basic metrics. Some image descriptors come with a natural associated total ordering of images (for example, time of shot). However, we can also define a total ordering for any image descriptor

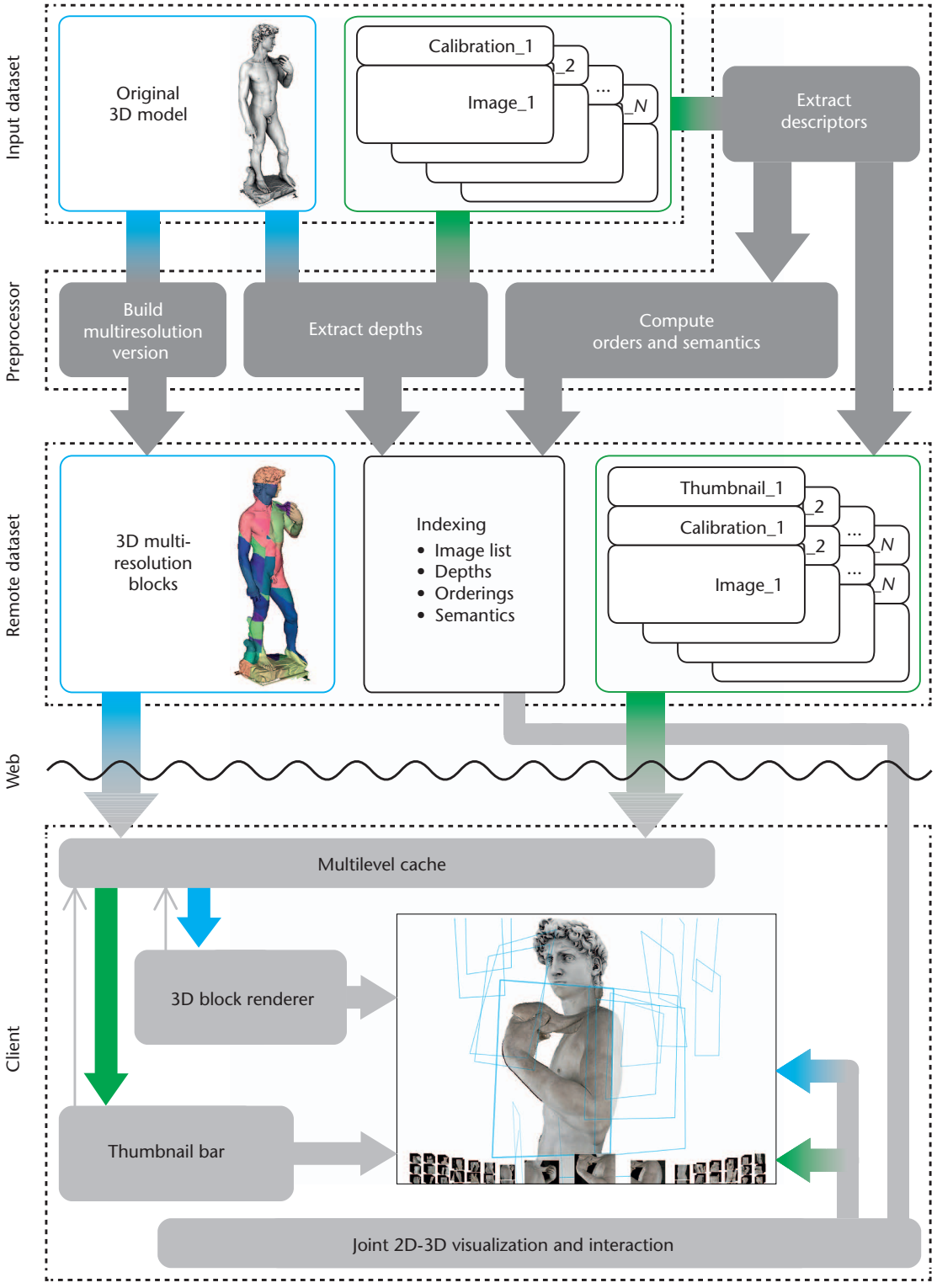


Figure 1. PhotoCloud overview. The input comprises a 3D model (for example, a polygonal mesh or point cloud) and calibrated image set. The preprocessor converts the 3D model to a multiresolution format and computes image thumbnails, the average depth, the ordering, and the semantic distance. The system then creates an index file containing references to the model and images, associating them with their depth, order, and semantic information. A remote server stores the preprocessed data. Using the client, users download data through a unified cache system. A dynamic multiresolution renderer visualizes 3D scenes featuring large models; PhotoCloud uses a thumbnail bar to visualize large collections of 2D images. The system also implements joint 2D-3D visualization and interaction mechanisms in a tightly coupled interface. Blue arrows represent the flow of 3D data; green arrows represent 2D data.

by using the associated image-to-image distance metric. For example, a global ordering could be the approximated Hamiltonian path defined according to that metric (by computing the depth-first visit of the minimum spanning tree).

Once we've ordered images according to a distance metric D_O, we compute and store semantic distances among consecutive images with a metric D_S. The metrics D_O and D_S can be chosen independently from each other.

At the end of this process, we store in the server a set of image orderings and, for each ordering, a set of distances from one image to the next.

The multiresolution 3D data structure. A brute-force approach to rendering massive digital 3D models isn't feasible, and the loss of detail produced by the simplification needed to reduce their size usually isn't acceptable. PhotoCloud adopts a clustered multiresolution data structure[2] that's suitable for encoding multiple representations of the same shape. This structure also supports interactive rendering by selecting the representation visible from the current viewpoint that best fits the rendering budget.

During preprocessing, we split the original data into blocks at different resolutions. Each block consists of roughly the same number of primitives (generally several thousands), but covering very differently sized portions of the 3D objects. We organize the geometry into a bounding-sphere hierarchy that easily allows for occlusion culling and collision detection. This data structure also allows for out-of-core management, compression, and HTTP streaming. Each block is optimized for rendering.

Multilevel Cached Memory Management

Image thumbnails, high-resolution images, and 3D data blocks compete for RAM, GPU memory, and bandwidth. PhotoCloud's priority-based cache system strives to optimally allocate resources.[3] There are four cache levels: HTTP, disk, RAM, and GPU. When new items reach the last level, the rendering updates. This way, PhotoCloud can handle datasets that are too large for GPU memory.

Implementation. Such a cache system requires managing thousands of items, allowing for frequent priority updates and locking of items, and synchronizing different threads. When rendering each frame, we use atomic integer operations instead of mutexes to lock resources. To minimize the overhead due to the priority sorting, we adopt a double heap-based queue coupled with lazy updating. Each cache level operates in its own thread, allowing for blocking on files and sockets.

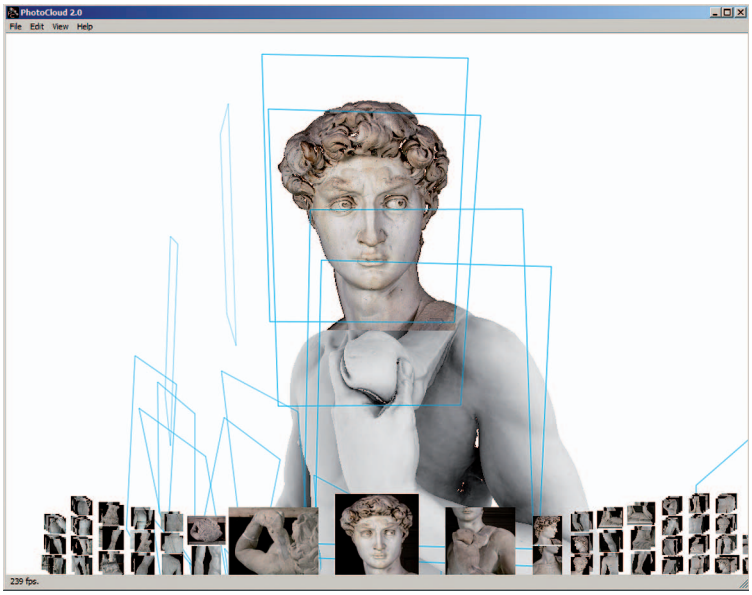


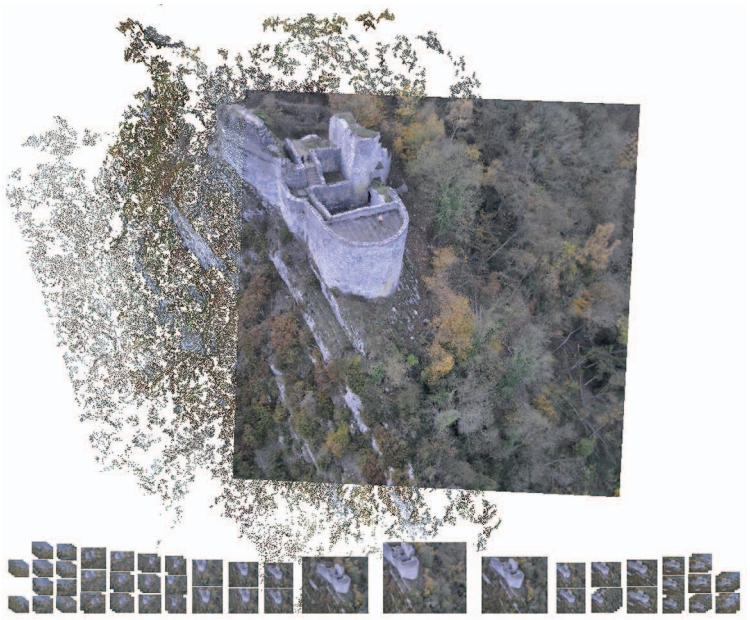
Figure 2. The PhotoCloud client main window integrates 3D and 2D representations of the object or scene under inspection. All images are visualized in the bottom thumbnail bar, which allows direct image browsing. The selected image is also rendered on the 3D area projected or overlapped on the model; framelet glyphs (in light blue) indicate available images with related content.

Priority assignments. Thumbnail images have very high priority because they're lightweight but crucial for interactions. An image's priority is proportional to the number of screen pixels it covers in the current thumbnail layout. High-resolution images have comparatively lower priority, with the selected image receiving the highest priority and the others receiving progressively lower priority according to their distance from the selected image. In this way, PhotoCloud attempts to predict the image the user will select next. The downsampled thumbnail images serve as a temporary substitute for high-resolution images not yet loaded. The multiresolution 3D subsystem determines the priority of 3D data blocks, taking in account visibility, distance, and resolution.

Data compression. Given the dataset's size, compression is a necessity. Using compression means trading speed for memory usage and bandwidth.

We store the 3D data blocks remotely in the form of vertex buffer objects. We also considered compression and decompression of 3D models, but the available techniques didn't deliver enough deflating speed.

Several image compression schemes, including combinations, are feasible. We experimented with some of them to identify a good compromise



(a)

(b)

Figure 3. Screenshots showing datasets of (a) an aerial view of a castle and (b) a city. PhotoCloud represents models as point clouds; it efficiently renders them with a multiresolution technique while dynamically rendering images on them.

between minimal signal deterioration, maximal compression rates (especially at the HTTP level), maximal decompression speed, and as-late-as-possible decompression. For most scenarios, the most appealing compromise is DXT1 compressed textures stored in zipped files. We send these files at the HTTP level, unzip them at the RAM level, and store the DXT1 images at the GPU level.

Rendering 3D Blocks

During rendering, blocks can be assembled in different combinations to produce the full model. The rendering algorithm guarantees a minimum frame rate by selecting the combination that best fits the rendering budget. The blocks' resolution decreases as they get farther from the viewpoint. This minimizes the primitive count while keeping the screen space primitive density roughly constant over the entire screen. Each block is cached in the GPU, so rendering has a low CPU load.

The final rendering primitives used depend on the data. We splat point clouds (see Figure 3) and send triangle meshes and triangle soups to the GPU as vertex buffer objects to be rasterized. PhotoCloud also supports attributes such as color, which it can also use to bake global lighting (for example, ambient occlusion), and normals defined per vertex on the models. We add standard features such as depth cueing by fog or dynamic relighting as needed.

Joint 2D and 3D Visualization

We now show how we visually merge information from the 3D model and color from the 2D pictures.

Rendering images on the 3D model. Traditional approaches, such as texture mapping or vertex color baking, can't provide a viable solution for the datasets we target, because of these problems:

- *2D-to-2D incoherence*. Photos are taken under different conditions. These differences can be drastic (for example, daytime and nighttime photos) but are part of the richness of the data. (So, you shouldn't cancel them by blending images into a final texture.)
- *2D-to-3D incoherence*. Photos represent information that's not in the 3D model (such as transient data, people, cars, and scaffolding) or that's represented at a lower resolution.
- *Misalignments*. The 2D-to-2D and 2D-to-3D alignments won't always be accurate.
- *3D incompleteness*. Even the objects in the 3D data often aren't fully represented.
- *2D uneven distribution*. The amount of photographic data is massive (gigapixels), and classic texturing approaches are infeasible. Also, photos often depict the scene from a few clustered viewpoints, overlapping some model features many times while leaving others completely uncovered.

Following Paolo Brivio and his colleagues,[4] we never mix images, to avoid color artifacts (resolving 2D-to-2D incoherence). We project only the selected image on the 3D model, as if it were cast from a slide projector (bypassing 2D uneven distribution). We also add a sky-dome mesh around the scene, which fills the screen regardless of the 3D data's completeness (addressing 3D incompleteness).

When the user views a scene from a position p_v coinciding with the position p_s from which the picture was shot, the image looks overlaid on the 3D model. When the view direction also coincides, the

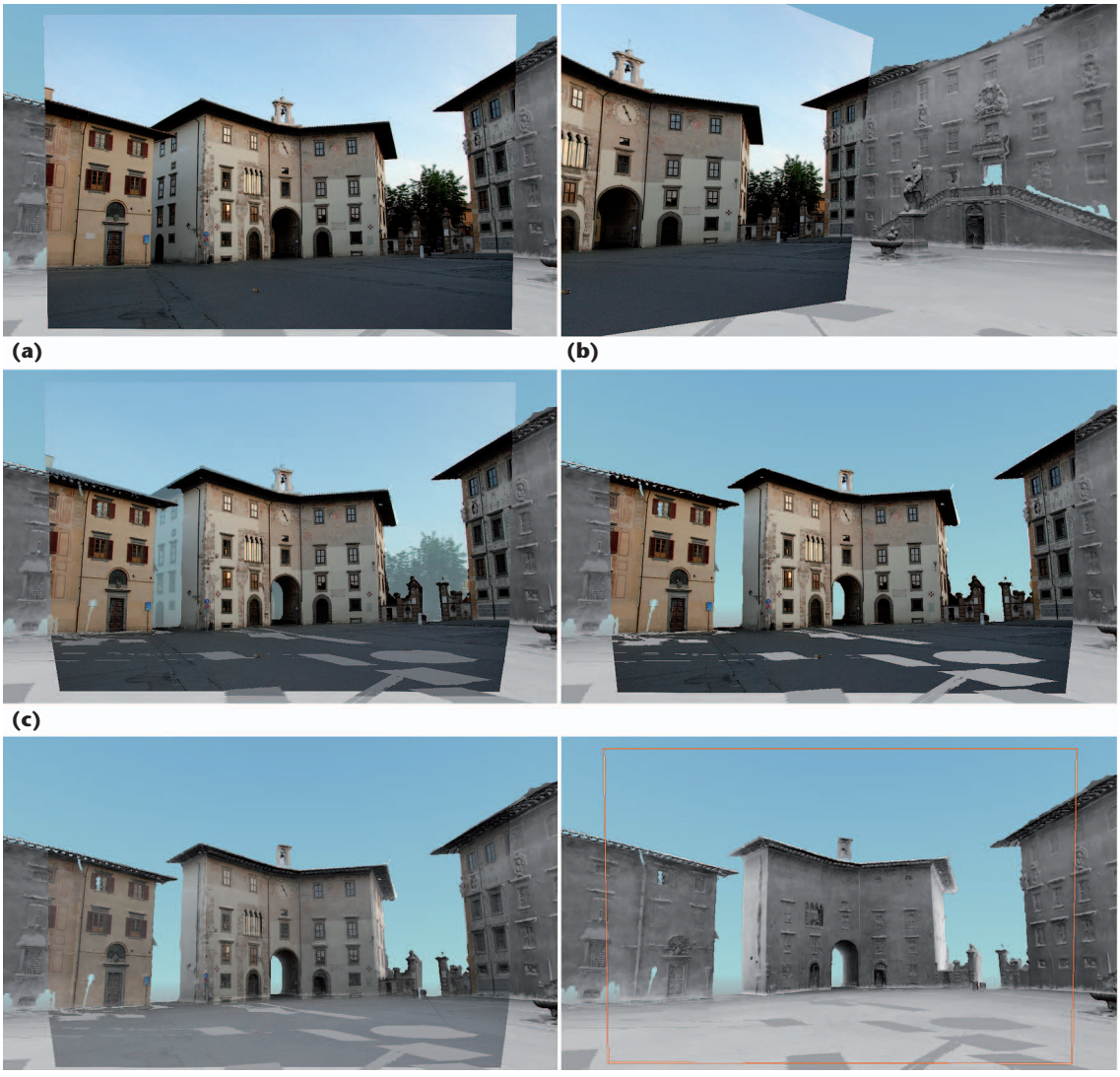


Figure 4. How texture projection changes with the view position and orientation. (a) When the current view matches the camera view, texture projection produces the effect of rendering a textured quad in front of the model. (b) The projection stays consistent for arbitrary view direction changes. (c) However, translating away from the camera view would gradually reveal mismatches between the image and 3D model. To avoid artifacts, the image gradually fades out for small view translations and more rapidly fades out in unreliable parts of the models, like in the background. (d) As the view-position discrepancy increases, the system progressively disables texture projection completely.

final effect is equivalent to splashing the selected 2D image on the main area of the screen, just like in a standard image browser (see Figure 4a). This produces no artifacts regardless of any 3D-to-2D misalignment (addressing 2D-to-3D incoherence and misalignments), even with view rotations and focal-length changes. (See Figure 4b and the video "view_rotation." All the videos mentioned in this article are at www.youtube.com/playlist?list=PLHJB2bhmgB7cmYD0ST9CEDMRv1JIX4xPH.) The illusion breaks only when the view position changes and, even then, only if there are 2D-to-3D discrepancies. So, we selectively fade out parts of the image color, in a view-dependent way.

For each pixel, the transparency level is $\alpha = C_i(p_i) \cdot (1 - C_m(p_m)) \cdot |p_v - p_s|_2$, where $C_m(p_m)$ is the confidence of the geometry model position p_m and $C_i(p_i)$ is the confidence of the image 2D position p_i being projected over p_m. The sky-dome model has the lowest confidence. So, as the viewpoint moves, the items that are depicted in the image and projected over the sky disappear more rapidly, followed, as the shot distance increases, by the 3D model's colors (see Figures 4c and 4d).

Framelets. The most common glyphs representing calibrated images in a 3D space—that is, cones and frustums—suffer several problems when, as in our

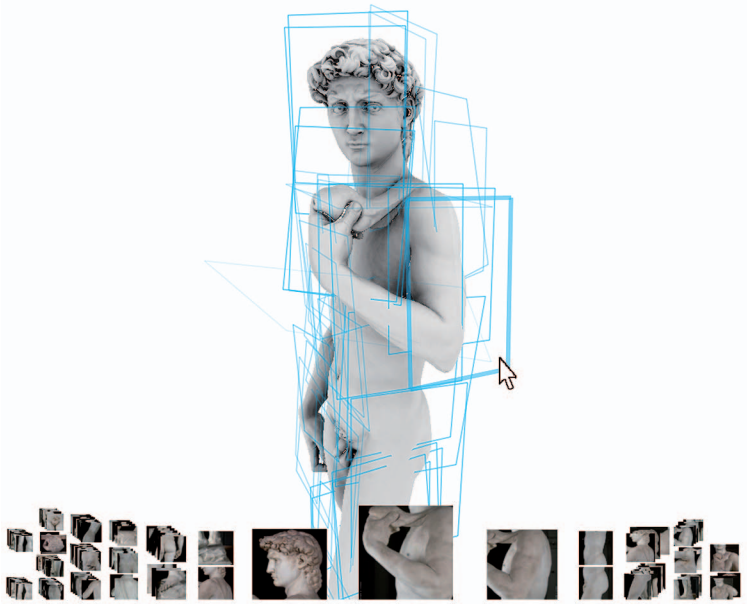


Figure 5. Framelets (the blue rectangles) indicate images facing the front of the current view and forming angles of up to 90 degrees with the current view's direction. Framelets are more opaque for similar view directions and tend to disappear for perpendicular ones. The framelet under the pointer has a slightly thicker line.

case, the scene is explored by a viewer embedded in it.

For example, *clutter* is often an issue when many glyphs overlap. Also, most glyph lines meet with nonsquare angles. Once glyphs are projected on a 2D screen with the necessary perspective projection, users can have difficulty perceiving their 3D shape owing to *projection ambiguity*. For instance, it's difficult to tell apart glyphs seen frontally from those seen from the opposite direction. Glyphs close to the current viewpoint are particularly difficult to interpret.

A related problem is the *unpredictability* of which part of the scene is observed by the shot represented by a given glyph. Some approaches texture-map the image on the frustum's base,[1] but the overlap with the 3D scene is inconsistent when the current view position differs from the image position.

Another problem is *zoom-out*. Often, glyphs tied to images featuring the part of the scene directly in front of the viewer are behind the user, out of his or her view.

The specular problem also applies, which we call an *unintended u-turn*. Images featuring objects behind the user and usually out of his or her current interest are represented by large, central glyphs.

To partly overcome these problems, we use framelets to achieve better scalability with the number of images. A framelet is the semitransparent outline of an oriented rectangle with the image's aspect ratio (see Figures 2 and 5). The rectangle is the section of the view frustum pyramid of the corresponding shot, cut at distance $dist_C$ from the camera. Its opacity depends on the observer's viewpoint.

Each framelet's transparency dynamically varies, depending on how similar its direction is to the current view direction. The more orthogonal it is, the less relevant the corresponding image is and the less opaque the framelet is. A framelet is completely transparent when seen from the back, avoiding unintended u-turns and helping to declutter the scene. Because this glyph is a simple rectangle embedded in 3D space, comprising just four orthogonal lines, it's less prone to clutter and projection ambiguity.

Given a 3D scene and a set of shots in it, the value of $dist_C$ of a shot is a percentile k of the average depth of the objects in the shot. This way, the framelet rectangle's size is linked to the shot's intrinsic parameters, in a way that helps users predict the image's content.

Framelets present a few drawbacks. The appropriate value for k depends on the application scenario. Scenes depicting one artwork usually require larger values than scenes depicting city streets or squares. We used values ranging from 0.1 for the Dubrovnik City, Cavalieri Square, and Signoria Square datasets (which we describe later) to 1 for the Michelangelo's *David* dataset (depicted in Figures 2 and 5).

We also considered using framelets shaped as quadrilaterals consisting of the intersection of the view pyramid with the plane that most closely approximates the viewed scene. This would have helped deal with content unpredictability but would have excessively exacerbated back-projection ambiguity owing to the introduced nonsquare angles. In addition, zoom-out is still an open issue.

The Thumbnail Bar

A 3D interface alone doesn't suffice to browse large image collections. Traditionally, basic 2D-image browsers just display image thumbnails over a scrollable panel. For a browsing mechanism to scale with the number of images, it must take a hierarchical, focus-and-context approach.[6] Also, the PhotoCloud client can devote only a small part of the screen to image browsing, further emphasizing the need for dynamic image hierarchies.

PhotoCloud's focus-and-context thumbnail bar[5] (see Figure 6) exploits precomputed image-to-image semantic distances and total orderings to cluster thumbnails and arrange them into proper 2D layouts. The ordering makes the browsing more intuitive. Given a thumbnail, all thumbnails on its left come before it; all the others come after it.

Figure 6. The thumbnail bar. This focus-and-context image browser arranges image thumbnails into stacked piles. The focus image is the largest thumbnail, in the middle of the bar. The others get clustered into piles whose size decreases and whose depth increases as the piles get farther from the focus. Images in the same pile are semantically close. The interface includes mechanisms to scroll, select, and preview images.

The selected image (the *focus*) is the largest thumbnail, in the middle of the bar. Thumbnails farther from the focus are smaller and clustered into deeper piles.

PhotoCloud uses additional information about the semantic distance between each pair of consecutive images to meaningfully cluster thumbnails. Semantically closer images tend to cluster, whereas distant ones go in different piles. However, a pile's visual size decreases as it gets farther from the focus.

Thumbnail selection guides navigation. When the user selects a thumbnail with the pointer, that image becomes the new focus. PhotoCloud reconfigures the layout (affecting the thumbnails' size, position, and clustering) without breaking temporal coherence. Users can also scroll the focus and preview piled thumbnails. They can dynamically change the image ordering and clustering, choosing from a pool of precomputed alternatives.

Joint 2D–3D Interaction

A central point of our system is that it offers 2D and 3D navigation metaphors that are coherently synchronized to aid browsing.

Navigating the 3D scene. Users can customize the navigation controls according to the nature of the dataset. In any case, users can employ the mouse pointer and keyboard to control the view position, orientation, and focal length.

For datasets featuring virtual environments with an open ground, we adopt a freely moving avatar metaphor. The mouse controls the view direction (the up direction is constrained), and the user can employ the keyboard to move the point of view on the horizontal plane. Another key controls the field of view, and the mouse wheel controls altitude.

For datasets featuring a single object of interest such as a statue, we adopt a trackball interface. The view position, controlled by mouse gestures, is constrained to lie on the surface of an ellipsoid around the inspected object, with the default view direction along the surface normal. Users can temporarily override the view direction with right-button mouse drags.

When the view position nears one viewpoint of an image, we enable projective texturing for that image. Also, as we mentioned earlier, moving around the 3D scene changes the visible framelets' opacity to help users find related images. When the user picks a framelet, the view flies to the associated view, also enabling texture projection (see the video "david_framelet").

3D-to-2D syncing. When the user selects a framelet (see the video "david_framelet") or when a viewpoint change enables a different texture for projection on the geometry (see the videos "david_proj" and "cavalieri_proj"), the thumbnail bar's focus changes. When the pointer moves over a framelet, the corresponding thumbnail preview is enabled in the thumbnail bar.

2D-to-3D syncing. Moving the pointer over a thumbnail highlights the corresponding framelet in the 3D scene, if that framelet is visible. Selecting a thumbnail triggers a view change in the 3D viewer through a soft transition (see the video "signoria_bar"). However, thumbnail scrolling and dragging aren't connected to the 3D view, even if they determine focus changes. That's because in this case the user is looking for a specific image by glancing at the thumbnails' visual content.

Implementation and Evaluation

We implemented PhotoCloud as an open source project. The current version offers

- support for the most common calibration and 2D and 3D data formats;

- efficient, GPU-friendly implementation, achieving real-time performance on entry-level PCs;
- cross-platform source code (available at http://vcg.isti.cnr.it/photocloud) for Windows, Mac OS X, and Unix. The multiresolution 3D model encoding is at vcg.isti.cnr.it/nexus.

We tested PhotoCloud on different machines, using five datasets:

- Dubrovnik City (6,844 photographs at different high resolutions and a cloud of 2 million points. See Figure 3b),
- Bouvignes Castle (97 photographs at a resolution of 2,000 × 2,000 pixels and a cloud of 350K points. See Figure 3a),
- Michelangelo's David (125 photographs at 2,336 × 3,504 resolution and a mesh of 56 million triangles. See Figures 2 and 5),
- Cavalieri Square (458 photographs at 3,872 × 2,592 resolution and a mesh of 15 million triangles. See Figures 4 and 6), and
- Signoria Square (507 photographs at 2,592 × 1,728 resolution and a mesh of 65 million triangles).

In all cases, the client constantly achieved more than 60 frames per second. It occupied only approximately 128 Mbytes of RAM and 80 Mbytes of GPU memory on a laptop with 1,600 × 900 resolution, a 2.6-GHz dual-core processor, and an Nvidia GeForce GT 130M graphics card.

Whereas data loading is subject to network latency, the caching mechanism and incremental data structures optimize performance with respect to the underlying network layer's limitations. They require approximately 4 percent of CPU usage to handle data across the memory levels. In our tests, a standard 100-Mbit Ethernet network connection (with a peak nominal bandwidth of roughly 11.8 Mbytes per second) always provided the necessary bandwidth to keep latencies small.

We presented PhotoCloud to several cultural-heritage experts, some of whom had no strong IT competence, and collected their impressions and comments. They all reported that the system was appealing and easy to use. Image-based navigation let the unskilled users avoid the "I'm lost" situation that often occurs when they face a 3D navigation system. In addition, the system's 3D navigation effectively helped the users select images.

User Study

We quantitatively compared PhotoCloud's image-browsing interface with that of Photosynth (www.photosynth.net), a publicly available Web-based implementation of Photo Tourism.[1] (For more on Photo Tourism and other approaches to navigating joint 2D and 3D datasets, see the sidebar.) We chose Photosynth because of its similar objectives. User studies on attitudes toward image browsing revealed that people tend to concentrate on events and thus on location cues.[7] In our case, we wanted to evaluate the effectiveness of interaction mechanisms in a 3D environment.

The Participants

Eighteen university students and young researchers participated. We separated them into three levels of self-assessed experience with 3D navigation: low, medium, and high. None had previously used either system, and none was familiar with the dataset. All had normal or corrected-to-normal vision with no color blindness.

The Procedure

All the experiments took place under the same lighting conditions in a silent room. We allowed each participant a preliminary five-minute test run on each browser, using a training dataset. Each participant received a sheet with illustrated instructions about each tool's functionalities.

Then, each participant performed a sequence of tasks on the Cavalieri Square B dataset (a subset of the Cavalieri Square dataset). It featured a square with a statue in the middle and consisted of 202 photos and a point cloud recovered from the calibrated images.

A written assignment described the four tasks:

1. Read what's written on the front of the church (which required finding any of the five pictures featuring that writing).
2. Find any of the three images that feature the left staircase of a specific building.
3. Find any of the two pictures that feature that building's entire facade (that is, both the left and right borders of the facade in a single image).
4. Determine whether there's an image showing the statue's back, and, if so, show it.

Timings started only after the participants read and understood each task. They were to work on each task until they completed it, and they received no assistance while performing the tasks.

The participants performed the tasks first on one system and then on the other. Although the two systems used the same dataset, the picture orders differed because our system computes the picture order as part of preprocessing. Because

Related Work in Navigating Joint 2D and 3D Datasets

Few image browsers support the navigation of joint 2D and 3D datasets; each pursues different goals.

Google Street View exploits accelerometers and GPS geolocation to feed a structure-from-motion system.[1] The output calibration enables blending among images overlapping in a 3D environment. The browser uses the complementary, sparse point cloud to infer planar structures' approximate position, size, and orientation. As the user navigates the scene, the browser divides it into photo bubbles (panoramas). Each bubble, separately streamed from a server, comprises photographs shot from the same position that are visible by rotating the view.

More recently, Street Slide introduced a technique to smoothly switch from panorama bubbles to a multiperspective view and vice versa to give a broader planar view of streets.[2] Google Street View and Street Slide explicitly exploit street features, so they also display information such as street names and shop signs. In contrast, PhotoCloud (see the main article) targets arbitrary 3D environments. Its navigation approach adapts to both bubble-like visits and broader views and movement.

PhotoCloud recalls Photo Tourism[3] in that it proposes explicit integration and rendering of a sparse 3D geometry with a photo collection. Photo Tourism renders the 3D model in the central part of the interface, with the selected photograph overlaid on the model. Further details about that photo appear in a side menu, including the thumbnails of photographs partly overlapping it. A complementary thumbnail bar lays out images depicting the current subject from different views. Users can browse nearby photographs, optionally presented as a slide show, and jump to remote areas through an overhead map.

PhotoCloud improves upon Photo Tourism by offering more flexible management of the thumbnail bar and increased 3D data flexibility. It also handles high-quality 3D models and provides visualization and navigation features that fully exploit dataset peculiarities.

Other visualization mechanisms tackle different subproblems. Noah Snavely and his colleagues improved navigation by photograph selection by providing controls that move the user along paths of dense collections of photographs.[4] Their approach scores photographs according to how well they depict an object of interest and uses this information to compute orbit, panorama, or best-fit paths. Another approach, *ambient point clouds*, approximates view interpolations among pairs of images.[5] It uses depth maps to render the geometry from the view of the current active photograph within a negligible error, whereas a subsampled point cloud represents the rest of the scene. PhotoCloud bypasses this problem, avoiding the need for interpolated images (see the section "Rendering images on the 3D model" in the main article).

References

1. L. Vincent, "Taking Online Maps Down to Street Level," *Computer*, vol. 40, no. 12, 2007, pp. 118–120.
2. J. Kopf et al., "Street Slide: Browsing Street Level Imagery," *ACM Trans. Graphics*, vol. 29, no. 4, 2010, article 96.
3. N. Snavely, S.M. Seitz, and R. Szeliski, "Photo Tourism: Exploring Photo Collections in 3D," *ACM Trans. Graphics*, vol. 25, no. 3, 2006, pp. 835–846.
4. N. Snavely et al., "Finding Paths through the World's Photos," *ACM Trans. Graphics*, vol. 27, no. 3, 2008, article 15.
5. M. Goesele et al., "Ambient Point Clouds for View Interpolation," *ACM Trans. Graphics*, vol. 29, no. 4, 2010, article 95.

dataset knowledge clearly influences user performance, one randomly chosen half of the participants used PhotoCloud first; the other half used Photosynth first.

Results and Discussion

Table 1 summarizes the results. Scene familiarity turned out to be not very important because the times didn't change excessively according to which system the participants used first. As we expected, the times improved with the participants' skill level, particularly with PhotoCloud. The participants' ability with the keyboard-and-mouse interface significantly affected their performance with PhotoCloud. However, this partly contrasts with our original aim because we intended PhotoCloud for a broad class of users, including both computer science and cultural-heritage people.

Considering each task separately, the differences in the times are due partly to the different 3D-navigation mechanisms and partly to the visualization techniques. In Photosynth, the images' positions constrain the movement of the virtual 3D view, whereas PhotoCloud supports free view selection through keyboard-plus-mouse and frame-let interactions. In general, PhotoCloud allows users a larger variety of actions. For example, they can view the 3D model from points and angles not pictured in any image. Or, they can virtually walk in the 3D environment toward the part of the scene they're interested in before selecting the target image.

Our intent is that this approach should reasonably reduce the time to complete the tasks. However, the participants' performance noticeably deteriorated when they solved tasks in which the target image had to match a specific view (tasks 1 and 3). This was mainly because we displayed the

Table 1. The average time the participants took to complete each task (see the section "The Procedure"). The second column indicates how familiar the participants were with 3D interfaces. The best results are in bold.

Task	Participant skill	Time (sec.)		*B*/*A* (%)
		Photosynth (*A*)	PhotoCloud (*B*)	
1	Low	**34.4**	96.0	279
	Medium	**29.0**	50.5	150
	High	**21.6**	31.1	130
2	Low	247.4	**29.2**	12
	Medium	240.5	**23.8**	10
	High	149.0	**13.0**	9
3	Low	**59.2**	128.0	216
	Medium	47.7	**45.7**	96
	High	29.0	**23.6**	81
4	Low	**58.0**	71.4	123
	Medium	45.7	**24.8**	54
	High	29.0	**15.6**	54

framelets at 0.1 × $depth_C$ to prevent cluttering in areas with a higher density of shots.

Specifically, while solving task 1 with PhotoCloud, the participants tended to move near the front of the church instead of selecting an image and zooming in to read the writing. This resulted in longer times. A similar misunderstanding occurred during task 3. In these cases, constraining the view to the available images can noticeably decrease the times (by nearly two-thirds, for task 1 for low-skilled users), thanks to the spatial metaphor. However, this happens only if the frustums of that image and other images intersect.

In contrast, free 3D movements can reduce the search time up to 90 percent. In task 2, each requested picture featured the staircase (occupying the largest part of the picture) in the foreground and a more distant building in the background. During the tests, all but one participant initially used the Photosynth 3D browser and overhead map to reach that picture but finally relied on the 2D image browser to find it. With PhotoCloud, moving near the desired location and selecting the correct framelet accomplished the task.

During the experiments, we registered which interface mechanisms each user tried and which one was ultimately successful. In 75 percent of the cases with PhotoCloud, framelets were successful, but their use always followed either 3D free navigation (90 percent) or 2D browsing (10 percent). In 20 percent of the experiments, the participants used mainly the embedded 2D browser. On the other hand, when using Photosynth, the participants often switched between the 3D view and the overhead map, sometimes resorting to the conventional 2D browser, which proved time-consuming. With PhotoCloud, the participants mostly used the 2D-3D interface; the integration of the various tools in the interface helped them switch between different, effective navigation strategies.

After the test, we asked the participants for qualitative comments and impressions; most argued that picture localization was more natural and easier in PhotoCloud. As they pointed out, this is probably due to the ability to freely move in the scene in PhotoCloud. Photosynth only lets users jump from one picture to another, which isn't always the one they expect. The participants also reported that PhotoCloud's visualization techniques helped them better understand how the scene was structured, which objects were in it, and how to reach them.

As future work, we plan to test alternative solutions to framelet cluttering (for example, grouping framelets when they're seen from a distance). We could also evaluate the effectiveness of constrained versus free 3D movements in datasets featuring scenes consisting of multiple rooms, where movement is constrained through certain passages.

In addition, we could improve the policies for defining the ordering of the image dataset and add mechanisms to select image subsets. Image paths could be precomputed[8] and exploited to produce better transitions between images. Collision detection could help 3D navigation of a scene (for example, to avoid passing through walls).

Acknowledgments

The research leading to these results has received funding from the Tuscany Region (POR CREO FESR 2007-2013, Visito Tuscany project) and European Commission (FP7 IST IP, 3DCOFORM project, grant 231809). We thank Noah Snavely for the Dubrovnik City dataset, Visual Dimension for the Bouvignes Castle dataset, and Stanford University and Museo dell'Accademia di Firenze for the Michelangelo's David *dataset.*

References

1. N. Snavely, S.M. Seitz, and R. Szeliski, "Photo Tourism: Exploring Photo Collections in 3D," *ACM Trans. Graphics*, vol. 25, no. 3, 2006, pp. 835–846.
2. F. Ponchio, "Multiresolution Structures for Interactive Visualization of Very Large 3D Datasets," PhD thesis, Clausthal Univ. of Technology, Dec. 2008.

3. "Generic Cache System," ISTI-CNR Visual Computing Lab, 2012; http://vcg.isti.cnr.it/gcache.
4. P. Brivio et al., "Joint Interactive Visualization of 3D Models and Pictures in Walkable Scenes," *Proc. Eurographics Posters*, Eurographics Assoc., 2012, pp. 35–37.
5. P. Brivio et al., *PileBars: Scalable Dynamic Thumbnail Bars*, tech. report 2011-TR-006, ISTI-CNR, 2011.
6. A. Cockburn, A. Karlson, and B.B. Bederson, "A Review of Overview+Detail, Zooming, and Focus+Context Interfaces," *ACM Computing Surveys*, vol. 41, no. 1, 2009, article 2.
7. M. Naaman et al., "Context Data in Geo-referenced Digital Photo Collections," *Proc. 12th Ann. ACM Int'l Conf. Multimedia* (Multimedia 04), ACM, 2004, pp. 196–203.
8. N. Snavely et al., "Finding Paths through the World's Photos," *ACM Trans. Graphics*, vol. 27, no. 3, 2008, article 15.

Paolo Brivio *collaborates with the ISTI-CNR Visual Computing Lab on the 3DCOFORM and Visito Tuscany projects. His research interests are computer graphics and computer vision. Brivio received a PhD in informatics from Università degli Studi dell'Insubria. Contact him at paolo.brivio@uninsubria.it.*

Luca Benedetti *is a PhD student at University of Pisa, working on color processing for dense stereo matching algorithms, large-scale automatic image calibration, and smart navigation of heterogeneous image collections. He collaborates with the ISTI-CNR Visual Computing Lab on the Visito Tuscany and Indigo projects. Benedetti received a master's in computer science from Università di Pisa. Contact him at luca.benedetti@isti.cnr.it.*

Marco Tarini *is an assistant professor at Università degli Studi dell'Insubria and an associate researcher with the ISTI-CNR Visual Computing Lab. His research interest is computer graphics and its applications, particularly geometric modeling, real-time rendering, and large-dataset visualization. Tarini received a PhD in computer science from the University of Pisa. He's a Marie Curie fellow and received the 2006 Eurographics Young Researcher Award. Contact him at marco.tarini@isti.cnr.it.*

Federico Ponchio *is a research scientist at the ISTI-CNR Visual Computing Lab. His research interests include multiresolution representations, interactive visualization, geometric processing, and Web applications. Ponchio received a PhD in computer graphics from the University of Claustal. Contact him at federico.ponchio@isti.cnr.it.*

Paolo Cignoni *is a senior research scientist at the ISTI-CNR Visual Computing Lab. His research interest is computer graphics, particularly visualization and processing of huge 3D datasets, 3D scanning for cultural heritage, and scientific visualization. Cignoni received a PhD in computer science from the University of Pisa. He received the 2004 Eurographics Young Researcher Award. Contact him at paolo.cignoni @isti.cnr.it.*

Roberto Scopigno *is a research director at ISTI-CNR and leads the ISTI-CNR Visual Computing Lab. His research deals with 3D scanning, surface reconstruction, multiresolution data modeling and rendering, scientific visualization, and cultural heritage. Scopigno graduated in computer science at the University of Pisa. He received the 2008 Eurographics Outstanding Technical Contribution Award and has served as the chair of the Eurographics Association, coeditor in chief of* Computer Graphics Forum, *and a member of the editorial board of the* Journal on Computing and Cultural Heritage. *Contact him at roberto.scopigno @isti.cnr.it.*