Joint interactive visualization of 3D models and pictures in walkable scenes

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

[1]Università degli Studi dell'Insubria, Varese, Italy [2]ISTI-CNR, Pisa, Italy

Abstract

The 3D digitalization of buildings, urban scenes, and the like is now a mature technology. Highly complex, densely sampled, reasonably accurate 3D models can be obtained by range-scanners and even image-based reconstruction methods from dense image collections. Acquisition of naked geometry is not enough in Cultural Heritage applications, because the surface colors (e.g. pictorial data) are clearly of central importance. Moreover, the 3D geometry cannot be expected to be complete, lacking context, parts made of materials like glass and metal, difficult to reach surfaces, etc. Easily captured photographs are the natural source of the appearance data missing in the 3D geometry. In spite of the recent availability of reliable technologies to align 2D images on 3D data, the two sides of the dataset are not easy to combine satisfactorily in a visualization. Texture mapping techniques, perhaps the most obvious candidate for the task, assume strict content consistency (3D to 2D, and 2D to 2D) which these datasets do not and should not exhibit (the advantage of pictures consisting in their ability to feature details, lighting conditions, non-persistent items, etc. which are absent in the 3D models or in the other pictures). In this work, we present a simple but effective technique to jointly and interactively visualize 2D and 3D data of this kind. This technique is used within "PhotoCloud" [IV12], a flexible opensource tool which is being designed to browse, navigate, and visualize large, remotely stored 3D-2D datasets, and which emphasizes scalability, usability, and ability to cope with heterogeneous data from various sources.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

In several applications, including ours [IV12], it is necessary to visualize 3D models of real objects or scenes together with calibrated photo collections encoding their appearance under the most varied conditions (time, context, light, etc.). We target 2D-3D datasets which exhibit:

A *2D-2D incoherence:* photos are taken under different conditions. These differences can be drastic (think of daytime/nighttime pictures), should be considered part of the richness of the data, and should not be canceled by the blending approach used to produce the final texture;

B *2D-3D incoherence:* images represent information that does not exist in the 3D model (like transient data, people, cars, scaffolding, etc.) or that is represented at a lower resolution (as small details are usually lost in the 3D acquisition);

C *mis-alignments:* the 2D-2D and 2D-3D alignments cannot be expected to be always accurate;

D *3D incompleteness:* though the 3D model can be highly sampled, there's no guarantee that it represents the scene surface with continuity, featuring holes corresponding to non-modeled regions;

E *2D uneven distribution:* the amount of photographic data is massive (gigapixels) and highly complex to be managed with classical texturing approaches. Also, photos often depict the scene from a few clustered points of view, overlapping many times over some model features, while leaving others completely uncovered.

The traditional solution for combining 2D images with 3D models consists in using the images as texture maps for the model, or equivalently to "bake" color from the pictures over the 3D model as vertex colors, in a fixed, view-independent way (e.g. during a preprocessing stage). There are several techniques which blend RGB information from multiple source-images. These approaches are not feasible for the targeted datasets.

© The Eurographics Association 2012.

At the other extreme, another plain solution is to embed images in the scene in the form of flat rectangular textured panels, sized and placed coherently with internal and external calibration of shots [SSS06,Mic07]. This completely bypasses all the coherency problems and presents image contents more directly to the user. On the other hand, this integration with the 3D model is lousy, and views close to, but not matching the shot positions, can be confusing.

Several other approaches have been proposed to tackle some or all of the listed problems in specific contexts. For example, [GAF*10] proposes to solve the 2D-3D incoherence during quick transitions from a picture to the next.

Here we show a simple, conservative solution to address the issue. It emphasizes lack of visual artifacts with any view (still or in movement), sacrificing completeness of the visualization, when necessary (i.e. hiding parts of the dataset).

2. Our recipe

We choose that, at any moment, color data is to be borrowed from at most one given image. This choice is imposed by (A): we need to address datasets in which the actual image content can sensibly vary from one shot to the other, even though the same portion of the scene is featured. For example, image collections can feature daylight VS night scenes, different seasons, etc. Any attempt to merge contents from more than one image at a time would be prone to severe artifacts. In our interface, the one used image is the "currently selected" one, which the user selects during a navigation/browsing session in one of several direct or indirect ways (e.g. picking it directly, or automatically chosen by proximity/relevance).

Then, we adopt a standard projective texturing approach [SKv*92], which consists in rendering the 3D geometry and accessing, for each produced pixel, at the color in the selected RGB image according to its projected position on screen. The effect is equivalent to cast the image as a slide from a virtual projector into the 3D scene. A skydome mesh (a large sphere encompassing the entire model and the viewer), is added to the scene, so that a background fills the entire screen, and the projective texture is cast over it in places not covered by the original geometry (including holes); this allows also to paint the sky, backgrounds, or any other entity non present in the 3D model.

The main observation behind this approach is that, when the 3D scene is viewed from exactly the same view *position* from which the picture was shot, the image content appears just as if the image were depicted over a flat 2D panel (see Fig. 1-a), thus inheriting all the advantages (e.g. in terms of clarity) of that approach. The flat panel showing the photo looks exactly like a window through which the scene is seen.

We note that this property holds in an exact way regardless of *any* discrepancy between the image content and 3D scene, fully addressing point (B) at least in case of viewpoint matching.

Further, this still works even if we allow to change *view direction* or the *focal length* at will (as long as the *viewposition* remains fixed). For this reason in our interface the user can rotate freely the view direction and to change focal length (via zooming mechanisms) with no other effect (see Fig. 1-b).

The illusion breaks only in presence of both (i) position changes and (ii) 2D-3D discrepancies (or equivalently calibration misalignments). Moreover, with small amounts of *either* (i) or (ii) the degradation is still small. These considerations dictate our policy.

As the distance between the viewpoint p_v and position of the shot picture p_p increases, we progressively fade-out the color of the projective texture, with a transparency level α ($\alpha = 0$ means fully opaque). We use

$$\alpha = (1 - C_m(p_m) \cdot C_i(p_i)) \cdot |p_v - p_s|_2$$

where $C_m(p_m) \in [0,1]$ is the confidence of the geometry at that model position p_m, and $C_i(p_i) \in [0,1]$ is the confidence associated to the image 2D position p_i being projected over p_m.

If confidence values are defined over the 3D model (e.g. as per-vertex values), these can be used as C_m. The larger difference in that channel, however, the smaller difference is the one between the skydome and the 3D model. In our experiment we just assigned two different constant values: a low one for the skydome and a high one for the model.

Per pixel values $C_i(p_i)$ can be computed in various ways (e.g. checking for reprojection consistency). We use constant per-image values reflecting image calibration confidence.

The net effect is that as viewpoint is moved away, the sky and the background items depicted in the image disappear more rapidly, followed by the colors in the 3D model (see Fig. 1-d-e-f).

References

[GAF*10] GOESELE M., ACKERMANN J., FUHRMANN S., HAUBOLD C., KLOWSKY R., STEEDLY D., SZELISKI R.: Ambient Point Clouds for view interpolation. In *ACM SIGGRAPH 2010 papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 95:1–95:6. 2

[IV12] ISTI-VCG: PhotoCloud. `http://vcg.isti.cnr.it/photocloud`, 2012. 1

[Mic07] MICROSOFT: Photosynth. `http://photosynth.net`, 2007. 2

[SKv*92] SEGAL M., KOROBKIN C., VAN WIDENFELT R., FORAN J., HAEBERLI P.: Fast shadows and lighting effects using texture mapping. *SIGGRAPH Comput. Graph. 26* (July 1992), 249–252. 2

[SSS06] SNAVELY N., SEITZ S. M., SZELISKI R.: Photo Tourism: exploring photo collections in 3D. *ACM Transaction on Graphics 25* (July 2006), 835–846. 2

© The Eurographics Association 2012.

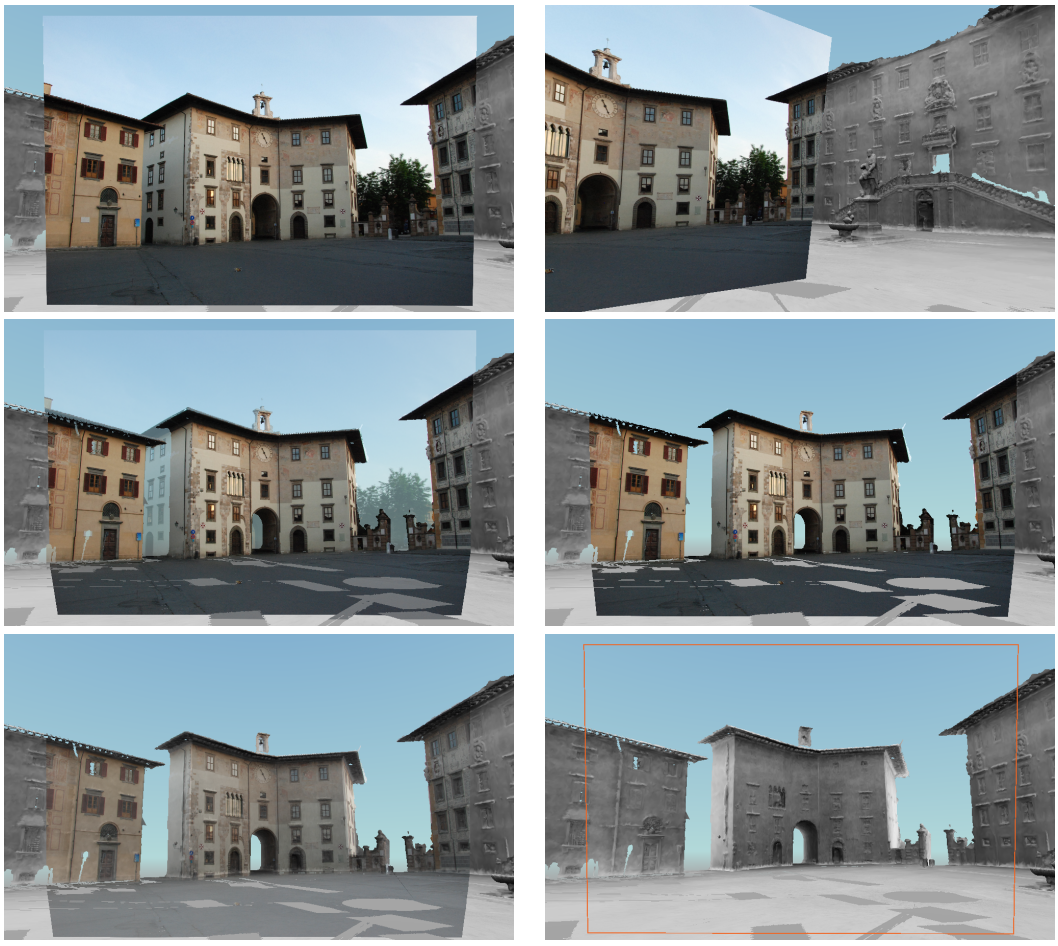

Figure 1: *All screenshots refer to the same selected image projected on the 3D model plus an additional skydome background. Only the view position and orientation change. When the current view exactly matches the camera view (a), projecting the texture has the same effect of rendering a textured quad in front of the model. The projection is still consistent for arbitrary view direction changes (b), but translating the point of view with respect to the camera view reveals mismatches between the image and the 3D model, the more the distance from the viewpoint. To hide projection artifacts, the image is gradually faded out for small view translations, more rapidly at the background (c, d). When the view-position discrepancy increases the texture projection is progressively disabled (e, f).*

© The Eurographics Association 2012.