Interactive HDR Image-Based Rendering from Unstructured LDR Photographs

Loubna Lechelek, Daniel Meneveaux, M. Ribardière, Romuald Perrot, Chaouki Babahenini

Computers & Graphics (2020)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Interactive HDR Image-Based Rendering from Unstructured LDR Photographs

Loubna Lechlek[a], Daniel Meneveaux[b,*], Mickaël Ribardière[b], Romuald Perrot[b], Chaouki Babahenini[a]

[a] *University of Biskra, Algeria*
[b] *University of Poitiers, XLIM Laboratory, France*

ARTICLE INFO

Article history:
Received March 11, 2020

Keywords: Image-based rendering, Point-based proxy, high-dynamic range images, interactive rendering

ABSTRACT

This paper proposes an interactive High Dynamic Range (HDR) image-based rendering system, dedicated to manually acquired photographs. It only relies on a point-based geometric proxy and the original photographs, calibrated using a standard structure-from-motion process. First, a depth map is estimated for each new rendered viewpoint, based on the point cloud. Second, pixel values are reconstructed from the original photographs, using a blending model that also handles occlusion. Our system can be used for producing HDR images from several series of unaligned photographs with different exposures. As shown in the results, it proves efficient with various types of objects, implemented in WebGL, making it practical for many purposes.

1. Introduction

Many authors have tackled the problem of producing photo-realistic 3D models and images from photographs. This subject is motivating for many fields such as cultural heritage and virtual museums, architectural capture for rehabilitation, or virtual tourism. One of the core questions for these approaches concerns the geometric reconstruction and its representation. Automatically reconstructing a precise 3D mesh remains a difficult problem [1]. In many cases, the reconstructed mesh introduces undesirable edges in the object geometry (sometimes even perpendicular to the actual object ones), and the resulting 3D object is finally flawed with additional or missing volumes. In addition, attaching a fixed texture to a mesh determines a level of detail in the object appearance, some details observed on the original photographs may thus be lost, as well as view-dependent appearance effects. Image-based rendering (IBR) methods offer elegant alternatives for observing objects or environments from photographs. Early approaches create novel views from a set of calibrated photographs [2, 3, 4], based on a dense and calibrated capture to achieve high quality images for new viewpoints; Geometric proxies are required to minimize visual artifacts due to parallax errors [5, 6] and to improve the rendered images quality [7, 8, 9]. The geometric proxy corresponds to a 3D mesh in most cases.

Besides, structure from motion (SFM) [10, 11, 12], multi-view stereo (MVS) [13, 14] and surface reconstruction [15, 16] have been successfully employed for calibrating large photo collections and retrieving geometric information. Some authors propose visualization systems based on image warping [10, 11] and super-pixel representations [17, 18, 19], and the obtained point cloud is only employed as a coarse representation. Dense point clouds contain a reliable information, fast to render [20]. Creating a polygonal mesh remains an additional process that may impair the geometry, with edges that do not actually appear on the real object. The work proposed in this paper directly exploits the point cloud produced by the multi-view stereo [14] as a geometric proxy (thus avoiding a mesh reconstruction phase), in an interactive IBR navigation system, from uncalibrated photographs. Figure 1 shows several images generated by our rendering system, comprising large viewpoint changes. The proposed framework also provides a straightforward method for automatically handling High Dynamic Range (HDR) images reconstruction and interactive visualization. View-dependent

*Corresponding author: Tel.: +33 549 497 438
e-mail: daniel.meneveaux@univ-poitiers.fr (Daniel Meneveaux)

Fig. 1. Images from our interactive navigation system, based on 22 photographs.

textures described by Yang *et al.* [21] are based on point splatting for live scenes, with specific acquisition systems. Instead, our system handles unstructured photographs, acquired with a standard hand-held camera. It is capable of automatically reconstructing HDR images that are not initially aligned, and the user can freely navigate around the object.

Each new image is built upon the point based rendering algorithm of Marroquim *et al.* [20] for estimating the depth map associated with each photograph, as well as a per-pixel depth for the generated images. The object surface observed through pixels for a new viewpoint is identified on the set of photographs thanks to occlusion management, and the photograph pixels are blended on the fly thanks to the chosen strategy. More precisely, our main contributions are:

- An interactive free viewpoint visualization system, that handles parallax constraints based on a depth map constructed from a point cloud;
- An improvement of the point based rendering algorithm of Marroquim *et al.* [20], that highly reduces flickering artifacts on object silhouettes;
- A method for producing HDR images from hand-held camera unstructured photographs without any positioning constraint.

We present results for a variety of scenes, demonstrating that our approach provides convincing results with free viewpoint navigation, even with only few photographs, and distant viewpoints. The use of HDR images allows the user to control the tone-mapping parameters during navigation. A video illustrating our results on the test scenes can be found at https://vimeo.com/307643834.

The remainder of this paper is organized as follows: Section 2 discusses the existing methods; Section 3 presents an overview of our pipeline; Section 4 describes our image based rendering approach; Section 5 describes our HDR image construction and rendering strategies; Section 6 presents the obtained results; Section 7 provides our conclusion and future work.

2. Related work

This paper focuses on image-based rendering methods dedicated to free viewpoint navigation around still objects acquired by cameras, without changing their illumination. This area has focused a lot of attention for more than two decades since they offer interesting tools for manipulating photographs in order to enhance navigation through real or augmented environments. Historically, they have been introduced with the plenoptic function [2] and view-dependent textures [4] and gave birth to light fields and lumigraphs representations [3, 7]. Rendering from photographs remains difficult since the generation of novel views may result in ghosting artifacts due to parallax errors. This is the reason why several authors have combined the photographs with a geometric proxy [4, 3, 7, 8, 21, 6], that handles occlusions and thus importantly improves the resulting rendered image quality. Buehler *et al.* [8] further extends the method to unstructured sets of calibrated photographs, with a reconstructed geometry. Unfortunately, the reconstruction of a precise geometric mesh still is an active research field, and the rendering process may also be impacted by the scene complexity since meshes are often composed of several millions of triangles. Furthermore, meshing is often constructed from a 3D point cloud, and the process hardly reconstruct sharp edges. This lack of precision leads to artifacts since they may add or remove material volumes, thus corresponding to an incorrect object geometry. This problem has been addressed by several authors [22, 19] using with a maximum a posteriori (MAP) estimate that accounts for uncertainty in the reconstructed geometry and image sensor noise. Eisemann *et al.* [9] use optical flow to correct for inaccurate object geometry, and Grégoire *et al.* [23] add more constraints on image gradients to reduce the visual artifacts caused by the discontinuities in the blending weights of the input views. However, the temporal complexity remains the major drawback of these approaches. Davis *et al.* [24] introduce an unstructured light field acquisition system with a hand-held camera for managing many images with a more precise representation, but the system is more adapted to a high number of images. Penner and Zhang [25] propose a rendering approach dedicated to challenging scenes, in which they use a soft 3D volumetric reconstruction instead of 3D meshes. However, despite their approach provides high quality results, it is still limited to a range of observer viewpoints due to discretization.

Besides, computer vision methods have been developed with various types of acquisition systems, often producing dense

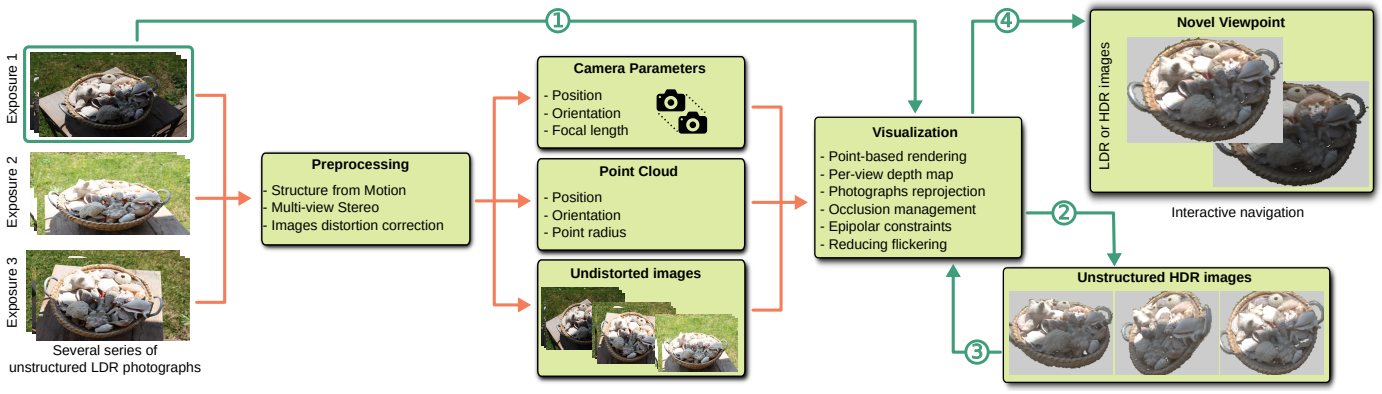


Fig. 2. Our interactive HDR image-based rendering system: Several series of LDR unstructured photographs, each with a fixed exposure, given as input to a SFM and MVS software; The result is a dense point cloud and the projection matrices associated with the cameras. (1) The viewpoints corresponding to one of the series of photographs is employed for aligning the images corresponding to the other LDR series (2), thanks to our rendering process. The aligned LDR images are used to produce the HDR images. (3) The resulting set of unstructured HDR photographs can be used as input of our interactive rendering system (4). Note that our system allows to observe view-dependent material effects, and improves under or over-exposed regions thanks to the HDR reconstruction process.

point clouds [10, 14, 26, 27]. SFM methods offer a way to both calibrate photographs and generate point clouds from hand-held cameras [10, 11]. The rendering approaches employ warping techniques to align photographs with the point cloud for producing new views [11]. However producing accurate images from such a structure requires numerous and complex processing [18, 19, 6], including manual processing [17, 28], or deep learning [29], and the construction of a mesh.

Another approach consists in visualizing the point cloud directly, in association with reconstructed textures, such as SIFT rendering [30]. However with this latter approach, texture resolution is not view-dependent. Yang *et al.* employ view-dependent textures with a point-based rendering system [21]. It is based on a standard splatting method, dedicated to live scenes and relighting, involving complex acquisition system with a depth associated with images, a segmentation process for handling large splats, and the blending of pixels relies on the observation angles and visibility, without managing resolution issues.

HDR image reconstruction from photographs taken freehand has been intensively studied [31, 32, 33]; they are currently available on most smartphones, when the viewpoint varies only slightly. Most of them are based on the idea of alignment of all the input photographs to a selected reference before merging an HDR image. For instance, the method proposed by Tomaszewska *et al.* [34] use SIFT descriptors while Zimmer *et al.* [35] use optical flows to align the images. More recently Sen *et al.* [36] propose to integrate alignment and reconstruction using a variant of PatchMatch algorithm.

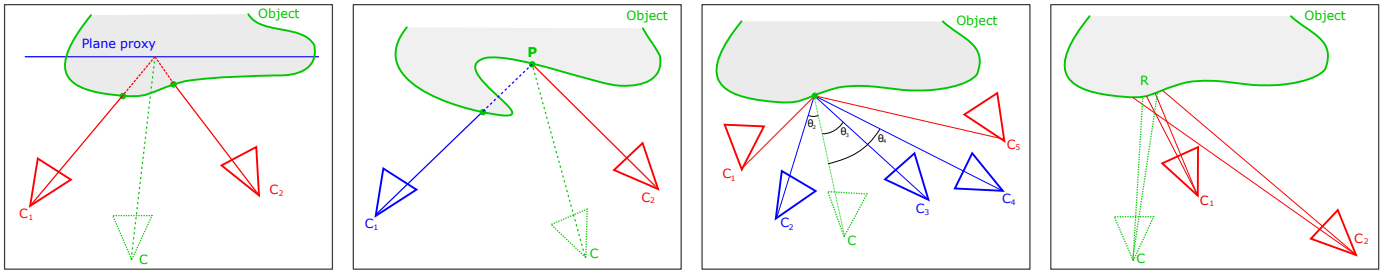
In this paper, we describe an interactive rendering system, based on only few photographs acquired with a hand-held camera, and robust to large camera motions. Our approach handles several series of photographs that do not require to be aligned for automatically reconstructing HDR images from any viewpoint. The proposed system relies on the point cloud splatting proposed by Marroquim *et al.* [20], with an additional pass for reducing depth flickering; it is used to reconstruct a depth map for each photograph, as well as a depth map for each new viewpoint during visualization. For each pixel, all the photographs are employed and the corresponding pixels are blended according to a blending strategy that handles depth management and occlusions [8, 6]. In addition, our system allows an automatic reconstruction and interactive rendering of HDR images, with completely unaligned viewpoints.

3. System overview

Figure 2 presents the main architecture of the general process, including our visualization system, core of this work. The set of input photographs is first processed using an existing SFM software [37], followed by an MVS stereo algorithm [14] for producing a dense point cloud corresponding to the surface of the captured object [38], as well as an estimation of intrinsic and extrinsic parameters for each camera viewpoint. A radius is associated with 3D points, in order to ensure overlapping on the object surface [39]. The resulting camera calibration and the dense point cloud are directly used as input by our visualization system. The 3D point cloud is employed directly as a geometric proxy for reconstructing per-pixel depth from any viewpoint.

The 3D point cloud is projected onto image space, thanks to a modified version of the point based rendering method introduced by Marroquim *et al.* [20], improved by an additional rendering pass, with only little impact on visualization performances. We have chosen this splatting method for its effectiveness with dense point clouds. First, it is applied to each photograph, so as to construct a per-view depth map. Second, during interactive navigation, it is used for estimating per-pixel depth on the novel views, and the final color of pixels is obtained by inverse projection on the original photographs, taking occlusions into account and weighting the color contributions of relevant photographs according to the blending strategy. This latter corresponds to a slight variation of the one proposed previously [8, 6].

Our HDR reconstruction process and interactive rendering system relies on this former process. Let us consider several series of unstructured photographs, each of which is acquired with



(a) Unsuitable plane proxy (b) Occlusion management (c) Orientation variations (d) Resolution variations

Fig. 3. Geometric proxy and important configurations: (a) With an inaccurate geometric proxy, the appearance of a point P observed through an arbitrary camera C is not properly reconstructed from other views C_1 and C_2; (b) Occlusions should be accounted for, in order to avoid ghosting effects; In this example the pixel values corresponding to C_1 projected onto P are occluded and should thus not be used. (c) The appearance associated with P observed from a camera C can be reconstructed from many views; View-dependent variations depend on the observation angles. (d) Relative camera positions also introduce resolution differences so that observing distances should also be considered.

a fixed exposure. The SFM reconstruction process is applied on the whole set of photographs, in order to calibrate all of them and produce a single point cloud. For any arbitrary viewpoint, one LDR image can be constructed from each series of photograph using our rendering pipeline; The HDR image can thus be obtained by merging the obtained LDR images. This process can be performed on the fly, but better performance has been obtained with precomputed HDR images, based on the viewpoints associated with one of the LDR series. The resulting HDR images can then be employed for interactive navigation, with an exposure defined by the user.

4. Interactive free-viewpoint rendering

This section describes our IBR process, considering either LDR or HDR images. It relies on the following stages: (i) Point cloud rendering, for estimating per-pixel depth and surface orientation; (ii) Per-pixel back-projection onto the object surface (using pixel depth); (iii) Occlusion management and pixel blending from the original photographs.

Figure 3 illustrates the important criteria that have to be accounted for during the rendering process. The geometric proxy is employed to discard useless photographs, when the observed surface is occluded, (as illustrated in Figure 3.b). The observation orientation corresponding to the new viewpoint should also be compared to the photographs orientation, in order to properly blend the original pixels color and better capture the object reflection variations (Figure 3.c). Finally, the resulting texture resolution also depends on camera distance during pixels blending (Figure 3.d). All these criteria have to be managed for rendering the scene with the best possible level of detail. The remaining of this section describes our choices.

Point cloud rendering

The geometric proxy is employed to determine the object surface visible through each pixel of the novel view, in order to properly choose and blend the corresponding observed pixels on the original views (Figure 3). The point cloud is managed thanks to projection method proposed by Marroquim *et al.* [20]. The main idea of the algorithm is to project the points on the screen plane and interpolate the attributes (normal, depth, etc.) using a pull-push algorithm. Figure 4 illustrates the principle: The pull phase fills image pixels in the image pyramid, from the full-resolution image up to the lower resolution image. The upper image pixel values in the upper level image correspond to an average of the valid lower image pixel values. The push phase fills the interpolated pixel values from the low resolution image down to the full resolution. The ellipse corresponding to each projected surface disk is employed to limit the region of influence of projected point.

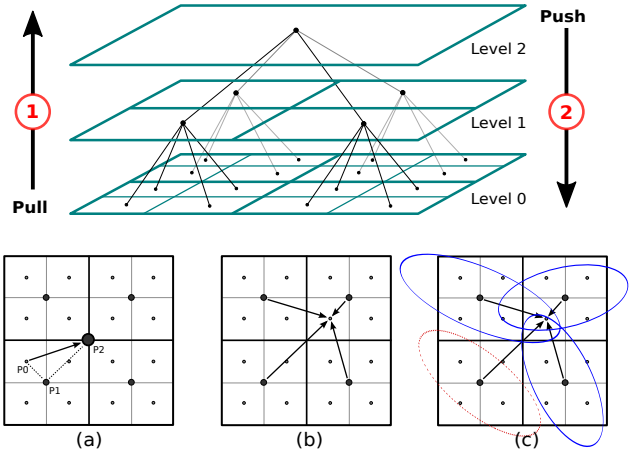


Fig. 4. Original algorithm proposed by Marroquim *et al.* [20]. Top: Image pyramid illustrating (1) the pull stage that gathers depth from high resolution up to low resolution images, and (2) the push stage that interpolates values down to the high resolution image. (a) Pull stage: Pixel values are associated with the above level from $P0$, to $P1$ and $P2$, and averaged; (b-c) Push stage: Averaged pixel values are pushed in the level below, and the ellipses associated with the projected points are used to limit their influence.

However, it produces flickering on object silhouettes when the viewpoint moves. This is mainly due to the screen space nature of the interpolation (if the point moves one pixel, it may be interpolated differently). Furthermore, when the camera moves closer to the object, splats belonging to the back surface are not occluded but interpolated between the front-most ones (Figure 5.g). This basically happens when the depth intervals of the front-most points intersect the depth intervals of the back points. Back splat attributes are also propagated up through

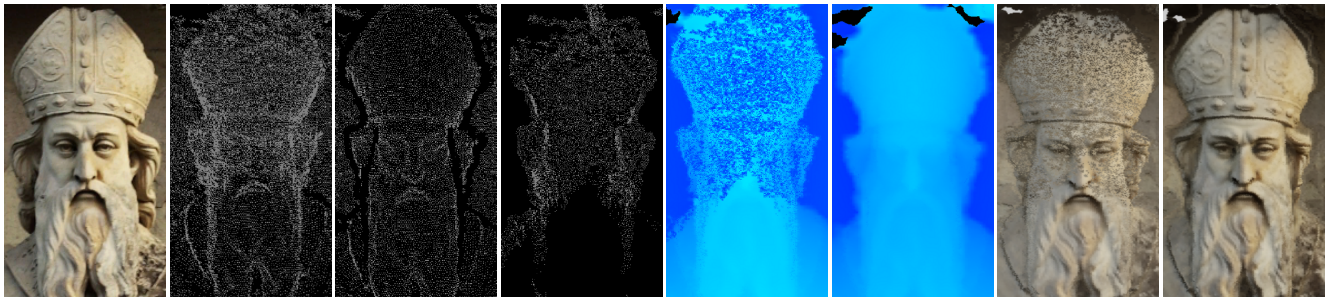


Fig. 5. Point splatting corrections: (a) Region selected on an original photograph; (b) Corresponding point cloud rendered with our system; (c) and (d) Separation of the whole point cloud into front and back points respectively using our first pass; (e) Depth reconstructed with Marroquim *et al.* [20]; (f) Depth reconstructed with our improvements; (g) and (h) final rendered images with our system using the depth maps (e) and (f) respectively;

the pyramid and interpolated with values of front splats during the push stage. This problem becomes critical when the front and back surfaces are close to each other. The improvements described by Marroquim *et al.* [40] reduce flickering on object silhouette, based on small 5x5 kernels used to distribute the splats along the pyramid hierarchy, in order to avoid unnecessary interpolations. However, they also notably slow-down the rendering process because of the tests required in the push phase, based on depth intervals to interpolate between projected points.

In this paper, we introduce a more simple method that both reduces flickering artifacts on the object silhouette and avoids undesirable depth interpolations. It is based on two additional *a priori* passes that manage silhouettes and provide a per-pixel depth approximation of the foreground. The general depth estimation method is performed according to the following passes:

1. Render the point cloud, using a point size corresponding to the projected ellipse in screen space, with the depth test enabled; Record the resulting depth map and binary mask defined by the resulting ellipses. Note that in that case, depth is constant for all ellipse pixels for a given point.
2. Classify projected points as *front-points* or *back-points*: Compare the depth of each projected point with the corresponding pixel depth produced in pass 1. A point is copied in a framebuffer called *BackgroundFB* if the difference is too large (and thus occluded by other splats). Otherwise, it is copied in a second framebuffer, called *ForegroundFB*.
3. Apply the pull-push algorithm separately on both *ForegroundFB* and *BackgroundFB*.
4. Use pass 1 to avoid the reconstruction of flickering pixels: If the pixel is filled in pass 1, the blending process is applied; Otherwise, it is discarded.
5. Merge the two resulting depth images.

Note that the background depth image is used for filling foreground holes, either for parts of the objects that could not be reconstructed, or for holes that may exist in the real objects.

During the first pass, each point is projected onto the screen plane using a point size that includes the correct corresponding ellipse. An image space square centered at the current vertex's projected position is rasterized. The point size is twice the projected radius r_{proj}, approximated similarly as that of Marroquim *et al.* [20]:

$$r_{proj} = r.\frac{f}{d_z}.h$$
$$f = \frac{1}{2.\tan(\frac{fov}{2})}$$

where r is the splat's ellipse radii, d_z is the distance from the eye (camera) to the center of the point, f is the focal length obtained from the camera view angle fov and h denotes the height (in pixels) of the viewport. For each pixel in the rasterized square, the fragment shader determines whether or not it belongs to the disk projected in the image plane (as an ellipse). The major and minor axis are aligned so that the length of the semi major axis a is the projected radius r_{proj} while the magnitude of the semi minor axis b corresponds to the length of the semi major axis scaled by the projected normal's z coordinate N_z. A pixel (x, y) is discarded when it does not belong to the corresponding ellipse: $\frac{d_x^2}{a^2} + \frac{d_y^2}{b^2} > 1$, where d_x and d_y correspond to the distance from the pixel (x, y) to the center of the square, rotated to ellipse coordinate system.

This process is employed for producing a depth map associated with any viewpoint during the interactive visualization process. It is also used for the original photographs, when they are loaded in the rendering process, in order to create the image depth map and prepare occlusion management.

Occlusion and photographs selection

A set of n photographs is defined by the associated cameras $\{C_k\}$. For each novel viewpoint (corresponding to a virtual camera C), the first step consists in producing the depth-map image based on the point cloud, according to the splatting process described above. The depth associated with each pixel $I(i, j)$ of C defines the 3D point P lying on the observed surface (Figure 3.b). P is back-projected onto the original photographs C_k, and the final pixel value of C is estimated thanks to a blending of their pixel values, provided that they are considered as valid in terms of depth (similar to the shadowmapping process [41]). Several conditions have to be met:

- The projection of P on a camera C_k should fall inside its field of view;
- The depth of P, projected onto C_k should be consistent with the associated pixel depth. Otherwise, P is considered as

occluded from camera C_k;

- The angle between a given viewing direction of P (denoted as $\mathbf{V_k}$ for camera C_k) and the normal vector $\mathbf{N_p}$ of P should be less than $\pi/2$, so as to avoid unreliable grazing angles: $\mathbf{N_p}.\mathbf{V_k} < \lambda$, with λ set to 10^{-4} in our implementation, after some experiments. $\mathbf{N_p}$ is estimated from the point cloud, during the pull-push process.

Blending pixels from photographs

Our epipolar consistency model corresponds to the previous ones [8, 6]. For a given pixel $I(i, j)$ in the new image defined by a camera C, the observed point P lying on the object surface is obtained thanks to the point cloud projection. P is also potentially observed on the photographs defined by cameras $\{C_k\}$. The goal is to blend the corresponding pixels from $\{C_k\}$ (practically in the fragment shader).

Camera orientation management may provide interesting details, notably with glossy effects, where the appearance may vary according to the viewing angle. Our approach makes use of all the cameras that can contribute to each pixel of C. This accounts for avoiding flickering effects with black pixels when the viewpoint changes, especially when only a few number of photographs are used. We use the angle θ_k between the view vector $\mathbf{V}$ from camera C and each original photograph view vector $\mathbf{V_k}$ for the 3D observed point P (Figure 3.c):

$$I(i,j) = \frac{1}{\sum_{k=1}^{N} < \cos\theta_k >} \sum_{k=1}^{N} I_k(u_k, v_k) < \cos\theta_k >, \quad (1)$$

where $I(i, j)$ is the current pixel value on C, corresponding to a 3D surface point P; $I_k(u_k, v_k)$ is the pixel value corresponding to P projected in image I_k and θ_k is the angle between the two viewing vectors $\mathbf{V_k}$ and $\mathbf{V}$:

$$< \cos\theta_k >= \max(0, \mathrm{dot}(\mathbf{V}, \mathbf{V_k})), \quad (2)$$

with $\mathbf{V_k} = \frac{C_k^c - P}{\|C_k^c - P\|}$, C_k^c being the center of projection of camera C_k.

The distance between the object and the camera is another important criterion, since some details may appear when the new viewpoint gets closer to the object. The goal is thus to associate a higher weight with original photographs that better match the distance at each new viewpoint will provide adaptively an adequate texture definition, with sharper details. In practice, the (R, G, B) value associated with a given pixel corresponds to the integral of reflected radiances on the object surface, toward the camera. Figure 3.d illustrates the observation of a same region $\mathcal{R}$ with two cameras C_1 and C_2 through two respective pixels $I_1(u_1, v_1)$ and $I_2(u_2, v_2)$. The solid angle corresponding to pixels do not cover perfectly the same surface on the object [8]. We propose to employ the following weighting function, based on the distance between the new viewpoint and the original photographs:

$$I(i,j) = \frac{1}{\sum_{k=1}^{N} \frac{1}{\delta(C,P,C_k)+1}} \sum_{k=1}^{N} \frac{I_k(u,v)}{\delta(C,P,C_k)+1}, \quad (3)$$

where $\delta(C, P, C_k)$ is the relative difference of distance between C and the new viewpoint C_k with respect to P:

$$\delta(C,P,C_k) = \mathrm{abs}(\|C^c - P\| - \|C_k^c - P\|), \quad (4)$$

where C^c and C_k^c are the respective centers of projection of C and C_k. Finally, integrating the model described above leads to the following equation:

$$I(i,j) = \frac{1}{\sum_{k=1}^{N} \frac{<cos\ \theta_k>}{\delta(C,P,C_k)+1}} \sum_{k=1}^{N} \frac{I_k(u,v) < cos\theta_k >}{\delta(C,P,C_k)+1}. \quad (5)$$

5. HDR Composition and Rendering

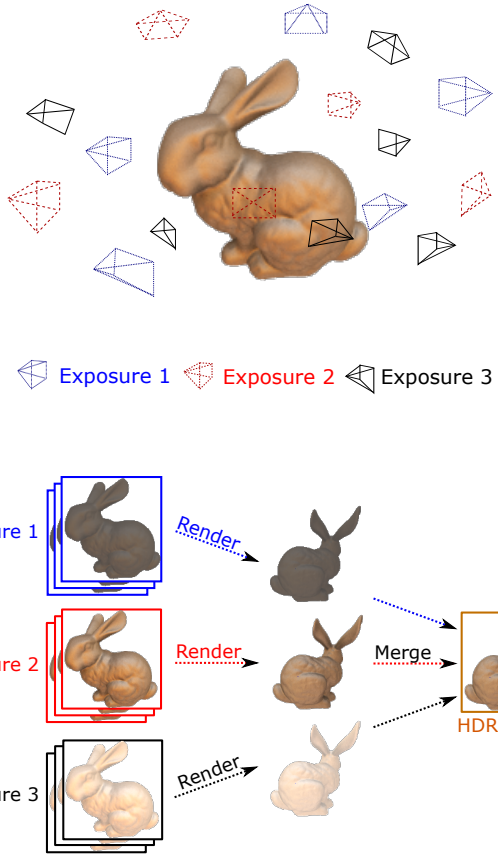


Fig. 6. HDR image reconstruction from three series of unstructured photographs. Each series is acquired with a fixed exposure. For a given fixed viewpoint, one image is reconstructed thanks to our rendering pipeline.

Our rendering system offers a method for producing an image from any specific viewpoint in space providing a transformation matrix. Thus, several perfectly aligned LDR images can be produced for each series of unstructured photographs (each with a fixed exposure, taken with standard hand-held cameras). The principle of our HDR reconstruction process is illustrated in Figure 6. Given an arbitrary viewpoint C, one image is rendered for each series of photographs independently, providing a set of LDR images aligned with C. These images are thus aligned and they can then be combined to produce an HDR image.

HDR reconstruction can be performed either on-the-fly during visualization, or as a pre-computation that reconstructs a unique series of HDR images as input of our rendering system. We have chosen the second option for two main reasons:

- The number of images increases according to the number of chosen exposition values, also increasing memory requirements on the GPU;
- When performed on the fly, the HDR reconstruction process reduces performance, thus impacting interactivity.

Ideally, each series of photograph should cover entirely the object surface, as well as the resulting HDR image series. We have chosen to pick arbitrarily the viewpoints associated with one of the LDR series denoted as S_{ref}. All the original viewpoints could be used as well for more precision in the visualization quality, there is no difference in the reconstruction process nor during visualization.

More precisely, let us consider M series of LDR images $\{S^1, S^2, ..., S^M\}$, each with a fixed exposure. Our goal is to reconstruct one series of K HDR images aligned with a set of K viewpoints $\{C^{new}_{i\in[1..K]}\}$. For each viewpoint C^{new}_i, M LDR images $\{I^1_i, I^2_i, ..., I^M_i\}$ are produced independently thanks to our visualization system. Note that the HDR image associated with C^{new}_i is reconstructed from LDR images computed from exactly the same viewpoint C^{new}_i. Our method does thus not suffer from misalignment issues. In addition, considering that the object surface is well covered by the set of photographs in each series, we have chosen the standard HDR process proposed by Debevec and Malik [31]. However, any other merging function could be employed as well. Each LDR view records a range of light power according to the chosen camera exposure. For a given pixel (sensor location (x, y)), the radiance reflected by the object is mapped according to a response function f, that combines the collected radiance E during an exposure time Δt, providing a value $I(x, y) = f(E(x, y).\Delta t)$. The HDR image corresponds to an estimation of radiance $E(x, y)$:

$$E(x, y) = \frac{f^{-1}(I(x, y))}{\Delta t}. \tag{6}$$

The goal is to determine the inverse of the response function f^{-1}. Considering the natural logarithm g of the invertible camera function, estimated thanks to a minimization process, the radiance map corresponding to the HDR image can be reconstructed with a combination of the M exposures:

$$\ln(E_{(x,y)}) = \frac{\sum_{i=1}^{M} w(I^i(x, y))[g(I^i(x, y)) - \ln(T^i)]}{\sum_{i=1}^{M} w(I^i(x, y))}, \tag{7}$$

w being a weighting function that controls the smoothness of g. In practice, the HDR image is reconstructed in an external program.

Finally, each viewpoint associated with the series $S_{\text{ref}}$is used producing an HDR image. Tone-mapping is performed using Reinhard's operator [42]. Let I^H be the input HDR image and N the total number of pixels; each RGB value is first converted to a luminance value L:

$$L = 0.212 * I^H_R + 0.715 * I^H_G + 0.072 * I^H_B. \tag{8}$$

The average luminance level L_a is estimated with a log-average:

$$L_a = \frac{1}{N} \exp(\sum_{(x,y)} \log(\epsilon + L(x, y))), \tag{9}$$

where ϵ is a fixed offset value used to avoid undefined log(0). Each pixel is then scaled using the average luminance value:

$$L_s(x, y) = \frac{k}{L_a} L(x, y), \tag{10}$$

k defines the overall spanning range of scaled luminance values. The pixels values are then scaled down to the range of [0,1], with the following operator:

$$L_d(x, y) = \frac{L_s(x, y)}{1 + L_s(x, y)}. \tag{11}$$

The final color of tone-mapped HDR image is given according to *gamma* and *exposure*:

$$I(x, y) = (\text{exposure} * L_d(x, y))^{1.0/\text{gamma}}. \tag{12}$$

Since L_a has to be estimated independently for each new viewpoint, flickering may alter the visualization process, and this computation would require an additional pass. In practice, we did not notice any flickering effect due to this method. However if such variations were visible, a fixed value of L_a could be estimated *a priori* from all the HDR images of S_{ref}.

6. Implementation and results

The above rendering system presented in the previous sections has been implemented in WebGL 1.0. All the results presented in this paper have been produced with an NVIDIA GeForce 750 GTX. The original photographs resolution vary from 1500×1000 to 5600×3700 pixels and the storage on the GPU uses texture compression. This configuration demonstrates the efficiency and versatility of our method, even with a moderately powerful graphics processor. In practice 8 passes are required to render a novel view. The first three passes construct the object silhouette binary mask, depth map, and classify the points as front-points or back-points (Section 4). Three passes implement the pull-push algorithm that produces the depth map associated with a viewpoint. Two more passes are required for the implementation of the epipolar consistency model (blending scheme). This latter provides auto-adaptive textures and reproduces glossy effects during navigation. Rendering can be performed with series of either LDR or HDR images.

HDR images are managed thanks to a *DDS* format. Each of them is firstly tone-mapped using Reinhard's operator: The resulting LDR image is accompanied by the ratio obtained by dividing the HDR original luminance pixel values by the tone mapped luminance. The tone mapped image is stored in a *PNG* file where the ratio information is log-encoded and stored in the alpha channel as in [27]. The resulting PNG image is then compressed using DXT5 codec, widely supported on graphics hardware and stored in a *DDS* file. Decoding and restitution of the HDR image is performed in the fragment shader using Equation 12.

This approach has been applied to a variety of datasets, with LDR and HDR images. Figure 7 illustrates the test scenes used in this paper. Table 1 presents their characteristics, including the number of series of images, the total number of images in case of HDR reconstruction, the number of images actually used during visualization, and the number of 3D points generated by the SFM pre-process.

Fig. 7. Example scenes displayed with our interactive image-based rendering system.

Scene	# series	# images		# 3D
		HDR	Visu	points
Church	1	-	20	2'802'150
Cathedral	1	-	33	2'900'986
Mirebeau	1	-	22	2'172'103
Bell	1	-	97	510'238
Statue	4	160	58	3'819'846
Seashell	3	44	20	222'228
Horse	3	55	23	117'891

Table 1. Scenes' characteristics.

Figure 8 illustrates comparisons between our method and a conventional point-based rendering system with colored vertices. Our rendering system benefits from both worlds: The point cloud projection is fast with the rasterization process and the image quality is also high since the texture is not fixed a priori onto the 3D geometry; The original photographs are adaptively blended depending on the observer viewpoint.

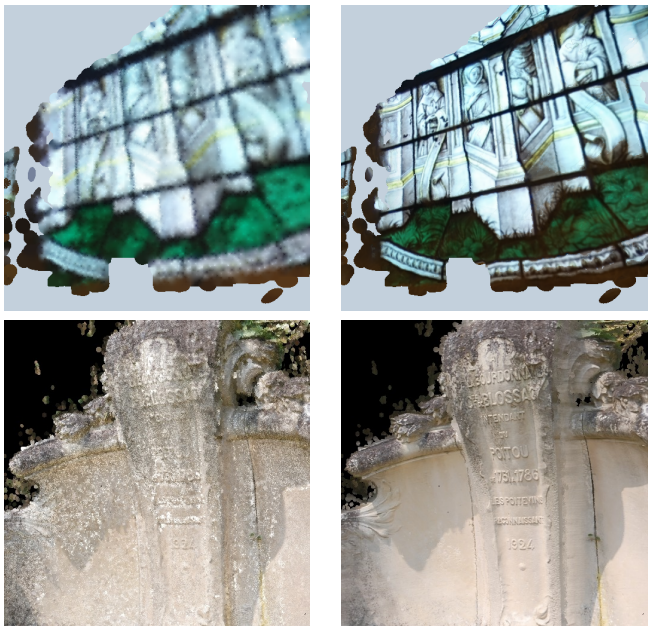


Fig. 8. Point-based rendering with one color per point (left); our image-based rendering approach (right).

Figure 9 shows examples of view-dependent appearance changes with a glossy bell. When the user moves around the object, the highlights move accordingly. This is ensured by the blending model that handles observation orientations.

Scene	Initialization (ms)	Visualization (ms)	Calibrated FPS
Church	100	15	66
Cathedral	167	17	58
Mirebeau	91	13	76
Bell	100.5	8	111
Statue	335	22	45
Seashell	28	6	166
Horse	17	5	200

Table 2. Running time for 512×512 image resolution.

When several series of LDR images are provided with varying exposures, the user may choose one or several series as reference viewpoints. For each of them, LDR views are rendered for each exposure, as described in Section 5. The resulting images are transferred on the CPU and merged as an HDR image, saved in *DDR* format using HDR tools [43, 44], as explained above.

For each original photograph, the depth map corresponding to each camera position is rendered from the point cloud in a separate framebuffer, when the point cloud and photographs are loaded (as well as their corresponding projection matrices). Table 2 provides running time for each task, as well as the frame rate obtained during visualization, with an image resolution fixed to 512×512 pixels. Column *Initialization* indicates the point cloud projection time onto original views and depth reconstruction; Column *visualization* corresponds to the point

Fig. 9. View-dependent appearance: Highlights move according to the observer. Top: Two regions on the object are encircled, the highlight visible on the yellow and blue regions disappears on the right image (the text on the top of the bell can be used as reference). Bottom: The highlight in the yellow region moves on the right according to the viewpoint.

Fig. 10. Comparison between two strategies for HDR rendering. Left: The HDR image is reconstructed by computing a separate rendering pass for each input exposure and tone-mapping on the fly, resulting in visually sharper image details. Right: The HDR image is reconstructed after a precomputation of a unique series of viewpoints, leading to a slightly lower image quality.

cloud projection time for a novel viewpoint, including point splatting, photograph pixels back projection and blending. Performances essentially depend on the number of 3D points used as geometric proxy, and on the number of input images. Depth reconstruction time remains constant for a fixed image resolution since it only depends on the point cloud and image resolution.

Since the framerate is capped at 60 fps with WebGL, we have employed profiling tips [45]. The following performance information provides an idea about Javascript timings, and our rendering system is in practice faster.

Figure 10 shows the difference in terms of quality between two approaches for reconstructing HDR images. The first one reconstructs the HDR images on-the-fly during the rendering process; It better preserves sharp details, but it is more time consuming and it requires additional GPU memory because all the original photographs have to be stored. The second one favors performance and memory requirements, but the resulting images are slightly blurred due to the two sets of projections (one for generating the HDR dataset and the other for generating the novel view). The running time of the first approach corresponds to the running time of the second approach multiplied by the number of series to which we add the tone-mapping time which can not be neglected. The second approach is a trade-off between quality, performance and memory consumption.

Figure 11 illustrates various tone mappings for our HDR rendering system, for the three scenes acquired with a hand-held camera, composed of 3 or 4 independent series of photographs.

Limitations

The current implementation of our method requires to store all the images on the GPU, potentially with high memory consumption (depending on the number of images and their resolution). Reducing photographs resolution may be a solution in some cases (Figure 12), but the resulting image quality is also reduced visually. Loading photographs on demand according to the viewpoint would be interesting, but the chosen strategy should handle performance drops due to bandwidth limitations.

Figure 13 illustrates a comparison between our point-based rendering approach and the use of a mesh-based proxy [8] constructed from a dense point cloud, using the method proposed by Kazhdan *et al.* [46]. Ghosting artifacts are more visible due to incorrect geometry reconstruction, since edges may be added at undesired surface regions, resulting in erroneous depth maps, even with very dense meshes.

Another limitation comes from the flickering artifacts still due to *z-fighting* on object silhouettes, produced by the pull-push approximation during splatting. The improvement we propose only handles the flickering observed on the background and also improves the object silhouettes, but it does not account for subparts of objects that project onto the point cloud itself. Splats segmentation as proposed in VDTS [21] could be an interesting solution if employed on real photographs, but automation is not immediate for complex configurations.

Finally, our method is interesting with hand-held acquisitions and relatively few photographs. However, as stated by Davis et al [24], when the number of images increases, the process requires to backproject each observed point onto all the original cameras, which results in computation overheads.

7. Conclusion and Future Work

This paper presents an image-based rendering method that allows interactive free viewpoint navigation within environments acquired from hand-held cameras, while allowing an automatic and straightforward reconstruction of HDR images. It is based

Fig. 11. Interactive HDR rendering with exposure control, with scenes captured from 3 or 4 different series of photographs. For each line: An original photograph and 3 different exposures rendered with our system. *Statue* (first line), *Seashell* (second line); *Horse* (third line).

on a point-based proxy, avoiding mesh reconstruction that often generates undesirable geometry. Point-based geometry is also faster to render on GPU.

We propose corrections to reduce flickering and better manage depth artifacts that appear on the previous methods. Our enhancement corresponds to an additional pass of splatting, that produces a binary mask for guiding silhouette management.

We have also shown that our system can be employed for constructing and rendering HDR images from several series of photographs, each of them having a fixed exposure. It does not require any user intervention during the reconstruction process, and only few tens of unstructured photographs are sufficient for providing interactive and realistic visualization.

The presented results have been purposely produced with a WebGL implementation, using a standard laptop computer. Visualization is interactive with good performance, even with dense point clouds composed of several millions of primitives.

In the future, we aim at using HDR images produced by our process in association with an HDR acquisition of the lighting environment, so as to estimate reflectance information and change the lighting conditions for the acquired objects. Such a study would require to integrate the incoming radiance for each position on the object surface; the problem remains ill-posed and an a priori analysis of the lighting environment could be necessary to identify the most important factors.

References

[1] Berger, M, Tagliasacchi, A, Seversky, LM, Alliez, P, Levine, JA, Sharf, A, et al. State of the Art in Surface Reconstruction from Point Clouds. In: Lefebvre, S, Spagnuolo, M, editors. Eurographics 2014 - State of the Art Reports. 2014,.

[2] McMillan, L, Bishop, G. Plenoptic modeling: An image-based rendering system. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. ACM; 1995, p. 39–46.

[3] Levoy, M, Hanrahan, P. Light field rendering. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. ACM; 1996, p. 31–42.

[4] Debevec, PE, Taylor, CJ, Malik, J. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. ACM; 1996, p. 11–20.

[5] Vangorp, P, Chaurasia, G, Laffont, PY, Fleming, RW, Drettakis, G. Perception of visual artifacts in image-based rendering of façades. In: Computer Graphics Forum; vol. 30. Wiley Online Library; 2011, p. 1241–1250.

[6] Hedman, P, Ritschel, T, Drettakis, G, Brostow, G. Scalable inside-out image-based rendering. ACM Transactions on Graphics (TOG) 2016;35(6):231.

[7] Gortler, SJ, Grzeszczuk, R, Szeliski, R, Cohen, MF. The lumigraph. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. ACM; 1996, p. 43–54.

[8] Buehler, C, Bosse, M, McMillan, L, Gortler, S, Cohen, M. Unstructured lumigraph rendering. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM; 2001, p. 425–432.

[9] Eisemann, M, De Decker, B, Magnor, M, Bekaert, P, De Aguiar, E, Ahmed, N, et al. Floating textures. In: Computer graphics forum; vol. 27. Wiley Online Library; 2008, p. 409–418.

[10] Snavely, N, Seitz, SM, Szeliski, R. Photo tourism: Exploring photo collections in 3d. In: SIGGRAPH Conference Proceedings. New York, NY, USA: ACM Press. ISBN 1-59593-364-6; 2006, p. 835–846.

[11] Snavely, N, Garg, R, Seitz, SM, Szeliski, R. Finding paths through the world's photos. ACM Transactions on Graphics (SIGGRAPH 2008) 2008;27(3):11–21.

[12] Moulon, P, Monasse, P, Marlet, R. Global fusion of relative motions for robust, accurate and scalable structure from motion. In: Proceedings of the IEEE International Conference on Computer Vision. 2013, p. 3248–3255.

[13] Goesele, M, Snavely, N, Curless, B, Hoppe, H, Seitz, SM. Multi-view stereo for community photo collections. In: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on. IEEE; 2007, p. 1–8.

[14] Furukawa, Y, Ponce, J. Accurate, dense, and robust multiview stere-

(a) 5616×3744 pixels (original), 456 Mo (b) 1/4 size, 297 Mo (c) 1/16 size, 257 Mo

Fig. 12. Church visualization with various resolutions of input photographs, with corresponding memory usage (the given size corresponds to the total requirement, including geometry).

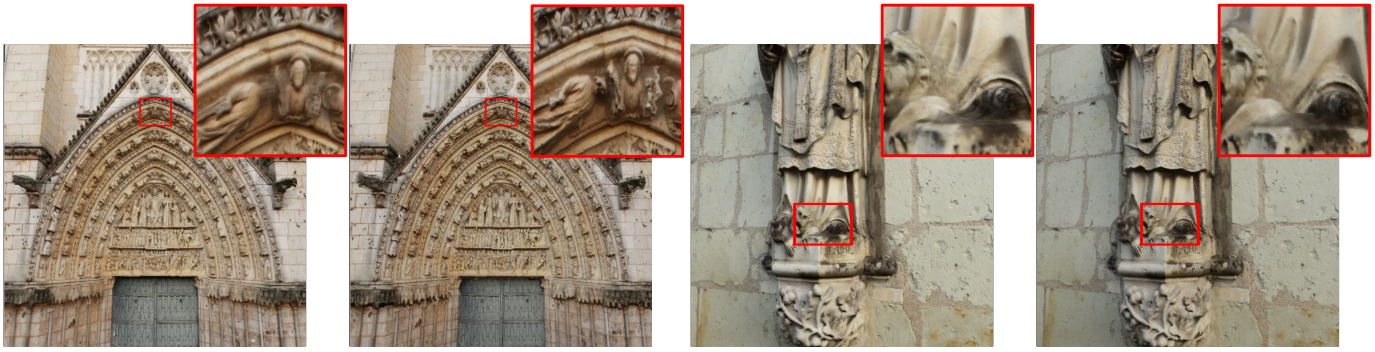


Fig. 13. Comparisons between our approach and mesh-based proxies for scenes Cathedral (left, 1.875 million triangles) and Mirebeau (right, 2.174 million triangles). For each scene, the subimages illustrate our rendering method (top-left), the mesh-based method (top-right), the depth difference between point-based rendering and mesh-based rendering (bottom-left), and the rendered mesh (bottom-right).

opsis. IEEE transactions on pattern analysis and machine intelligence 2010;32(8):1362–1376.

[15] Kazhdan, M, Bolitho, M, Hoppe, H. Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. Eurographics Association; 2006, p. 61–70.

[16] Fuhrmann, S, Goesele, M. Floating scale surface reconstruction. ACM Transactions on Graphics (TOG) 2014;33(4):46.

[17] Chaurasia, G, Sorkine, O, Drettakis, G. Silhouette-aware warping for image-based rendering. In: Computer Graphics Forum; vol. 30. Wiley Online Library; 2011, p. 1223–1232.

[18] Chaurasia, G, Duchene, S, Sorkine-Hornung, O, Drettakis, G. Depth synthesis and local warps for plausible image-based navigation. ACM Transactions on Graphics (TOG) 2013;32(3):30.

[19] Ortiz-Cayon, R, Djelouah, A, Drettakis, G. A bayesian approach for selective image-based rendering using superpixels. In: International Conference on 3D Vision-3DV. 2015,.

[20] Marroquim, R, Kraus, M, Cavalcanti, PR. Efficient point-based rendering using image reconstruction. In: Proceedings Symposium on Point-Based Graphics. 2007, p. 101–108.

[21] Yang, R, Guinnip, D, Wang, L. View-dependent textured splatting. The Visual Computer 2006;22(7):456–467.

[22] Pujades, S, Devernay, F, Goldluecke, B. Bayesian view synthesis and image-based rendering principles. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014, p. 3906–3913.

[23] Nieto, G, Devernay, F, Crowley, J. Variational image-based rendering with gradient constraints. In: 3D Imaging (IC3D), 2016 International Conference on. IEEE; 2016, p. 1–8.

[24] Davis, A, Levoy, M, Durand, F. Unstructured light fields. In: Computer Graphics Forum; vol. 31. Wiley Online Library; 2012, p. 305–314.

[25] Penner, E, Zhang, L. Soft 3d reconstruction for view synthesis. ACM Transactions on Graphics (TOG) 2017;36(6):235.

[26] Song, L, Li, X, Yang, Yg, Zhu, X, Guo, Q, Liu, H. Structured-light based 3d reconstruction system for cultural relic packaging. Sensors 2018;18(9):2981.

[27] Wang, J, Zhang, J, Xu, Q. Research on 3d laser scanning technology based on point cloud data acquisition. ICALIP 2014 - 2014 International Conference on Audio, Language and Image Processing, Proceedings 2015;:631–634.

[28] Thonat, T, Djelouah, A, Durand, F, Drettakis, G. Thin structures in image based rendering. Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 2018;37(4):12.

[29] Hedman, P, Philip, J, Price, T, Frahm, JM, Drettakis, G, Brostow, G. Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (SIGGRAPH Asia Conference Proceedings) 2018;37(6).

[30] Sibbing, D, Sattler, T, Leibe, B, Kobbelt, L. Sift-realistic rendering. In: Proceedings of the 2013 International Conference on 3D Vision. 3DV '13; 2013, p. 56–63.

[31] Debevec, PE, Malik, J. Recovering high dynamic range radiance maps from photographs. In: Annual Conference on Computer Graphics. 1997, p. 369–378.

[32] Reinhard, E, Ward, G, Pattanaik, S, Debevec, P. High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (The Morgan Kaufmann Series in Computer Graphics). San Francisco, CA, USA: Mor-

gan Kaufmann Publishers Inc.; 2005. ISBN 0125852630.
[33] Abebe, MA, Booth, A, Kervec, J, Pouli, T, Larabi, MC. Towards an automatic correction of over-exposure in photographs. Comput Vis Image Underst 2018;168(C):3–20.
[34] Tomaszewska, A, Mantiuk, R. Image registration for multi-exposure high dynamic range image acquisition. In: WSCG. 2007,.
[35] Zimmer, H, Bruhn, A, Weickert, J. Freehand hdr imaging of moving scenes with simultaneous resolution enhancement. In: Computer Graphics Forum; vol. 30. Wiley Online Library; 2011, p. 405–414.
[36] Sen, P, Kalantari, NK, Yaesoubi, M, Darabi, S, Goldman, DB, Shechtman, E. Robust Patch-Based HDR Reconstruction of Dynamic Scenes. ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH Asia 2012) 2012;31(6):203:1–203:11.
[37] Moulon, P, Monasse, P, Perrot, R, Marlet, R. Openmvg: Open multiple view geometry. In: International Workshop on Reproducible Research in Pattern Recognition. Springer; 2016, p. 60–74.
[38] Moulon, P, Monasse, P, Marlet, R. Adaptive structure from motion with a contrario model estimation. In: Asian Conference on Computer Vision. Springer; 2012, p. 257–270.
[39] Rusu, RB, Cousins, S. 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA). Shanghai, China; 2011,.
[40] Marroquim, R, Oliveira, A, Cavalcanti, PR. High quality image reconstruction of point models. In: Computer Graphics and Image Processing, 2008. SIBGRAPI'08. XXI Brazilian Symposium on. IEEE; 2008, p. 297–304.
[41] Williams, L. Casting curved shadows on curved surfaces. In: ACM Siggraph Computer Graphics; vol. 12. ACM; 1978, p. 270–274.
[42] Reinhard, E, Stark, M, Shirley, P, Ferwerda, J. Photographic tone reproduction for digital images. ACM transactions on graphics (TOG) 2002;21(3):267–276.
[43] Cotter, A. Hdr tools. 2011. URL: `http://ttic.uchicago.edu/~cotter/projects/hdr_tools/`.
[44] Banterle, F. Spidergl. 2013. URL: `http://spidergl.sourceforge.net`.
[45] Cozzi, P. Webgl profiling tips. 2014. URL: `https://cesium.com/blog/2014/12/01/webgl-profiling-tips/`.
[46] Kazhdan, M, Hoppe, H. Screened poisson surface reconstruction. ACM Transactions on Graphics (ToG) 2013;32(3):29.