From the Digitization of Cultural Artifacts to the Web Publishing of Digital 3D Collections: an Automatic Pipeline for Knowledge Sharing

Frédéric Larue, Marco Di Benedetto, Matteo Dellepiane and Roberto Scopigno
ISTI-CNR, Pisa, Italy

Abstract— **In this paper, we introduce a novel approach intended to simplify the production of multimedia content from real objects for the purpose of knowledge sharing, which is particularly appropriate to the cultural heritage field. It consists in a pipeline that covers all steps from the digitization of the objects up to the Web publishing of the resulting digital copies. During a first stage, the digitization is performed by a high speed 3D scanner that recovers the object's geometry. A second stage then extracts from the recovered data a color texture as well as a texture of details, in order to enrich the acquired geometry in a more realistic way. Finally, a third stage converts these data so that they are compatible with the recent WebGL paradigm, then providing 3D multimedia content directly exploitable by end-users by means of standard Internet browsers.**
The pipeline design is centered on automation and speed, so that it can be used by non expert users to produce multimedia content from potentially large object's collections, like it may be the case in cultural heritage. The choice of a high speed scanner is particularly adapted for such a design, since this kind of devices has the advantage of being fast and intuitive. Processing stages that follow the digitization are both completely automatic and "seamless", in the sense that it is not incumbent upon the user to perform tasks manually, nor to use external softwares that generally need additional operations to solve compatibility issues.

I. Introduction

In the field of cultural heritage (CH), knowledge sharing is one of the most essential aspects for communication activities between museal institutions, that conserve and take care of cultural collections, and the public. Among other things, these activities include education, research and study as well as entertainment. All of them are really precious for the spread of culture. However, the public is not the only one to benefit from knowledge sharing: it is important for promotion and advertisement purposes as well, which are both of a high interest for the museal institutions themselves regarding visibility, development and long term sustainability.

In order to preserve the integrity of cultural goods, knowledge sharing generally makes use of surrogates so as to avoid to expose directly the real artifacts to potential risks of deterioration. For this purpose, multimedia technologies are becoming more and more widespread in the CH field, where these surrogates are then represented by digital copies. This popularity can be explained by at least two reasons. On the one hand, computing tools clearly provide an underlying easiness for data storage, indexation, browsing and sharing, due to the existing network facilities and to the new Web technologies. On the other hand, recent advances in 3D scanning give the possibility to create multimedia content from real artifacts, producing faithful digital imprints and avoiding the tedious and time consuming task of a manual modeling through CAD softwares.

Particularly, new high speed systems like in-hand scanners present big advantages for CH. Firstly, they are able to acquire digital copies in a few minutes, which is really important when the multimedia content must be produced from huge collections in reasonable times, or when several fragments must be scanned in order to plan the restoration of destroyed pieces. Moreover, they can be manipulated by non-expert users as well, since they provide an interactive feedback and rely on the temporal coherency of the high-speed acquisition to get rid of the traditional alignment problems that generally need to be solved manually during a tedious post-processing phase.

Despite the availability of these technologies and their increasing popularity, there is still a lack of global and automatic solutions enabling to cover the whole processing chain that ranges from content creation to content publishing.

The first weak point of this chain occurs during the acquisition itself: for CH applications, a good representation of the geometry is not sufficient to produce faithful surrogates, since interactive visualization requires to be able to provide synthetic images as near as possible to the real appearance of the depicted object. In that case, the geometry needs to be paired with an accurate representation of the surface appearance (color, small shape details, reflection characteristics). Unfortunately, commercial scanning systems mostly focused on shape measurement, putting aside until recently the problem of recovering quality textures from real objects. This has led to a lack of efficient and automatic processing tools for color acquisition and reconstruction.

The second problem is that both tasks of creation and publishing of multimedia content are generally totally uncorrelated from each other. From a practical point of view, it means that a different software must be used for each of them. A work for converting the various inputs/outputs in a compatible way is then necessary, and generally consists in a manual task that is incumbent upon the user.

doi:10.4304/jmm.7.2.132-144

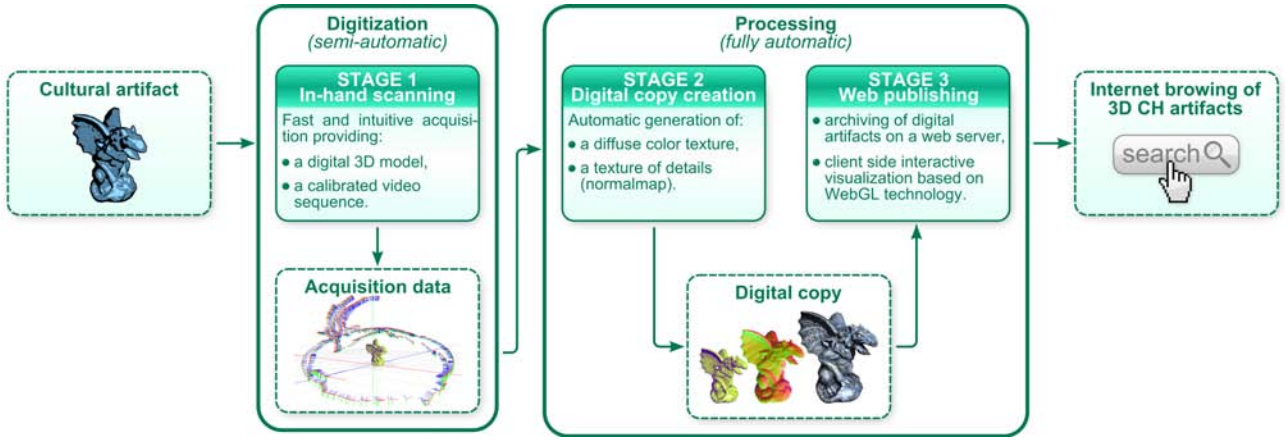

Figure 1. Overview of the presented framework, that covers the whole chain from the 3D digitization of real CH artifacts up to the Web publishing of the resulting digital 3D copies for archiving, browsing and visualization through Internet.

In this paper, we present a complete system that enables to create colored 3D digital copies from existing artifacts and to publish them directly on Internet, through an interactive visualization based on WebGL technology. This system, outlined on Figure 1, consists in three stages.

During the first one, acquisition is performed directly by the user in an intuitive manner thanks to an in-hand digitization device performing 3D scanning in real-time. The data provided by the scanner, as well as some properties specific to this kind of devices, are then exploited to automatically produce a diffuse color texture for the 3D model. This texture is deprived of the traditional visual artifacts that may appear due to the presence in the input pictures of shadows, specular highlights, lighting inconsistencies or calibration inaccuracies. Moreover, inasmuch as high speed scanning systems are often prone to a lack of accuracy with respect to more classic digitization technologies, we also estimate a normal texture from these data, once again in an automatic manner. This texture captures the finest geometric details that may be missed during the 3D acquisition, and can then be used afterwards to enrich the original geometry during visualization.

Once geometry and texture information have been processed, the third and last stage of the production pipeline performs an optimization phase aimed at producing a compact and Web-friendly version of the data. The output of this stage will be used for real-time visualization on commodity platforms. One of our main goals is the archival and deployment of digital copies using standard, well-settled and widely accessible technologies: in this view, we use a standard Web server as our data provider, and the WebGL technology to visualize and integrate the digital copy on standard Web pages.

The contributions proposed in this paper can be summarized as follows:

- a complete and almost fully automatic pipeline for the production of 3D multimedia content for Internet applications, covering a chain ranging from the digitization of real artifacts to the Web publishing of the produced digital copies;
- a texturing method specifically designed for real-time scanning systems, that accounts for specific properties of this kind of devices in order to improve the acquired 3D model with a good quality color texture without cracks nor illumination related visual artifacts, as well as a normal texture capturing the finest geometric features;
- the coupling of intuitive acquisition techniques with the recent paradigms proposed by WebGL technology for Web publishing. Hence, the archival and the sharing of vast item collections becomes possible and easy also for non expert users.

The remainder of this paper is organized as follows. Section II reviews the related work on software approaches or complete systems for color acquisition, texture reconstruction and real-time visualization on the Web platform. Section III presents the two first stages of our system, namely the in-hand scanner used for the acquisition, as well as our processing step for generating a digital copy from the acquired data. The third and last stage dedicated to the preparation of the digital copy for Web publishing is then presented in section IV. Finally, section V shows the results achieved and section VI draws the conclusions.

II. Related work

A. Real-time 3D scanning

An overview of the 3D scanning and stereo reconstruction goes well beyond the scope of this paper. We will mainly focus on systems for real-time, in-hand acquisition of geometry and/or color. Their main issues are the availability of technology and the problem of aligning data in a very fast way.

Concerning the first point, 3D acquisition can be based on stereo techniques or on active optical scanning solutions. Among the latter, the most robust approach is based on the use of fast structured-light scanners [1], where a high speed camera and a projector are used to recover the range maps in real-time. The alignment problem is usually solved with smart implementations of the ICP algorithm [2], [3], where the most difficult aspect to solve

is related to the loop closure during registration.
In the last few years, some in-hand scanning solutions have been proposed [2], [4], [5]: they essentially differ on the way projection patterns are handled, and in the implementation of ICP. None of the proposed systems takes into account the acquisition of color, although the one proposed by Weise et al. [5] contains also a color camera (see next section for a detailed description). This is essentially due to the low resolution of the cameras, and to the difficulty of handling the peculiar illumination provided by the projector. Other systems have been proposed which take into account also the color, but they are not able to achieve real-time performances [6] or to reconstruct the geometry in an accurate way [7].

B. Color acquisition and visualization on 3D models

Adding color information to an acquired 3D model is a complex task. The most flexible approach starts from a set of images acquired either in a second stage with respect to the geometry acquisition, or simultaneously but using different devices. Image-to-geometry registration, which can be solved by automatic [8]–[10] or semi-automatic [11] approaches, is then necessary. In our case, this registration step is not required, because the in-hand scanning system provides images which are already aligned to the 3D model.
Once alignment is performed, it is necessary to extract information about the surface material appearance and transfer it on the geometry. The most correct way to represent the material properties of an object is to describe them through a reflection function (e.g. BRDF), which attempts to model the observed scattering behavior of a class of real surfaces. A detailed presentation of its theory and applications can be found in Dorsey [12]. Unfortunately, state-of-the-art BRDF acquisition approaches rely on complex and controlled illumination setups, making them difficult to apply in more general cases, or when fast or unconstrained acquisition is needed.
A less accurate but more robust solution is the direct use of images to transfer the color to the 3D model. In this case, the apparent color value is mapped onto the digital object's surface by applying an inverse projection. In addition to other important issues, there are numerous difficulties in selecting the correct color when multiple candidates come from different images.
To solve these problems, a first group of methods selects, for each surface part, a portion of a representative image following a specific criterion − in most cases, the orthogonality between the surface and the view direction [13], [14]. However, due to the lack of consistency from one image to another, artifacts are visible at the junctions between surface areas receiving color from different images. They can be partially removed by working on these junctions [13]–[15].
Another group of methods "blends" all image contributions by assigning a weight to each one or to each input pixel, and by selecting the final surface color as the weighted average of the input data, as in Pulli et al. [16]. The weight is usually a combination of various quality metrics [17]–[19]. In particular, Callieri et al. [20] presented a flexible weighting system that can be extended in order to accommodate additional criteria. These methods provide better visual results and their implementation permits very complex datasets to be used, i.e. hundreds of images and very dense 3D models. Nevertheless, undesirable ghosting effects may be produced when the starting set of calibrated images is not perfectly aligned. This problem can be solved, for example, by applying a local warping using optical flow [21], [22].
Another issue, which is common to all the cited methods, is the projection of lighting artifacts on the model, i.e. shadows, highlights, and peculiar BRDF effects, since the lighting environment is usually not known in advance. In order to correct (or to avoid to project) lighting artifacts, two possible approaches include the estimation of the lighting environment [23] or the use of easily controllable lighting setups [24].

C. 3D graphics on the Web platform

Since the birth of Internet, content of Web document has been characterized by several types of media, ranging from plain text to images, audio or video streams. When personal computers have started being equipped with fast enough graphics acceleration hardware, 3D content began in its turn to have an important role in the multimedia sphere. The first tools aimed at visualizing 3D models in Web pages were based on embedded software components, such as Java applets or ActiveX controls [25]. Several proprietary plug-ins and extensions for Web browsers were developed, giving evidence at the lack of standardization for this new content type. Beside the developers fragmentation that arises due to this wide variety of available tools and to their incompatibilities, the burden incumbent upon the user for the installation of additional software components prevented a wide adoption of online 3D content.
Steps toward a standardization have been taken with the introduction of the Virtual Reality Modeling Language (VRML) [26] in 1995 and X3D [27] in 2007. However, even though they have been well-accepted by the community, the 3D scene visualization was still delegated to external software components.
The fundamental change happened in 2009 with the introduction of the WebGL standard [28], promoted by the Khronos Group [29]. With minor restrictions related to security issues, the WebGL API is a one-to-one mapping of the OpenGL|ES 2.0 specifications [30] in JavaScript. This implies that modern Web browsers, like Google Chrome or Mozilla Firefox, are able to *natively* access the graphics hardware without needing additional plug-ins or extensions. WebGL being a low-level API, a series of higher-level libraries have been developed on top of it. They differ from each other by the programming paradigm they use, ranging from scene-graph-based interfaces, like Scene.js [31] and GLGE [32], to procedural paradigms, like SpiderGL [33] and WebGLU [34].

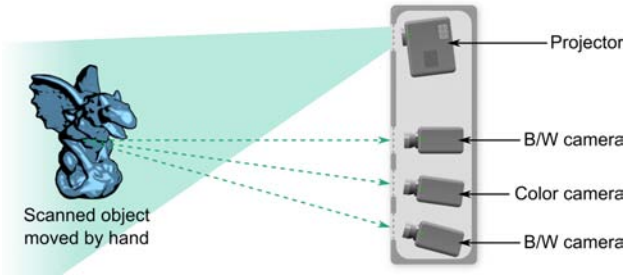

Figure 2. The in-hand scanning device used during the first step of the presented workflow, producing the data flow required for the generation of cultural artifacts digital copies.

In our pipeline, as it will be shown, we use SpiderGL as the rendering library for the real-time visualization of the acquired digital copies.

III. Digitization and Processing of CH Artifacts for Generating Digital Copies

Cultural heritage has been a privileged field of application for 3D scanning since the beginning of its evolution. This is due to the enormous variety and variability of the types of objects that can be acquired. Moreover, archival and preservation are extremely important issues as well. Although 3D scanning can be considered now a "mature" technology, the acquisition of a large number of objects can be expensive both in terms of hardware and time needed for data processing. Very good results can be achieved by customizing solutions for collections where objects are almost of the same size and material, but this can be expensive [35] or hard to extend to generic cases [36]. Although some low cost and/or hand-held devices are available, they usually need the placement of markers on the object, which is something that is hard to make on CH artifacts. Conversely, the presented method uses only an affordable scanning system and does not make any particular assumption on the measured objects (except the fact that they are manipulable by hand), neither for the scanning session itself nor for the post-processing steps.
This section describes the two first stages of our workflow: how and with which technology real artifacts can be easily digitized by the user (section III-A) and how the resulting data are exploited to recover automatically a color texture (section III-B) and a texture of details (section III-C) to enrich the 3D model provided by the acquisition.

A. Acquisition by in-hand scanning

The first stage of our workflow, namely the one producing all data required for the creation of digital copies from cultural artifacts, is based on an in-hand scanner whose hardware configuration is shown in Figure 2. This scanner, like most of the high speed digitization systems, is based on structured light. Shape measurement is performed by phase-shifting, using three different sinusoidal patterns to establish correspondences (and then to perform optical triangulation) between the projector and the two black and white video cameras. The phase unwrapping, namely how the different signal periods are demodulated, is achieved by a GPU stereo matching between both cameras (see [3], [5] for more details). The whole process produces one range map in $14ms$. Simultaneously, a color video flow is captured by the third camera. During an acquisition, the only light source in the scene is the scanner projector itself, for which the position is always perfectly known.
The scanning can be performed in two different ways. If the object color is neither red nor brown, it can be done by holding the object directly by hand. In this case, occlusions are detected by a hue analysis which produces, for each video frame, a map of skin presence probability. Otherwise, a black glove must be used. Although much less convenient for the scanning itself, it makes the occlusion detection trivial by simply ignoring dark regions in the input pictures.
Each scanning session then produces a 3D mesh and a color video flow. For each frame of this video, the viewpoint and the position of the light (*ie.* the scanner projector) are given, as well as the skin probability map in the case of a digitization performed by hand. These data are then used in the next stage of the workflow in order to produce the color texture and the texture of details, as explained hereafter, in section III-B.
Even if this stage requires the intervention of the user, the choice of a real-time scanner to perform the digitization is particularly appropriate for non expert operators. Indeed, for reasons already discussed in the introduction, its usage is particularly easy and intuitive, and does not require technical knowledge or manual post-processing. Finally, it must be noticed that our method does not work only with the presented scanner, but can be implemented for any high speed digitization device that is based on the same principle.

B. Recovery of a diffuse color texture

Our texturing method extends the work proposed in [20] so as to adapt it to the data flow produced by the scanning system presented above. The idea, summarized in Figure 3, is to weight each input picture by a mask (typically a gray scale image) which represents a per-pixel confidence value. The final color of a given surface point is then computed as the weighted average of all color contributions coming from the pictures into which this point is visible. Masks are built by the composition of multiple elementary masks, which are themselves computed by image processing applied either on the input image or on a rendering of the mesh performed from the same viewpoint.
In the original paper, three criteria related to viewing conditions have been considered for the mask computation: the distance to the camera, the orientation with respect to the viewpoint, and the proximity to a step discontinuity. These criteria have been chosen so as to penalize image regions that are known to lack of accuracy, in order to deal with data redundancy from one image to another in

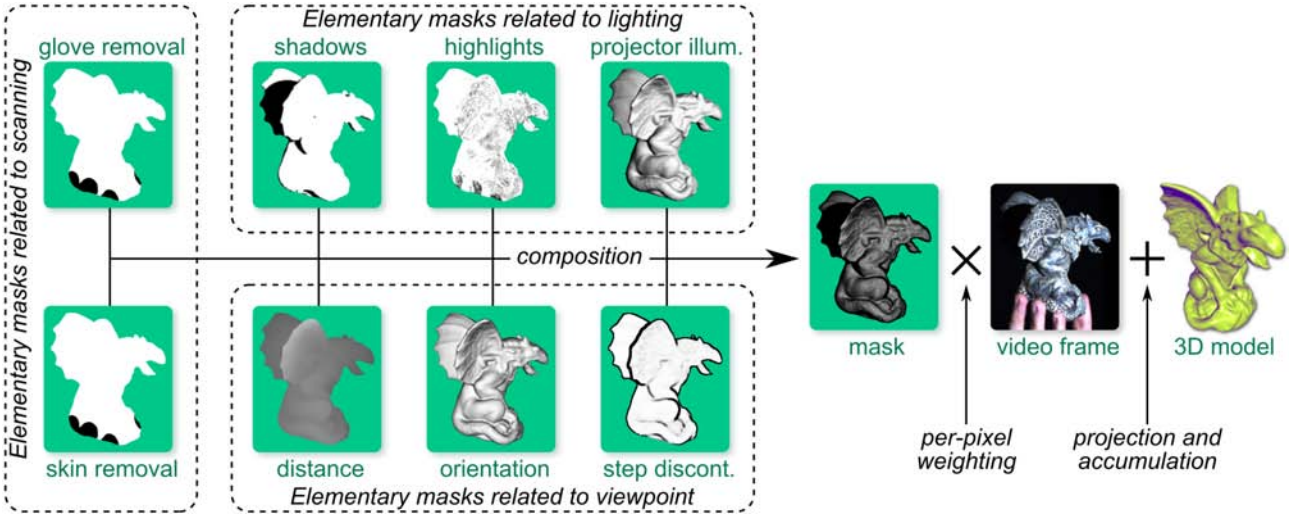

Figure 3. The diffuse color texture is computed as the weighted average of all video frames. Weights are obtained by the composition of multiple elementary masks, each one corresponding to a particular criterion related to viewing, lighting or scanning conditions.

a way that ensures seamless color transitions. More details about these masks can be found in [20].

Although sufficient to avoid texture cracks, these masks cannot handle self projected shadows or specular highlights since knowledge about the lighting is necessary. In our case, both positions of the viewpoint and the light (projector lamp) are always exactly known. Moreover, the light moves with the scanner, which means that highlights and shadows are different for each frame, as well as the illumination direction. We then define the three following additional masks, that aim at making prevailing image parts deprived of illumination effects:

- *Shadows*. Since the complete geometric configuration of the scene is known, we can use a simple shadow mapping algorithm to estimate shadowed areas, to which a null weight is assigned.
- *Specular highlights*. Conversely to shadows, highlights partially depend on the object material, which is unknown. For this reason, we use a multi-pass algorithm to detect them. The first pass computes the object texture without accounting for highlights. Due to the high data redundancy, the averaging tends to reduce their visual impact. During the subsequent passes, highlights are identified by computing the luminosity difference between the texture obtained at the previous pass and the input picture. This difference corresponds to our highlight removal mask. In practice, only two passes are sufficient.
- *Projector illumination*. This mask aims at avoiding luminosity loss during the averaging by giving more influence to surface parts facing the light source. It corresponds to the dot product between the surface normal and the line of sight.

We also introduce two other masks to cope with the occlusions that are inherent to in-hand scanning. Indeed, if they are ignored, picture regions corresponding to the operator's hand may be introduced in the computation, leading to visible artifacts in the final texture. Thus, when digitization is performed with the dark glove, an occlusion mask is simply computed by thresholding pixel intensities. In the case of a digitization made by hand, the mask corresponds to the aforementioned skin probability map produced by the scanner.

Each elementary mask contains values in the range $]0, 1]$, zero being excluded in order to ensure that texturing is guaranteed for every surface point that is visible in at least one picture. They are multiplied all together to produce the final mask that selectively weights the pixels of the corresponding picture. During this operation, each elementary mask can obviously be applied more than once. The influence of each criterion can then be tuned independently, although we empirically determined default values that work quite well for most cases.

C. Recovery of a texture of details

Despite the fact that in-hand scanning is a really convenient technology, it often leads to a loss of accuracy with respect to traditional scanning devices, thus preventing the acquisition of the finest geometric details. Nevertheless, thanks to the fact that we know the light position for each video frame, it is possible to partially recover them by using a photometric stereo approach.

Photometric stereo consists in computing high quality normal/range maps by taking several photographs from the same viewpoint but with different illumination directions [37]–[39], or by moving the object in front of a camera and a light source that are fixed with respect to each other [40], [41]. We use here a similar approach for extracting a normal map from the video flow produced by the in-hand scanner.

In the following, vectors are assumed to be column vectors. Let $\{F_i\}$ be the set of frames corresponding to the acquisition sequence. A light position ℓ_i, corresponding to the scanner projector's location, is associated to each frame F_i. Assuming that the object surface is Lambertian, the color c_i observed at a given surface point p in F_i is

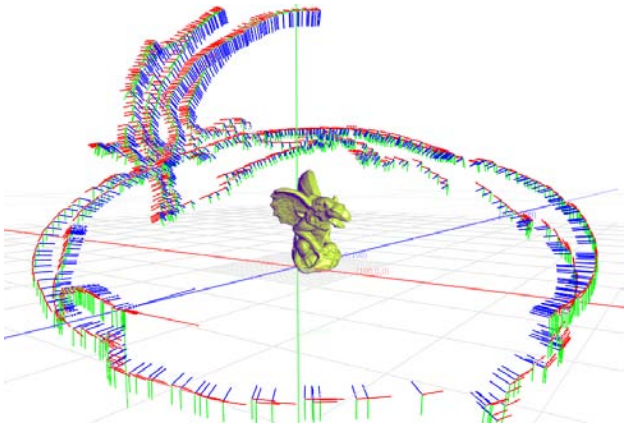

Figure 4. Example of a 3D model with the associated acquisition frames, which shows a typical path for a small object.

given by equation 1:

$$c_i = \frac{\rho_d}{d_i^2} \left(n_p^T \; l_i\right) \tag{1}$$

with:

$$d_i = \|\ell_i - p\| \quad \text{and} \quad l_i = \frac{\ell_i - p}{\|\ell_i - p\|} \tag{2}$$

where n_p is the normal at p and ρ_d a constant depending on the light intensity, the surface diffuse albedo and the camera transfer function. If p is visible in several frames $\{F_i\}_{1 \leq i \leq N}$, the normal n_p that fits the best the N measurements in the least square sense can be found by solving the following equation:

$$\rho_d \; n_p = \arg\min \; \xi(X) \tag{3}$$

where ξ is a quadratic form defined as:

$$\xi(X) = (LX - C)^2 \tag{4}$$

with:

$$L = \begin{bmatrix} w_1 \; l_1^T \\ \vdots \\ w_N \; l_N^T \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} w_1 \; c_1 \; d_1^2 \\ \vdots \\ w_N \; c_N \; d_N^2 \end{bmatrix} \tag{5}$$

In equation 5, w_i represents a confidence value for the i^{th} measurement F_i, that corresponds in our case to the values of the weighting masks presented in the previous section. Solving equation 3 is equivalent to finding the value of X for which the first derivative of ξ is null, which leads to equation 6:

$$\xi'(X) = 0 \quad \Longleftrightarrow \quad X = (L^T L)^{-1} \; (L^T C) \tag{6}$$

The normal vector n_p is finally obtained by normalizing X.

This fitting method works well when light directions are correctly distributed on the hemisphere over p. However, in the case of high-speed scanning systems, digitization is generally performed by following a simple trajectory (sometimes using a turntable) which often leads to a direction sampling almost constrained to a plane (see Figure 4). This uneven sampling distribution may result in an estimated normal which is reliable along this plane

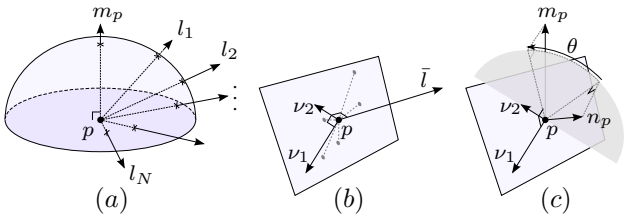

Figure 5. Outline of our normal fitting correction.

but really uncertain along the orthogonal direction.

To alleviate this problem, we propose to analyze the sampling distribution at each point p by performing a PCA on the set of light directions. The idea is then to rotate the fitted normal n_p so as to move it closer to the initial mesh normal m_p, but only along the direction of smallest sampling dispersion. To achieve this, we first compute the mean light direction $\bar{l}$ (Figure 5-a). Vectors l_i are then projected onto the plane passing through p and orthogonal to $\bar{l}$, and the PCA is performed on the resulting set of 2D points. This leads to two unit eigenvectors ν_1 and ν_2, as well as their respective eigenvalues λ_1 and λ_2, with $\lambda_1 > \lambda_2 > 0$ (Figure 5-b). By definition, ν_2 is the direction along which the sampling is the poorest. The correction is then applied by rotating n_p around ν_1 of an angle $\left(1 - \frac{\lambda_2}{\lambda_1}\right)\theta$, where θ is the angle between the projections of n_p and m_p onto the orthogonal plane to ν_1 (Figure 5-c).

When the sampling is well distributed over the hemisphere of light directions, the eigenvalue ratio is close to 1 and then the fitted normal is almost not modified. On the contrary, when this ratio decreases, it means that the sampling distribution is not regular. In this case, n_p is corrected but only along the less reliable direction.

IV. Web Publishing

After geometry and texture images have been processed, the third and last stage of our pipeline optimizes the generated data for network transmission and real-time rendering on standard Web browsers. The optimized version of the 3D model is stored in the server file system and is accessed by a standard HTTP server to serve requests of visualization clients. In the following, we describe the steps we use for preparing and storing the data.

A. Data optimization

The optimization phase is composed of two sequential steps: *geometry partitioning* and *rendering optimizations*. The goal is to create a data representation that minimizes the work needed by both server and client to retrieve and access it, and whose layout exploits the full power of WebGL for real-time rendering.

a) Geometry partitioning: we use an indexed triangle mesh to represent the geometry information of the 3D model, storing in two separate arrays the

Information on acquired data					*Processing times (sec)*					*Information on produced data*			
Model	*Model height (cm)*	*Mesh size (tri.)*	*Video resol. (pixels)*	*Sequence length (frames)*	In-hand scanning	*Texturing* *Color only*	*Texturing* *Color+ details*	*WebGL Formatting*	*Total*	*Textures resol. (pixels)*	*Textures size (MB)*	*Mesh size (MB)*	*Total size on server (MB)*
Gargoyle	13	41.5K	780×580	1070	54	194	198	< 1	447	1500^2	4.28	1.20	5.48
Strawberry	15	86.5K	780×580	2110	110	266	277	< 1	654	1500^2	3.24	2.47	5.71
Pot	11	179.5K	780×580	330	16	28	29	< 1	74	1024^2	2.44	5.12	7.56

TABLE I.
INFORMATION AND TIMINGS FOR VARIOUS ACQUIRED OBJECTS.

data associated to vertices (i.e. position and texture coordinates) and the connectivity information (stored as triplets of indices in the vertex data array).
One of the fundamental limitation of WebGL is the maximum value for a vertex index: only 8- and 16-bit unsigned integers are allowed as data types, thus limiting the maximum number of addressable vertices to 65536. It is therefore needed to partition the original mesh into sub-meshes, or chunks, each of them containing no more than the allowed maximum amount of vertices. To this end, we use a simple greedy method that iteratively adds triangles to a chunk until the maximum number of vertices is reached.

b) Rendering optimizations: one advantage of the indexed triangle mesh representation is that vertices referenced by more than one triangle need to be stored only once. It has the property to save a significant amount of space for the vast majority of 3D models, for which, on average, a vertex is referenced by six triangles. To convey this advantage from memory occupancy to rendering performances, graphics accelerators have introduced a *vertex cache* capable of storing data associated to up to 32 vertices, thus allowing to reuse the results of a considerable amount of per-vertex calculations.
However, to fully exploit the built-in vertex cache, triangles in the topology array must be rearranged. Even though the problem does not have a polynomial-time solution, several works have been developed [42], [43] that produce a very good approximate solution in a relatively small amount of time. In our pipeline we apply the *tipsify* method [44] on each sub-mesh produced by the previous step. For a whole model, this method is able to create an optimized sequence of triangles in few tenths of a second.

B. Data storage and retrieval

One of our goal is to exploit standard and easily available technologies for making the produced models accessible on the Web platform. To this end, we decided to use the well-known Apache HTTP server and use the server file system as the storage database.
Model data is saved under standard file formats: to store geometry information we use the Stanford polygon file format (PLY), which support multiple vertex attributes and binary encoding, while Portable Network Graphics (PNG) images are used for color and normal textures. Even though those formats are already compact, we take advantage of the automatic compression (*gzip*) applied by the Apache server on data transmission, as well as automatic decompression executed by browsers on data arrival.
To access the remote 3D model, visualization clients use JavaScript to issue a HTTP request with a base URL of the form *http://example-data-domain.org/model-name/*, and appending predefined file names to discriminate among geometry and texture files, such as *geometry.ply*, *color.png* and *normal.png*. Upon data arrival, the appropriate WebGL resources (e.g. vertex/index buffers and textures) are created using SpiderGL, and model visualization and exploration start.

V. RESULTS

We present in this section some results of our pipeline, as well as some implementation details. The proposed objects are a sample of a group of artifacts which were used to test the entire system. In order to provide results, we chose to use objects which were different in size and material, instead of applying the acquisition to sets with similar characteristics. The proposed results and processing times show that an extension to large collections is straightforward.

A. Implementation and performances

It is important to notice that all operations involved in the pre-processing stages consist in local computations, making the process suitable for a GPU support:

- For the color texture, masks are generated thanks to simple image processing shaders applied on the input images or on a rendering of the mesh geometry.
- For the texture of details, both matrices $(L^T L)$ and $(L^T C)$ of equation 6 can be constructed incrementally by processing input pictures one by one on the GPU and accumulating intermediate results using buffer textures.
- Similarly, the PCA used for the analysis of the light sampling distribution can be efficiently computed using shaders, inasmuch as eigenvectors and eigenvalues have a simple analytical formulation in the 2D case.

Moreover, pictures are processed sequentially, which makes the memory consumption independent to the length of the video flow.
The total time needed to acquire and publish a real object varies depending on the size and complexity of the object,

Figure 6. Comparison of texturing obtained by a naive averaging of all input frames (*top row*) and the proposed approach using weighting masks (*bottom row*).

Figure 7. Color reconstruction for the Elephant model.

and the needed accuracy. As shown on Table I, in the case of the objects shown in this paper, the geometry acquisition time is in the order of a few minutes (provided that the user has gained a bit of experience on how to move the object in front of the scanner). The diffuse color and detail textures recovery can take up to 5-10 minutes, while the data optimization for Web publishing is almost instantaneous. Hence, the presented pipeline is usually completed in less than 20 minutes. Moreover, the second and third stages are automatic, so that another object can be acquired while the previous one is under preparation. This permits to obtain the 3D models of several objects within an hour of work.

On the base of this, the proposed system is really suitable for any institution which possesses big collections of similar data, or archives of fragmented pieces. Tens of high quality 3D models can be made available every day, for any kind of use (archival, study, presentation to the public).

B. Diffuse color texture reconstruction

Our texturing results are shown in the bottom row of Figure 6. The top row proposes a comparison to the results obtained by a naive approach that performs a direct averaging of color contributions, ignoring weighting masks. The most obvious difference that can be noticed is clearly the drastic loss of luminosity that occurs in the case of the naive approach. As expected, the *projector illumination* mask tends to increase the influence of image regions that correspond to the most illuminated surface parts, which leads to a conservation of luminosity.

Other improvements can be observed. For the Gargoyle model (*left*), we can see that fine details on the wings are much less blurry when the weights are introduced in the computations, thanks to the fact that the surface orientation with respect to the viewpoint is considered. For the Pot model (*middle*), the big vertical crack in the white rectangle results from the fact that one portion of the surface was depicted by a much greater number of frames with respect to the adjacent one: this produces an imbalance among the number of summed color contributions, and the consequent abrupt change of color. Thanks to masks related to visibility and illumination criteria, with our approach the big crack completely disappears. For the Strawberry model (*right*), some regions are slightly brighter in the case of the naive texturing. They arise from strong specular highlights in the input pictures and are highly reduced in the case of our method.

Figure 7 shows a color texture reconstructed for the Elephant model. In spite of its relatively complex geometry, it shows a color information of a significant quality, without cracks or noticeable visual artifacts.

During this texturing phase, the only parameters that must be set by the user are the number of applications for each elementary mask. As said before, default values working well for most cases have been determined. Nonetheless, considering the aforementioned computation times, the user can easily try several combinations by relaunching the texturing so as to see how different values impact the final result. The set of parameters is then really small and can be tuned in an easy and intuitive manner.

It is important to note that, to ensure the complete automation of the processing phase, the acquisition step must provide data guarantying a complete coverage of color measurement. For scanners designed like the one we used, it is not necessarily induced by a complete coverage of the shape, since color and depth are note computed by the same camera. If the coverage is not good enough, holes may appear in the reconstructed color/normal texture. However, this problem can be solved easily by an assisted user intervention, as we already explained in [45].

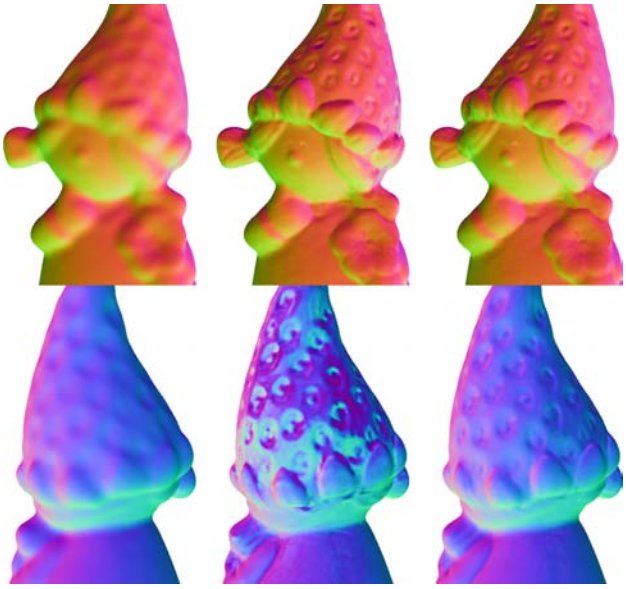

Figure 8. Differences induced by our constrained fitting on the normal computation for a region with a good sampling (*top row*) and a region with a poor sampling (*bottom row*). *From left to right:* initial mesh normals; normals from unconstrained fitting; normals from our constrained fitting.

C. Detail texture reconstruction

Figure 8 illustrates the efficiency of our normal correction procedure by showing the normal field computed for the same object with and without correction. The top row shows a region of the object where the sampling is well distributed (namely, that mostly covers the hemisphere of the possible lighting directions). There is almost no difference between both versions of the normal field, which is obvious since the ratio of the eigenvalues given by our PCA approach is, in this case, really close to 1. Both fittings then behave similarly.

Conversely, the bottom row shows a surface part sampled from scanner locations that are almost collinear. As it can be seen, the standard fitting approach produces normals that present an exaggerated curvature as well as some cracks in the surface orientation. By introducing our correction, this erroneous behavior of the normal field completely disappears. In this case, since the eigenvalue ratio decreases, the estimated normal is forced to get

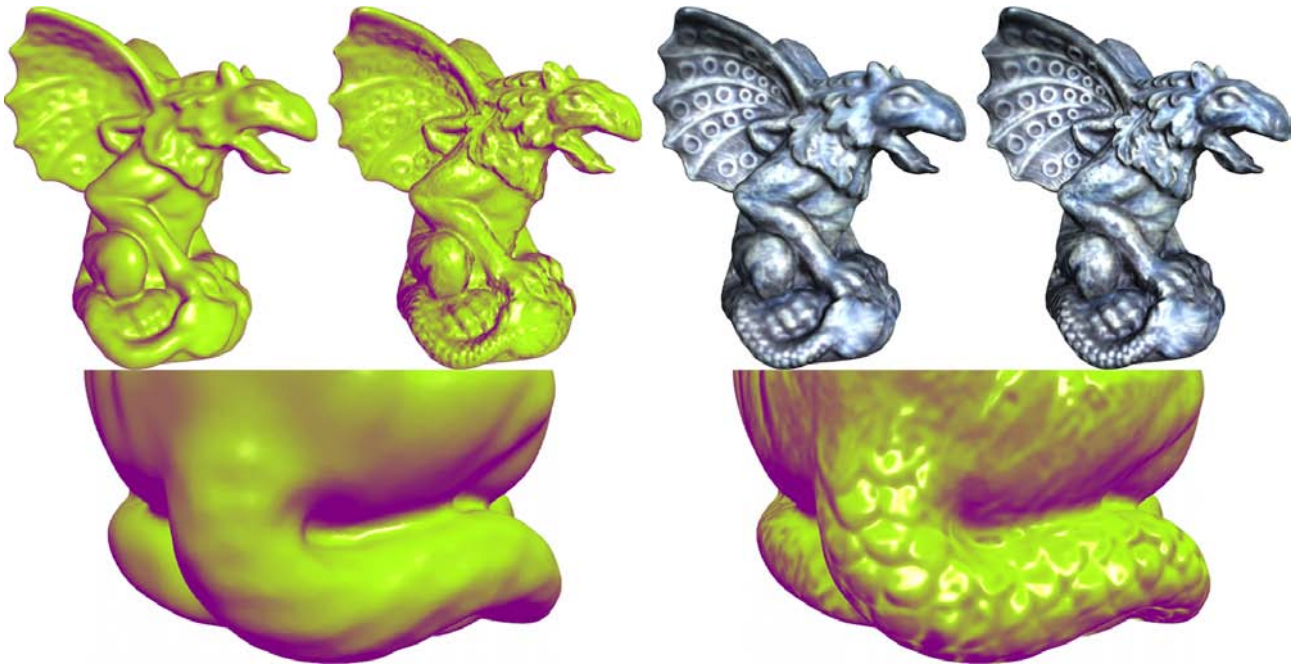

Figure 9. The Gargoyle model rendered with and without color, and with and without normal map. *Bottom row*: close-up on the tail details.

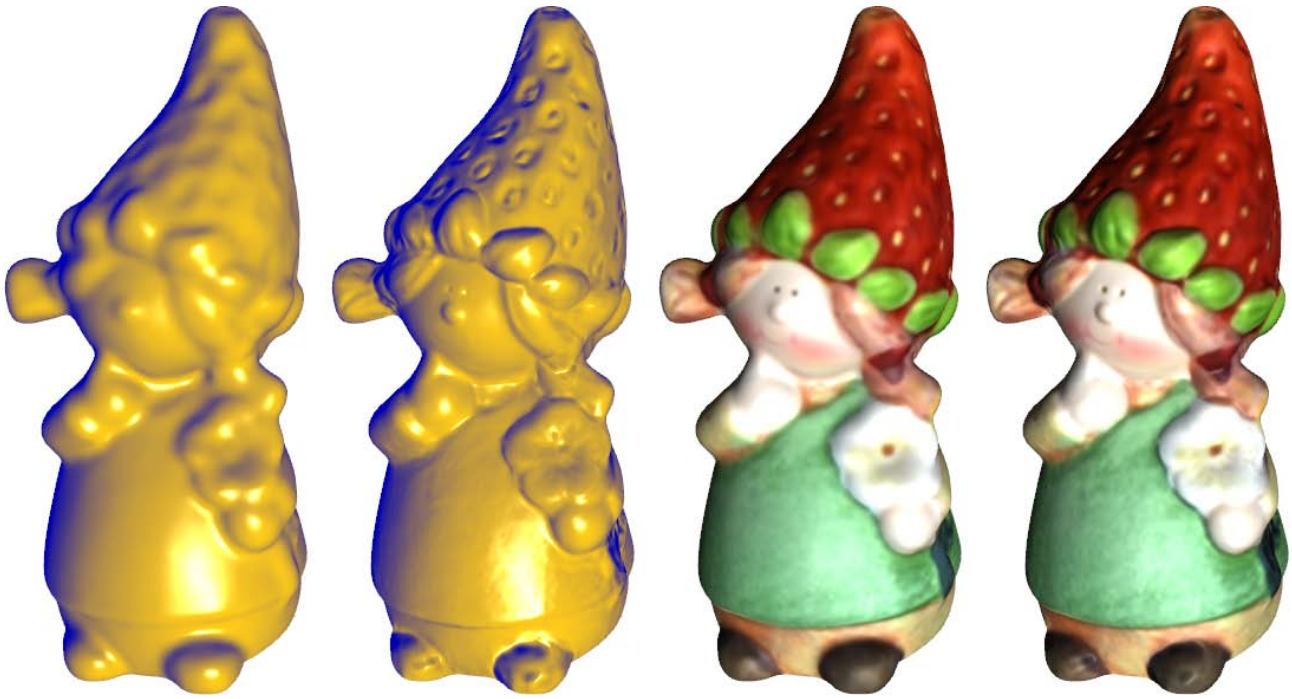

Figure 10. The Strawberry model rendered with and without color, and with and without normal map.

closer to the original mesh normal along the direction of highest uncertainty. However, as explained before, the estimated normal is not modified along the direction where the sampling has its best distribution. This is why feature enhancing can still be achieved.

Figures 9 and 10 show different renderings of the Gargoyle and the Strawberry models, with and without the normal map extracted by our system. As it can be seen, high-frequency shape details are clearly missing in the rough geometry acquired by the in-hand scanner. Thanks to our shape-from-shading approach, these details can be accurately captured and rendered afterward by bump mapping.

Figures 11 and 12 show different renderings of the final digital copies obtained by our pipeline from two Buddha statues. The frames on the right side of these images highlight once again the improvement resulting from the estimated texture of details. Particularly, for the white Buddha, the granularity of the stone on the broken part is well recovered by our technique, where it is completely

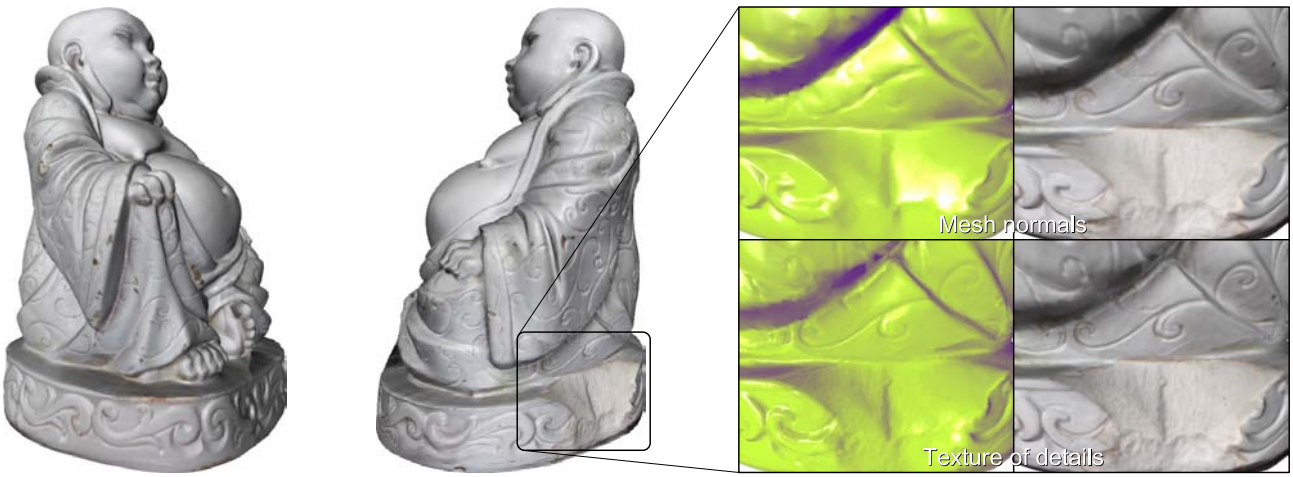

Figure 11. The white Buddha model.

Figure 12. The gold Buddha model.

absent from the initial geometry. Similarly, the veins of the wood material the gold Buddha is made of become visible thanks to the added details.
The main advantage of this correction approach is that a good quality detail texture can be recovered even in the presence of a poor sampling. Thus, the user do not have to perform on purpose an exhaustive measurement just to satisfy the fitting constraints. Once again, this leads to an automatic solution that uses data generated by high speed scanners as they are, without specific prerequisites.

D. Visualization in Web pages

The output produced by the third stage of our pipeline is used to prove the feasibility of a world-wide dissemination and accessibility of 3D digital copies of CH artifacts. We implemented a visualization client directly inside a HTML Web page using JavaScript and the SpiderGL library. Examples of the obtained rendering quality are shown in Figure 13. We developed a JavaScript utility library that allows designers of Web pages to integrate a 3D model renderer in a very simple way; in fact, it is only needed to provide the model URL and the HTML canvas element that will be used to display the rendering output and to handle user inputs. The interaction metaphor known as *world-in-hand* or *trackball* has been used to facilitate the artifact inspection by using the mouse.
After data has been retrieved as the result of an asynchronous XmlHttpRequest to the server, the client creates the WebGL resources related to model geometry (e.g. vertex and index buffers), and color/normal surface attributes image maps (e.g. textures). To measure the peak performances of our renderer, we installed a custom event handler on the page window that executes a frame redraw and re-fires the event, thus creating an infinite rendering loop; this is because the minimum time interval exposed by a JavaScript timer handler exceeds the time needed to render a single frame, thus becoming the bottleneck. As shown at the bottom of each Web page snapshot in Figure 13, the rendering performance is in the order of thousands of frames per second (FPS) for models that range from 50K to 100K triangles. Being 60 FPS the target performance for a very smooth interaction, reaching far higher rates allows us to use, if required, 3D models consisting of millions of triangles and to continue offering a proper user experience.

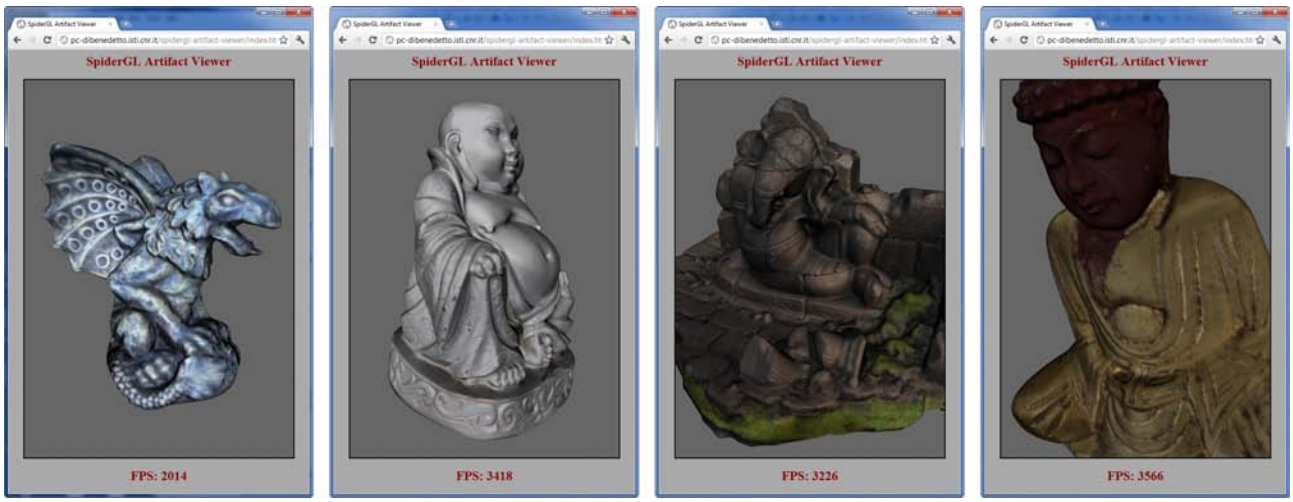

Figure 13. Rendering examples of our 3D digital artifacts using the SpiderGL library on an common Web browser.

VI. Conclusion

In this paper, we presented a complete pipeline for the creation of Web browsable digital content from real objects, consisting in 3D models enhanced by two textures respectively encoding artifact free color and fine geometric details. The design of this pipeline has been oriented to automation, so that the only task that has to be done by the user is the digitization, which relies on the use of a real-time in-hand scanner so as to make the task easy and intuitive, even for non expert users. All the following processing steps do not involve any kind of technical knowledge, nor interventions, and lead to data that can be directly shared on Internet through an Apache server, and visualized on the client side with a standard computer thanks to WebGL technologies.

Even though the proposed approach is generic enough to be used in any application for which producing and sharing digital content about real artifacts present an interest, its three main advantages (namely its ease of use, its high automation and its quickness) make it particularly appropriate to cases where huge collections have to be processed. This is why we think that our system greatly benefits CH, especially if we consider that, nowadays, this field still lacks complete, automatic and unified tools for the digital archiving and sharing.

Regarding future improvements, one of the current limitations of the system is that, during the scanning phase, only a feedback about geometry coverage is given. It is not easy to know in advance if enough frames have been acquired so that accurate color or fine geometric detail can be extracted safely. If the coverage is not good enough, holes may appear in the reconstructed textures, as already discussed in the result section. Nevertheless, an interesting improvement could be to obtain real-time feedback also about color and surface coverage. This can be implemented by analyzing how many frames (and directions of view) cover each part of the object.

Other appealing directions of work could include the possibility to enrich the Web publishing phase, by automatically formatting a Web page based not only on the 3D model, but on other types of data, like text and images. Moreover, since all the frames of the video stream are aligned to the 3D model, all or a portion of them could be made visible in order to add 2D data which is calibrated to the geometry. Hence, a "Photo Tourism-like" [46] image navigation could be possible.

Acknowledgment

The research leading to these results has received funding from the EU 7^{th} Framework Program (FP7/2007-2013), through the 3D-COFORM project, under grant agreement n. 231809. We thank the ETH of Zürich, Switzerland, for having provided the datasets from their in-hand scanner.

References

[1] O. Hall-Holt and S. Rusinkiewicz, "Stripe boundary codes for real-time structured-light range scanning of moving objects," in *ICCV 2001*, 2001, pp. 359–366 vol.2.

[2] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3d model acquisition," *ACM Trans. Graph.*, vol. 21, pp. 438–446, July 2002.

[3] T. Weise, T. Wismer, B. Leibe, and L. V. Gool, "In-hand scanning with online loop closure," in *3DIM09*, October 2009.

[4] L. Zhang, B. Curless, and S. Seitz, "Spacetime stereo: shape recovery for dynamic scenes," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, june 2003, pp. II – 367–74 vol.2.

[5] T. Weise, B. Leibe, and L. V. Gool, "Fast 3d scanning with automatic motion compensation," in *IEEE CVPR'07*, June 2007.

[6] F. Larue and J.-M. Dischler, "Automatic registration and calibration for efficient surface light field acquisition," in *7th VAST International Symposium on Virtual Reality, Archeology and Cultural Heritage*. Eurographics, 2006, pp. 171–178.

[7] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, and D. Cohen-Or, "Seamless montage for texturing models," *Comput. Graph. Forum*, vol. 29, no. 2, pp. 479–486, 2010.

[8] K. Ikeuchi, T. Oishi, J. Takamatsu, R. Sagawa, A. Nakazawa, R. Kurazume, K. Nishino, M. Kamakura, and Y. Okamoto, "The great buddha project: digitally archiving, restoring, and analyzing cultural heritage objects," *Int. J. Comput. Vision*, vol. 75, no. 1, pp. 189–208, 2007.

[9] I. Stamos, L. Liu, C. Chen, G. Wolberg, G. Yu, and S. Zokai, "Integrating automated range registration with multiview geometry for the photorealistic modeling of large-scale scenes," *Int. J. Comput. Vision*, vol. 78, pp. 237–260, July 2008.

[10] M. Corsini, M. Dellepiane, F. Ponchio, and R. Scopigno, "Image-to-geometry registration: a mutual information method exploiting illumination-related geometric properties," *Computer Graphics Forum*, vol. 28, no. 7, pp. 1755–1764, 2009.

[11] T. Franken, M. Dellepiane, F. Ganovelli, P. Cignoni, C. Montani, and R. Scopigno, "Minimizing user intervention in registering 2D images to 3D models," *The Visual Computer*, vol. 21, no. 8-10, pp. 619–628, sep 2005.

[12] J. Dorsey, H. Rushmeier, and F. Sillion, *Digital modeling of material appearance*. Morgan Kauf./Elsevier, 2007.

[13] M. Callieri, P. Cignoni, and R. Scopigno, "Reconstructing textured meshes from multiple range rgb maps," in *7th Int.l Fall Workshop on Vision, Modeling, and Visualization 2002*, Erlangen (D), Nov. 20 - 22 2002, pp. 419–426.

[14] N. Bannai, A. Agathos, and R. Fisher, "Fusing multiple color images for texturing models," in *3DPVT04*, 2004, pp. 558–565.

[15] M. Chuang, L. Luo, B. Brown, S. Rusinkiewicz, and M. Kazhdan, "Estimating the laplace-beltrami operator by restricting 3d functions," in *Proceedings of the Symposium on Geometry Processing*, July 2009, pp. 1475–1484.

[16] K. Pulli, H. Abi-Rached, T. Duchamp, L. Shapiro, and W. Stuetzle, "Acquisition and visualization of colored 3d objects," in *Proc. of ICPR 98*, 1998, pp. 11,15.

[17] F. Bernardini, I. Martin, and H. Rushmeier, "High-quality texture reconstruction from multiple scans," *IEEE Tr. on Visual. and Comp. Graph.*, vol. 7, no. 4, pp. 318–332, 2001.

[18] A. Baumberg, "Blending images for texturing 3d models," in *BMVC 2002*. Canon Research Center Europe, 2002.

[19] V. Rankov, R. Locke, R. Edens, P. Barber, and B. Vojnovic, "An algorithm for image stitching and blending," in *SPIE*, vol. 5701, 2005, pp. 190–199.

[20] M. Callieri, P. Cignoni, M. Corsini, and R. Scopigno, "Masked photo blending: mapping dense photographic dataset on high-resolution 3d models," *Computer & Graphics*, vol. 32, no. 4, pp. 464–473, Aug 2008.

[21] M. Eisemann, B. D. Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent, "Floating textures," *Computer Graphics Forum (Proc. Eurographics EG'08)*, vol. 27, no. 2, pp. 409–418, 4 2008.

[22] M. Dellepiane, M. Callieri, R. Marroquim, P. Cignoni, and R. Scopigno, "Flow-based local optimization for image-to-geometry projection," *IEEE Transaction on Visualization and Computer Graphics*, vol. (in press), 2011.

[23] J. Unger, A. Wenger, T. Hawkins, A. Gardner, and P. Debevec, "Capturing and rendering with incident light fields," in *EGRW '03*, 2003, pp. 141–149.

[24] M. Dellepiane, M. Callieri, M. Corsini, P. Cignoni, and R. Scopigno, "Improved color acquisition and mapping on 3d models via flash-based photography," *ACM Journ. on Computers and Cultural heritage*, vol. 2, no. 4, pp. 1–20, Feb. 2010.

[25] "Microsoft ActiveX Controls," http://msdn.microsoft.com/en-us/library/aa751968(VS.85).aspx.

[26] D. Raggett, "Extending WWW to support platform independent virtual reality," *Technical Report*, 1995.

[27] L. D. Don Brutzmann, *X3D: Extensible 3D Graphics for Web Authors*. Morgan Kaufmann, 2007.

[28] Khronos Group, "WebGL - OpenGL ES 2.0 for the Web," 2009.

[29] Khronos Group, "Khronos: Open Standards for Media Authoring and Acceleration," 2009.

[30] Khronos Group, "OpenGL ES - The Standard for Embedded Accelerated 3D Graphics," 2009.

[31] L. Kay, "SceneJS," http://www.scenejs.com, 2009.

[32] P. Brunt, "GLGE: WebGL for the lazy," http://www.glge.org/, 2010.

[33] M. Di Benedetto, F. Ponchio, F. Ganovelli, and R. Scopigno, "SpiderGL: A JavaScript 3D Graphics Library for Next-Generation WWW," in *Web3D 2010. 15th Conference on 3D Web Technology*. Web3D Consortium, 2010.

[34] B. DeLillo, "WebGLU: A utility library for working with WebGL ," http://webglu.sourceforge.org/, 2009.

[35] "3d scans of cuneiform tablets," More info on: http://www.cognitiones.de/doku.php/3d_scans_of_cuneiform_tablets.

[36] B. J. Brown, C. Toler-Franklin, D. Nehab, M. Burns, D. Dobkin, A. Vlachopoulos, C. Doumas, S. Rusinkiewicz, and T. Weyrich, "A system for high-volume acquisition and matching of fresco fragments: Reassembling Theran wall paintings," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 27, no. 3, Aug. 2008.

[37] H. E. Rushmeier, G. Taubin, and A. Guéziec, "Appying shape from lighting variation to bump map capture," in *Proc. of the Eurographics Workshop on Rendering Techniques '97*, London, UK, 1997, pp. 35–44.

[38] V. Nozick, "Pyramidal normal map integration for real-time photometric stereo," in *EAM Mechatronics 2010*, Japon, 2010, pp. 128–132.

[39] T. Malzbender, B. Wilburn, D. Gelb, and B. Ambrisco, "Surface enhancement using real-time photometric stereo and reflectance transformation," in *Eurographics Symposium on Rendering/Eurographics Workshop on Rendering Techniques*, 2006, pp. 245–250.

[40] J. Lim, J. Ho, M.-H. Yang, and D. Kriegman, "Passive photometric stereo from motion," in *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ser. ICCV '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 1635–1642.

[41] T. Higo, Y. Matsushita, N. Joshi, and K. Ikeuchi, "A hand-held photometric stereo camera for 3-d modeling." in *ICCV*. IEEE, 2009, pp. 1234–1241.

[42] H. Hoppe, "Optimization of mesh locality for transparent vertex caching," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 269–276.

[43] G. Lin and T. P. Y. Yu, "An improved vertex caching scheme for 3d mesh rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, pp. 640–648, July 2006.

[44] P. V. Sander, D. Nehab, and J. Barczak, "Fast triangle reordering for vertex locality and reduced overdraw," *ACM Trans. Graph.*, vol. 26, July 2007.

[45] F. Larue, M. Dellepiane, H. Hamer, and R. Scopigno, "Automatic texturing without illumination artifacts from in-hand scanning data flow," in *Int. Workshop on Multimedia for Cultural Heritage (MM4CH)*. Springer, April 2011.

[46] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from Internet photo collections," *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, November 2008.