Acquiring, Stitching and Blending Diffuse Appearance Attributes on 3D Models

C. Rocchini[*], P. Cignoni[†], C. Montani[‡], R. Scopigno[§]

Istituto Scienza e Tecnologia dell'Informazione[¶]

Consiglio Nazionale delle Ricerche

July 10, 2001

Corresponding author:

R. Scopigno, CNUCE-CNR, Area della Ricerca CNR, Loc. S. Cataldo, 56100 Pisa, ITALY

Email: roberto.scopigno@cnuce.cnr.it

Phone: +39 050 315 2929 Fax: +30 050 313 8091

Abstract

A new system for the construction of highly realistic models of real free-form 3D objects is proposed, based on the integration of several techniques (automatic 3D scanning, inverse illumination, inverse texture-mapping and textured 3D graphics). Our system improves the quality of a 3D model (e.g. acquired with a range scanning device) by adding color detail and, if required, high frequency shape detail. Detail is obtained by processing a set of digital photographs of the object. This is carried out by performing several subtasks: to compute camera calibration and position, to remove illumination effects obtaining both illumination-invariant reflectance properties and a high resolution surface normal field, and finally to blend and stitch the acquired detail on the triangle mesh via standard texture mapping. In particular, the smooth join between different images that map on adjacent sections of the surface is obtained by applying an accurate piecewise local registration of the original images and by blending textures. For each mesh face which is on the adjacency border between different observed images, a corresponding triangular texture patch can also be resampled as a weighted blend of the corresponding adjacent image sections. Examples of the results obtained with sample works of art are presented and discussed.

Keywords: 3D scanning, image processing, inverse illumination, texture_to_geometry registration, texture mapping,.

[*]Email: rocchini@iei.pi.cnr.it

[†]Email: cignoni@iei.pi.cnr.it

[‡]Email: montani@iei.pi.cnr.it

[§]Email: roberto.scopigno@cnuce.cnr.it

[¶]I.S.T.I. - C.N.R. (formerly I.E.I.-C.N.R. and CNUCE-C.N.R.), Area della Ricerca C.N.R., Loc. S. Cataldo, 56100 Pisa, ITALY

1 Introduction

Many applications require an accurate digital representation of both the shape and the surface attributes (e.g. the color) of an object of interest. An example is the representation and visualization of Cultural Heritage artifacts, which often require such high accuracy that the standard CAD modeling approach cannot be adopted [18]. The ideal situation would be to have an automatic 3D digital reproduction system capable of quickly and inexpensively building a very accurate and high resolution model from the real artifact (hereafter, we assume that shape is represented by a standard triangle mesh encoding). Unfortunately, this only partially applies to the current situation. Among the many automatic acquisition technologies proposed [22], optical range scanners provide the accuracy required by highly demanding applications and are fortunately becoming slightly cheaper, but performing a 3D scan of a complex object is still neither simple nor fast. Models with a complex topology are generally acquired with multiple scans, which have to be integrated into a single mesh [18]. The resolution of the acquisition device directly determines the complexity of the triangle mesh or points cloud produced. Mesh simplification tools [12, 7] are generally adopted to reduce the shape redundancy of the output produced, and to support simple management of the models even on a portable computer. One critical question is how to acquire the *surface attributes* and how to link it to the shape description [1, 14, 19, 10, 28, 31, 29, 25, 27, 35, 20]. The term "surface attributes" can be used to represent different concepts: from the color texture observed on the object surface, which very much depends on the lighting conditions, to the illumination-invariant surface reflectance properties (also called *albedo*), computed by removing highlights and diffuse shading or even by computing a bidirectional reflectance distribution function (BRDF). In this paper we focus mainly on the acquisition and management of surface attributes, and in particular on how we can acquire, blend and map on a standard textured mesh the detail contained in a set of images taken from different view points (see Figure 1).

Some possible problems found in current 3D scanning technology, with respect to surface attributes acquisition, are:

- some devices do not support surface attributes acquisition (e.g. contact scanners, some optical scanners, CAT devices);
- the insufficient accuracy or resolution of the surface attributes acquired by many commercial range scanners. Some applications may require the resolution of the color attribute to be higher than the shape resolution (consider again the acquisition of an artistic or archeological object with a very complex pattern on its surface). Moreover, controlling the lighting environment is not easy with commercial devices, and it is mandatory to obtain good quality results. Many applications require the acquisition of the illumination-invariant reflectance properties, and not the simple observed surface colors;
- insufficient software support for the smooth and accurate integration of the color data contained in multiple scans.

(a) (b) (c)

Figure 1: Some synthetic images of our *vase*: it is rendered without color in (a), with a naive mapping of the color textures acquired by a commercial laser scanner in (b) and, finally, with our un-shaded, locally registered and cross-faded textures in (c).

Our approach allows to de-couple, if necessary, the acquisition of shape and surface attributes. This allows the use of the most suitable 3D scanning technology available for acquiring the geometry of the target object. Our constraints and goals are as follows:

- the surface attributes acquisition should guarantee good accuracy and high resolution, especially in the case of objects with a complex shape and a highly textured surface with non-uniform reflectance properties (which cannot be acquired using most of the proposed approaches [1, 28, 19]);
- the hardware needed should be limited to a range scanning device and a cheap digital camera; the use of any non-standard or expensive devices should be avoided (e.g. special scanners [1] or robotic arms [29]);
- the complexity of the overall pipeline (acquisition and data post-processing) should be sufficiently low to permit execution on a low performance portable computer; moreover, the processing pipeline should be as automatic as possible, with only limited user intervention.

Our approach can be classified as *hybrid image-based modeling*. We propose the integration of different techniques (range scanning, inverse illumination, inverse texture-mapping and surface-based 3D graphics) to extract surface attributes from images and to past them onto a standard triangle-based mesh (see Section 2). The proposed system consists of four software modules, which operate in sequence. A set of images of the real object is acquired from different viewpoints (see Section 3). Each image is processed to obtain illumination-invariant color detail (here called *un-shaded* color) and geometry detail (a field of surface normal vectors). Because we want to produce an output suitable for rendering with standard texture-mapped polygon-based rendering API, the surface reflectance properties we compute

need not be extremely sophisticated: we assume that the target object possesses a Lambertian surface (see Section 4). Then, the un-shaded images are roughly *registered* on the 3D mesh (see Section 5). An optimal correspondence between regions of the 3D mesh and sections of the un-shaded images is built in the texture *stitching and blending* phase (see Section 6). In particular, a new approach is proposed to produce a smooth join between different images that map on adjacent sections of the surface (where it is crucial to preserve the continuity of the color attribute feature lines, e.g. the edges or lines on a painted surface or the grain of a wooden object, and the continuity of chromatic content)[1]. This is done by applying an accurate piecewise local registration of the un-shaded images: an overlapping region between adjacent patches is defined on the mesh, and a local registration is computed for each vertex in this region. For each mesh face which is in the overlapping region, texture blending is operated either: (a) by resampling a new texture patch, obtained as a weighted blend of the corresponding adjacent images; or (b) by simply blending the corresponding texture sections via multiple rendering of the corresponding frontier faces. Finally, all active texture patches are packed into a single rectangular texture. An evaluation of the proposed approach is presented in Section 7 and concluding remarks are in the last section.

2 Acquisition and mapping of color onto 3D shapes

Our approach de-couples shape and detail acquisition. We assume we have an accurate 3D model (a triangle mesh M) of the object of interest. The pipeline of the processing phases which support color acquisition and stitching is as follows.

Acquisition of surface attributes. Our detail acquisition methodology requires only inexpensive hardware: a digital still camera or a high-resolution video camera. We shoot images from a set of different view points and take *multiple images* for each viewpoint, with different lighting conditions for each viewpoint. The set of viewpoints needs to be planned such that each area of the sample object is visible at least from one viewpoint.

Un-shading, or computing illumination-invariant images. We take multiple images for each view to allow an easy removal of the illumination effects (light shading, highlights and shadows) from the images of the object, which may have a surface with *non-uniform* reflectance properties. We compute the object surface reflectance properties only partially: we do not produce the BRDF relative to the sampled object surface [35], but only determine the diffuse reflection coefficients i.e. the RGB albedo) for each observed surface point. The result of this phase is a set of so called *un-shaded* images, one for each of the initial views. Moreover, a by-product of this process is that we also compute a discrete field of normal vectors on the mesh surface.

Image Registration. All the different views are registered with respect to the object shape. This is a

[1]The *stitching and blending* approach, described in Section 6, has been presented in a preliminary version in [6].

well known problem, and we adopt a classical approach [33] based on the interactive selection of corresponding point pairs. The output of this phase is the definition, for each set of images taken from the same view, of the associated camera calibration and pose parameters (i.e. the inverse projection matrix).

Texture Patching and Stitching. In this phase all the detail images are subdivided into patches, stitched on the shape mesh and partially fused, to generate a single textured triangle mesh. Intuitively, we build an optimal patchwork such that all of the object surface is covered by image patches and the adjacent image patches join smoothly on the object surface (both in terms of color features and chromatic continuity). The texture patching process consists of four sub-phases:

1. ***vertex–to–image* binding:** for each mesh vertex v, we detect the subset of *valid* images that contain a *valid* projection of vertex v. Among all the valid images, we initially select as the *target* image the one which is most orthogonal to the surface in v;

2. **patch growing:** the *vertex-to-image* relation is iteratively refined, by visiting the vertex adjacency graph, to obtain an optimal texture patching. We iteratively change the *vertex-to-image* links, among the possible valid ones, to obtain larger contiguous sections of the mesh that map to the same target image (i.e. such that all three vertices of each triangular face in the mesh section are linked [if possible] to the same image, and adjacent faces are linked [if possible] to the same image);

3. **patch boundary blending:** after the previous step, each mesh face can have vertices associated with either the same image, or different images. We call the latter *frontier* faces, i.e. faces which are on the border of adjacency between different target images. One of our goals is to prevent the production of discontinuity in the representation of surface attributes. Therefore, we first apply a *local registration* to all of the vertices of the frontier faces; second, we either resample a new triangular texture patch for each of these faces (computed as a weighted composition of the corresponding triangular sections in the associated target images), or cross-fade the corresponding texture sections by rendering each frontier face twice or three times;

4. **texture-patches packing:** all the target image patches and the set of resampled triangular patches (if we have resampled some of them) are packed into a single rectangular texture map, and texture coordinates in the triangle mesh are updated accordingly.

Alternative texture patching approaches were proposed in [21, 20]. The approach proposed in [21] takes into consideration the issues regarding the visibility of the captured images and the smooth transition between adjoining textures, and the solution proposed performs a weighted blend of all the texture images on the mesh. An optimal disk-shaped subdivision of the surface mesh is built in [20] by taking into account mesh topology and available texture images. The aim is to reduce the possible texture distortion inherent in any *image to curved mesh* mapping. Optimal circular texture patches are then resampled for each one of these mesh patches. A disadvantage of these approaches is the possible texture degradation introduced either by complete texture blending [21] or by extensive texture resampling [20]. Conversely, because

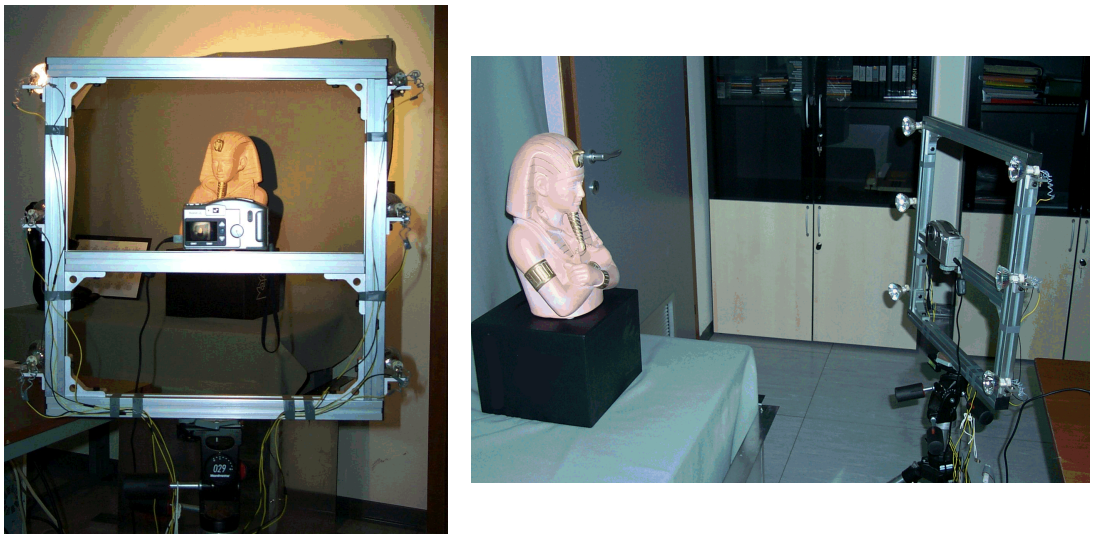

Figure 2: The acquisition instrumentation: a set of computer-driven lights is positioned around the digital camera, in fixed locations.

in our approach resampling is limited to the frontier faces alone (see Section 6.3), our output textures should be less blurred (an empirical comparison is unfortunately not possible).

3 Acquisition of surface attributes

In this phase multiple images are shot to acquire the surface attributes of the sample object from different view points. Images can be acquired with any digital device. We used a high-resolution digital still camera which can be directly connected to a PC. Computer-based control of the camera is crucial to drive a set of point light sources and to synchronize the lighting and multiple camera shots for each view (see Figure 2). A tripod is essential to take multiple images of each view (to ensure pixel correspondence). Reference color sheets can be used in the scene to run a color calibration filter on the acquired images.
The set of viewpoints has to be defined such that each portion of the sample object is visible at least from one viewpoint. If the images are post-processed on line, then the *Patching&Stitching* software module (see Subsection 6.1) can aid the user by highlighting those regions of the object surface that are not visible in any of the selected views, in order to guide the selection of a "complete" set of views.

4 Un-shading, or computing illumination-invariant images

Removing illumination effects from the observed color is crucial if we want to obtain high quality surface attribute detail: we should avoid assigning to a given point of the mesh a color which depends on the light direction at acquisition time. Therefore, we need to compute the illumination-invariant color of each surface parcel, hereafter *un-shaded* color.
The main shading effects that we want to remove are: *direct shading*, i.e. the color intensity variation

which depends on the incidence angle between the light direction and the object surface normal; *cast shadows*; and *specular highlights*, which dramatically affect the appearance of shiny materials (e.g. all vases or surfaces finished with glossy paint or wax). We do not consider here possible *inter-object reflections*, which might also affect the acquired color [35]. The approach we use is simpler than that presented in [35], but sufficiently accurate because we can work in a more controlled laboratory setting, since our target objects are small or medium sized and are generally stored in indoor locations (e.g. museums or labs).

The above three lighting effects can be removed by adopting a method based on the acquisition of multiple images [14, 26, 28], each characterized by different lighting. Following [26], images are taken under different lighting conditions by turning on one of the six point light sources. The relative positions of the light and the camera are fixed (see our acquisition instrumentation in Figure 2), and are computed by an automatic image-based calibration procedure executed once in a pre-processing phase.

An empirical method to remove highlights and shadows is to observe the different colors associated with the same pixel in different images. If a pixel has a much lower intensity in one image then it might correspond to a shadow area; a pixel value which has a very different saturation than the others could correspond to a specular highlight. Therefore, for each corresponding set of pixels, we discard those values which are excessively dark or unsaturated[2].

Because the diffuse reflection of a Lambertian surface in a point p is proportional to the dot-product of the unit length normal vector $\vec{n}$ on p and the direction $\vec{l}$ of the current light source incident on p, we can reconstruct the diffuse reflection term k_d of each surface point p from only three images (same view - different lighting). Let us call c_1, c_2, c_3 the different observed color intensities of a given pixel in these three different images and $\vec{l_1}, \vec{l_2}, \vec{l_3}$ the associated light directions. Then, a linear system of equations $k_d \, \vec{l_i} \cdot \vec{n} = c_i$ is defined with the unknown variables being the normalized vector $\vec{n}$ and the coefficient k_d. If after removing highlight or shadow values we have exactly three remaining values, then we solve the linear system above. If we have more than three valid values, we apply the *least squares* method to solve an over-constrained linear system (and the estimation is more robust). The results are values for k_d and $\vec{n}$ for each surface parcel represented by an image pixel. If the color detail is acquired with a sampling resolution higher than the one used for shape acquisition, these normal displacements can be stored into a bump map that can be used at rendering time to enhance the shape representation.
On the other hand, with less than three valid color values per sampled pixel we have insufficient infor-

[2]Using thresholds for discarding highlight or shadow values we apply a binary classification to the pixels in the image (where pixels weighted zero are not considered in the further un-shading computations). This has the disadvantage of producing haloed images (e.g a circular halo around any highlight spot). Conversely, we perform a smooth transition by assigning a weight $w_{i,j}$ to each pixel $p_{i,j}$, based on the following equation:

$$w_{i,j} = \begin{cases} 0 & : \quad p_{i,j} < t_s - \delta \\ \frac{p_{i,j} - (t_s - \delta)}{2\delta} & : \quad t_s - \delta \leq p_{i,j} < t_s - \delta \\ 1 & : \quad p_{i,j} > t_s + \delta \end{cases}$$

where, as an example, t_s is the shadow threshold.

Figure 3: Results of the *un-shading* phase: an image with highlight and shading is on the left, and the corresponding un-shaded image is on the right.

mation to compute its diffuse reflection term and normal vector. In our experiments the percentage of under-sampled pixels is low (generally less than 50 pixels in images with more than 250K pixels) and these pixels are estimated by interpolating the color and normal vectors of the adjacent pixels. The artefacts introduced are not easily perceptible.
The result of this phase is an *un-shaded* image for each view, plus an associated *bump texture*. An example is shown in Figure 3, where one of the six images acquired from the real object is shown on the left, and the corresponding *un-shaded* texture is on the right.

The inverse illumination computation is sensitive to possible noise contained in the acquired images. To reduce the possible aliasing due to noise, we apply a smoothing filter to the images to remove noise peaks. Moreover, computation precision is higher if the initial observed intensities are defined with a dynamic range wider than 8 bits, and linearly reflect the effective radiance of the observed surface. To obtain an accurate measure of the radiance we adopted the approach in [9], which inverts the non-linear mapping of the camera by acquiring and processing multiple exposures at different shutter speeds. Therefore, we processed six high dynamic range images for each view.

5 Image Registration

The following step registers all of the images with respect to the 3D digital model. To compute all the camera parameters that affect the current image (the roto-translation matrix, the perspective projection matrix and the radial distortion of the camera lenses) we adopted a classical approach based on the

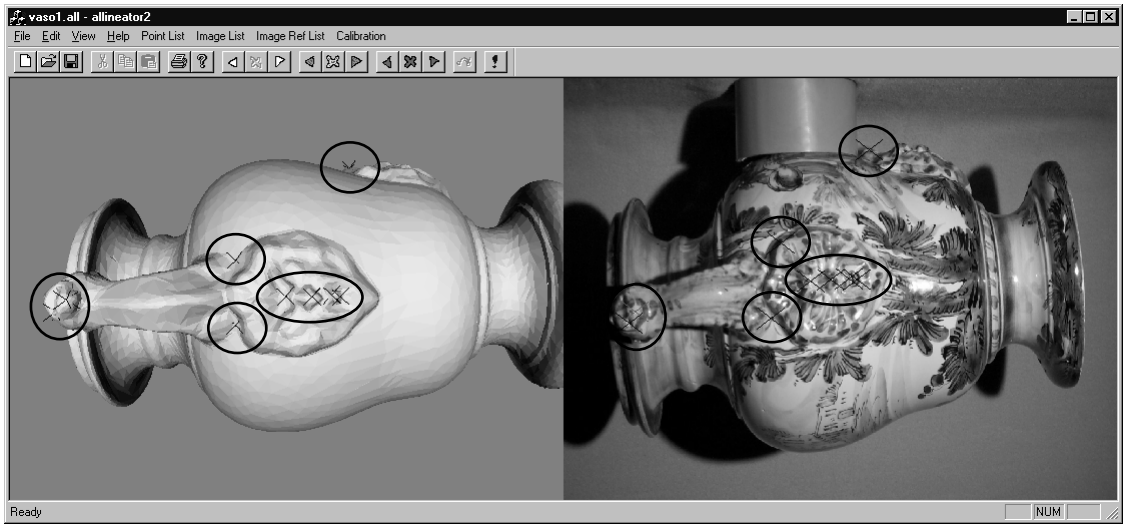

Figure 4: The simple interactive tool for the selection of matching pairs: the synthetic mesh is on the left and one of the acquired images is on the right; the crosses mark the current point pairs selected by the user (they are highlighted in the figure by circles, which are not part of the GUI).

interactive selection of six corresponding non-coplanar point pairs [33]. We developed a simple user-assisted tool to manage matching point pair selection (see Figure 4). For each image, the user must select at least six pixels, and for each of these pixels he also picks the corresponding point on the 3D mesh (rendered in the left-most area, see Figure 4). Obviously, the more corresponding pairs we select, the more precise and stable the camera parameters computed; in reality, we selected around 10 corresponding pairs. Moreover, the selection accuracy is improved by the tool capability of zooming the image and zooming/panning/rotating the 3D mesh.

Other registration methods have been proposed [3, 13, 32, 17]. One approach might be to derive camera parameters from the position in the images of some markers present in the scene [32]. Camera parameters can be computed completely automatically if three (or more) markers are visible in the image. There are two methods to insert markers in the scene: pasting markers on the object surface, or onto the space around the object. If the goal is to acquire the color attribute, pasting markers on the object surface has the disadvantage of losing the detail covered by the marker (and marker area should generally be not too small so it can be easily detected and centered). If, on the other hand, we distribute the markers in the space around the object, we do not lose detail but introduce an inconvenient constraint: we can no longer move the object, its position with respect to the markers has to remain fixed for the entire acquisition phase. For this reason we did not adopt an approach based on the intrusive introduction of markers.

A pair-matching approach was also adopted in [21] for computing a first, rough registration of still images on the 3D mesh. The rough registration is then followed by a further refinement based on the minimization of different objective functions, i.e. performing the registration of the silhouettes on the image and the 3D model and the registration of corresponding features detected by extracting features on the im-

ages. The approach proposed in [21] therefore produces in a single process a global registration of all the images on the 3D mesh. In our approach, on the other hand, the initial per-image rough registration is then followed by a *local* registration step (described in Subsection 6), which is operated independently on subsections of each image (the ones where different images have to be merged) and is extremely fast. Finally, a completely automatic registration approach has been recently proposed in [17], based on the completely automatic comparison of the silhouettes extracted from the textured image and from a rendered image of the 3D mesh. The advantage of this approach is that registration becomes completely automatic, but it introduces a constraint: to allow the method to converge without any user intervention, every image has to contain a complete view of the object. Because the image to geometry registration is the only user assisted phase in our overall pipeline, an automatic registration approach such as the one proposed in [17] could be integrated into our pipeline, to be used when the above constraint is fulfilled (e.g. for all objects of a sufficiently small size).

6 Texture Stitching and Blending

In this phase we build an optimal patchwork of image patches to be mapped onto the mesh, such that all of the object surface is covered and adjacent image patches join smoothly (both in terms of chromatic likeness and continuity of texture features). The texture stitching and blending process works on the mesh vertices and returns a textured triangle mesh. It consists of four sub-phases.

6.1 *Vertex–to–image* binding

The goal of this phase is to assign to each mesh vertex v both a *valid image set* and a *target image*, given the set of input images. The *valid image set* for a vertex v is defined as the subset of images $I_v = \{i_k\}$ such that v is *visible* in i_k and is a *non-silhouette* vertex.

To verify if a vertex v is **visible** in a given image i_k we must check that: (a) the projection of v on the image plane is contained in the image i_k; (b) the normal vector in v is directed towards the viewpoint (i.e. the angle between the normal and the view direction has to be lower than $\pi/2$); and (c) there are no other intersections of the mesh with the line that connects v and the viewpoint. Possible mesh intersections on the line of sight are tested in the current implementation by adopting a ray casting approach, accelerated by means of a uniform grid data structure [11]. A hardware-accelerated z-buffer solution could also be used.

Moreover, vertices are classified with respect to a given image as **silhouette** and **non-silhouette** vertices. In the first case, at least one face with an inverse orientation with respect to the view point needs to exist in the fan of triangles that are incident in v (see the vertices labeled "A" in Figure 5). We avoid including in the *valid set* an image in which v is classified as a silhouette vertex because possible small registration errors on v will be easily noticeable.

If at the end of the process we find a subset of vertices whose valid image sets are empty, then the *vertex–to–image binding* module will request that the user takes some more images, showing the area that

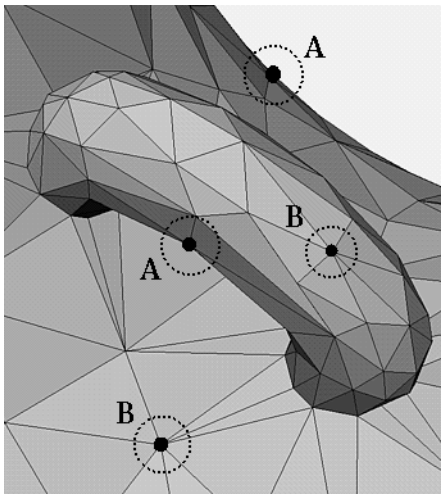

Figure 5: Classification of the mesh vertices with respect to a given image i_k: (A) silhouette vertices, (B) non-silhouette vertices.

has not been sampled. If, conversely, we have multiple images in the valid set of a vertex v, the *target image* is initially selected such that the angle between the surface normal in v and the view direction is the smallest one, that is the image showing the lowest projective distortion with respect to a small mesh area centered in v. An example of the texture distortion introduced while mapping an image to a nearly orthogonal mesh section is shown in Figure 6.

6.2 Patch growing

The previous *vertex-to-image* binding phase considers mainly vertices (geometry and normals). We now start to look at faces and topology. Mesh faces are classified and mapped to a given texture as follows:

- if the three vertices of a triangular face f are all linked to the same target image i_k, then face f is mapped on image i_k with texture coordinates equal to the projection of its vertices on i_k; this face is called **internal**;
- if, conversely, the vertices of f are linked to two (or even three) different target images, then face f is classified as a **frontier** one.

Because some aliasing might be introduced while resampling texture chunks for the *frontier* faces (compositing multiple images might introduce some degradation due to possible misalignment, see Subsection 6.3), we need to minimize the percentage of frontier faces. The goal is therefore to produce a mapping where contiguous sections of the mesh will be mapped, as much as possible, on contiguous patches of the same image.

A greedy iterative algorithm is therefore applied that analyzes each single vertex and changes the target image association (i.e. the *vertex-to-image* link) if this update reduces the overall number of frontier faces. During the iterative process a vertex can change its target image many times, until we get a minimum

Figure 6: An example of two *valid images*, (a) and (b); if we map image (b) on the mesh and render a synthetic image (b_1) using the same viewpoint of image (a), then we can see how poor and distorted the detail is in the right-most side of the mesh; obviously, mapping image (a) on this mesh section can give a much better local representation of the detail.

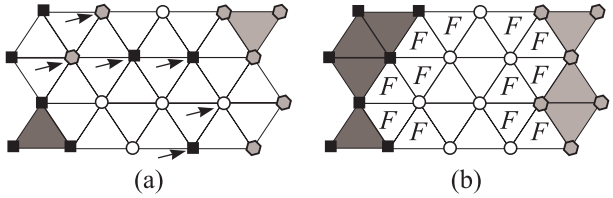

Figure 7: Iterative local optimization of texture coverage: in the sample drawing, vertices are initially assigned to three target images (represented by a hexagon, a square and a circle). Then, we select a set of frontier vertices (indicated with arrows) and change their target images, obtaining configuration (b), which now corresponds to a local minimum. Frontier faces are indicated with an *"F"* in (b).

(see Figure 7). A similar approach was adopted in [29] to optimize face–to–image mapping, but it is based on a simpler single-step evaluation phase. An example of the results produced by our multi-step greedy solution is shown in Figure 8. The few small isolated red regions in the figure remain after patch growing because, in this particular case, they are visible only from the red-coded image.

6.3 Patch boundary blending

View-dependent texture mapping approaches blend multiple weighted textures at rendering time [8, 25]. On the other hand, we restrict texture resampling or blending only to the sections corresponding to *frontier* faces. For each *frontier* face we can resample a triangular texture patch as a weighted composition (or cross-fading) of the corresponding triangular sections of the target images associated with the vertices. Let us suppose that three target images i_a, i_b, i_c are associated, respectively, with vertices v_1, v_2, v_3 of a frontier face f. For each vertex of f we have the corresponding texture coordinates of the three target

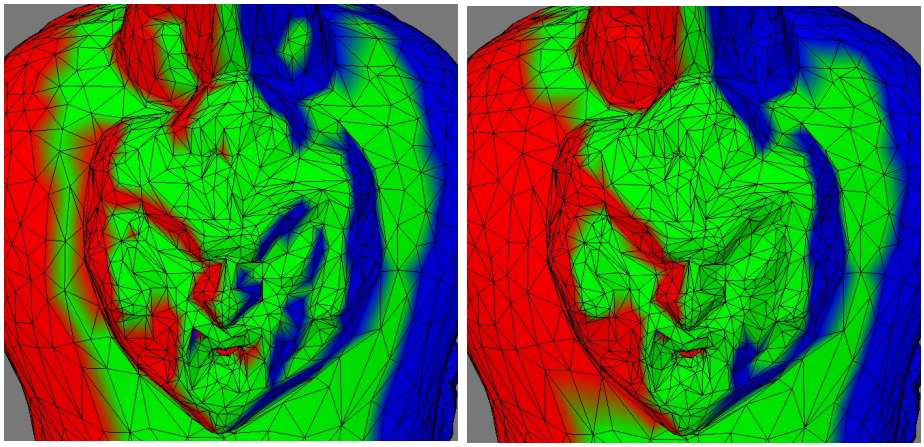

Figure 8: An example of optimized frontier face management: we have 1,137 frontier faces out of a total of 10,600 in the initial configuration (on the left); we get only 790 frontier faces after optimization (on the right).

images. To build the corresponding triangular texture patch we first have to decide its resolution (in texel units); this depends on the corresponding resolution of the target image sections, to avoid losing information while blending the images. Then, the frontier face f is scan-converted (using the above sampling step). For each sampling point p produced by the scan conversion, represented by its baricentric coordinates: $p = \alpha\ v_1 + \beta\ v_2 + \gamma\ v_3$, with $\alpha + \beta + \gamma = 1$, we determine the corresponding three color values $i_a[\bar{p_a}], i_b[\bar{p_b}], i_c[\bar{p_c}]$ in the three target images. The color c_p to be assigned to p (and stored in the texture patch) is therefore computed as a weighted composition of these three values:

$$c_p = \alpha\ i_a[\bar{p_a}] + \beta\ i_b[\bar{p_b}] + \gamma\ i_c[\bar{p_c}] \ . \tag{1}$$

The sampling point p might not be visible on one of the target images, due to a possible intersection of the mesh M with the line of sight. In this case, the color contained in this image is relative to another section of the mesh and therefore the contribution of this image is not added (i.e. the corresponding barycentric coordinate is set to zero). Visibility is therefore checked for each sampling point and each target image, by tracing rays from p to the viewpoints.

The detection of the triangular section in each target image which corresponds to a frontier face is critical, because we do not want to introduce discontinuity in the representation of detail. Insufficient accuracy of the *Image Registration* phase might therefore cause some aliasing. Image registration might not be sufficiently accurate, due to: (a) an imprecise selection of the corresponding point pairs; (b) the use of a simplified camera model that does not take into account all the possible non-radial distortions of the real camera lenses; (c) the use of finite numeric precision in the computations of the camera parameters. These misalignments may produce ghost effects in the resampled triangular texture patches. Ghost effects are more easily perceivable if the textures contain very thin lines or abrupt changes in color information (see an example in Figure 9). The vase used in our test is a very challenging object, because the pictorial detail consists of many thin painted lines. To drastically reduce this possible aliasing we

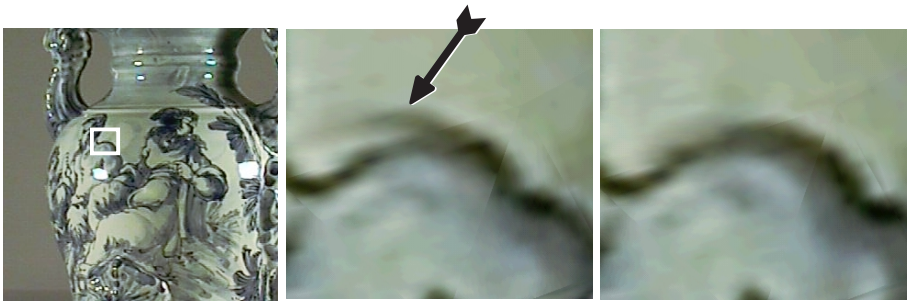

Figure 9: Local registration phase: the section of the vase surface marked in image (a) is represented (magnified) in images (b) and (c); an example of the ghost effect generated by a slight misalignment is shown in the composite texture section shown in (b), while the more precise texture mapping generated after *local registration* is shown in (c).

apply a ***local registration*** step which works on each single vertex of the frontier faces (in contrast to the *Image Registration* phase, which works on the entire image). The goal is now to *locally* remove those small misalignments that may be introduced during the different phases of our overall acquisition pipeline. Our solution is completely automatic and adopts an image-processing approach (see Figure 10).

One common approach to reduce the possible misalignments is to select corresponding points (e.g. points in the proximity of color discontinuity) on pairs of overlapping texture images (which have to be projected onto a common plane). We can then evaluate the rigid transformation which optimizes the alignment of the selected point pairs. An example is the Pulli's approach [24] to compute a global transformation between pairs of range+intensity images. A slightly more complex solution was proposed by Shum and Szelinski to manage the registration of multiple images for the construction of panoramic mosaics [30]. They propose a local alignment solution which removes ghosting effects by dividing the images into patches, aligning the center points of corresponding patches on different images, and then warping the images.

Our solution, conversely, is *mesh-centered*, in the sense that we start from the geometric model and limit the texture warping and blending to the texture section associated with the frontier faces alone. By definition, each frontier face is mapped to [at most] three target images. For each frontier face we compute an independent rigid transformation of the texture mapping coordinates of its vertices. These rigid transformations are evaluated by aligning, for each vertex p, the at most three corresponding small texture parcels centered on the (u_p, v_p) texture coordinate of p on each target image. Let us see in detail how our approach works.

For each frontier face f of the mesh we have: three target images i_1, i_2, i_3 associated with the three vertices v_1, v_2, v_3 of f; and nine texture coordinates $t_{hk} \in \mathbf{R}^2$, where t_{hk} is the texture coordinate relative to the h-th vertex on the k-th image. For each image i_k we know the projection function $\mathcal{P}_k : \mathbf{R}^3 \rightarrow \mathbf{R}^2$ which computes the texture coordinate corresponding to each mesh point p (clearly $t_{hk} = \mathcal{P}_k(v_h)$).

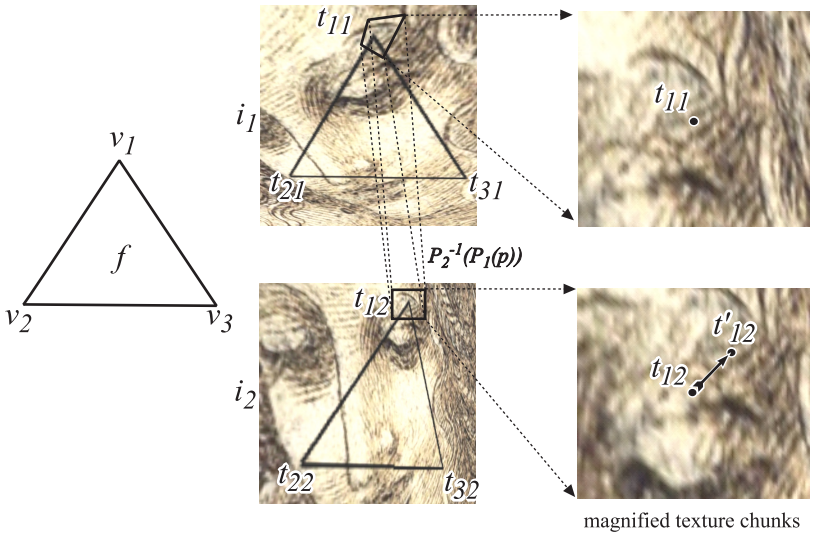

Figure 10: Local registration of the texture coordinates of a frontier face vertices.

The *local registration* processes the vertices of face f one at a time. It fixes the texture coordinates from the given vertex v_i to its target image (e.g. vertex v_1 and texture coordinates t_{11} in Figure 10). Given vertex v_1 of f, it computes an optimal translation of the texture coordinates t_{12} and t_{13} of the same vertex on the other two images. The search range for this translation depends on a global parameter, selected by the user according to: the mean size of the mesh faces, the resolution in pixels of the input texture images and in some cases the visual evaluation of intermediate results. Each translation is computed by independently comparing small corresponding sections of the texture images. For example, to compute the optimal translation of t_{12}, we select a rectangular section of i_2 centered on point t_{12} (see Figure 10). The size of this region is around three times the size of the search range. For each texel in this rectangular region, we compute the corresponding texel in i_1 (by projecting it firstly onto mesh M using the inverse projection transformation $\mathcal{P}_2^{-1}$ associated with i_2, and then projecting the obtained 3D point again to the plane of i_1 using the associated projective mapping $\mathcal{P}_1$). Now we have two corresponding texture subsections, immersed in the same space as i_2, which can be aligned by maximizing their cross-correlation [4]. This process is iterated for each vertex, and returns six new texture mapping coordinates for the vertices of the given face. An example of the improved texture obtained after local registration is shown in Figures 9 and 12. Note that even though we compute a simple translation of each vertex's texture coordinates, the result of the application of several translations onto a set of nearby vertices can also correct empirically a slight rotation or scaling error introduced in the *Image Registration* phase. An example is shown in Figure 11.

Clearly, if there are no features in the image sections considered then the *local registration* returns an identity transformation (but this is not a problem, because if the images have no discontinuity or features of interest, then the local registration is not needed at all).

Using resampled triangular texture patches has a disadvantage: we have to pack them in a global

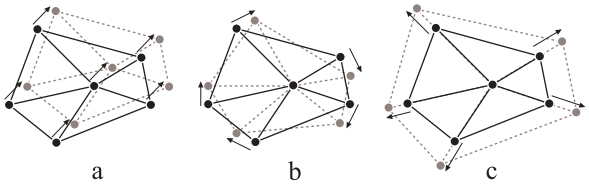

Figure 11: The combination of independent vertex translations may result in *warping* the corresponding target image section.

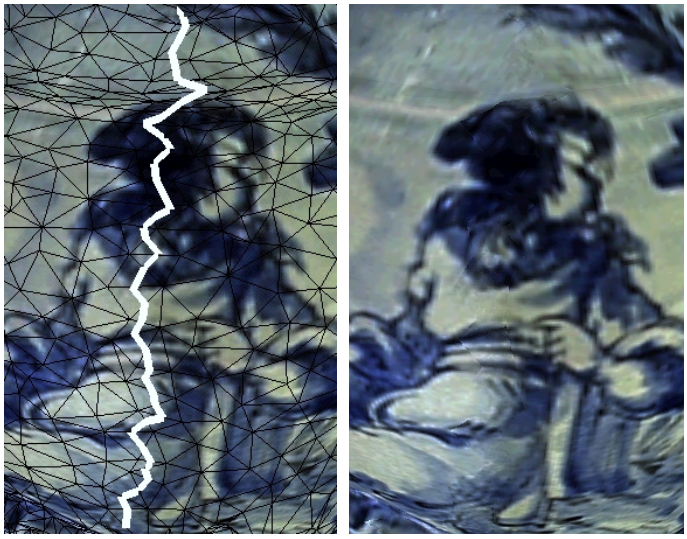

Figure 12: As an example, a strip of frontier faces is marked by the thick white polyline (on the left); the same mesh section is rendered on the right.

texture map (see the following section), and then we cannot use the standard texture mip-map approach to produce a hierarchy of textures to render the mesh according to the distances from the observer. This is because packing texture patches implies that adjacent texels do not always correspond to adjacent surface parcels. To avoid this problem, we can also manage texture reconstruction in a different way. Instead of explicitly resampling a triangular texture patch for each frontier face, we can duplicate (or triplicate) the corresponding face in the mesh, assigning to each of these instances one of the corresponding texture images and blending these textures at rendering time by setting vertex opacities according to Equation 1. To assign different texture mapping coordinates and opacities to the vertices of the replicated boundary faces, we need to duplicate the vertices as well. The opacity of a boundary vertex is set to completely opaque if this instance of the vertex is mapped to the vertex's target image; it is completely transparent otherwise. Explicit texture resampling is thus avoided, with slight overheads in rendering time and some restructuring of the mesh topology needed. All frontier faces have to be rendered many times, but these faces are generally a small fraction of the total (around 7% in the case of the *vase* mesh) and therefore the overheads can be considered negligible. In this case the output mesh structure slightly changes with respect to the input one because we introduce some topologically open boundaries in the mesh, and

Figure 13: On the left are shown (rendered wire-frame) the faces of M which are linked to a particular input image; in this case the corresponding texture section has an elongated shape, which can lead to some space overheads in the final texture T_M (shown on the right).

some overlapped components which are perfectly identical from the perspective of the geometry but are topologically distinct. If the main use of the attribute-enhanced mesh is not pure visualization, this could be a problem.

6.4 Texture-patches packing

Let us consider first the resampling-based approach described in the previous section to manage frontier faces. In this last sub-phase, all the target image sub-sections (i.e. the areas of the initial images that have faces mapped onto them) and the set of resampled triangular texture patches are packed into a standard texture map T_M (or a few if the amount of texture data is larger than the maximal texture size of the current rendering architecture). Texture coordinates in the triangle mesh are consequently updated.

The first step is to extract from each target image the polygonal sections which have been linked to some faces of M. To simplify texture packing, we extract from each target image the minimal *rectangular* texture area that contains each needed section (see an example in Figure 13, lefthand). If two rectangular areas on the same target image have a non-empty intersection, then we decide to store in T_M the bounding rectangle of their union *iff* the area of the union is smaller than the sum of the two areas. Triangular texture patches are managed by joining pairs of patches to form rectangular areas (obviously, by joining pairs of faces which have an identical or similar size). Then, all the rectangular texture sections are packed in a single texture T_M (see Figure 13) by using a *cutting-stock* algorithm [2]. Although more sophisticated solutions exist in the literature regarding the polygonal area packing problem, this packing solution has been chosen because it is simple to implement and reasonably efficient (at least, when the number of rectangular texture sections is not huge, as is the case of the meshes used in our experiments).

Once packed, a triangular texture clearly has adjacent patches in the packed texture which are probably different from the ones assigned to the geo-topologically adjacent faces. A packed texture therefore might not correctly filter across edges which are shared among all those triangle pairs that are now dis-

joint in the packed texture. But this problem can be simply overcome by an opportune sampling of each frontier face f: the associated triangular texture patch t_f can be resampled a few pixels larger than the required minimal size [7]. At rendering time the texture coordinates for each scan-converted pixel $p_i \in f$ are thus always inside the associated discrete texture area t_f.

On PC platforms, texture size has an upper bound of 1024x1024 texels (and each texture side has to be of size $2^n * 2^n$). A large texture map (which could be rendered as it is on SGI platforms) thus need to be split into chunks which satisfy the maximal size limit. A straightforward way to solve this problem is to produce multiple textures which contain all the data originally in T_M. This requires a simple modification of the packing algorithm (which now has to fill k textures in a pseudo-optimal manner).

The texture reconstruction and packing is slightly different if we choose to avoid texture resampling, and implement texture blending at rendering time (see previous section) to support easy mip-mapping of the final texture produced. In this case, we do not have resampled triangular textures, and packing is applied to the selected rectangular subregions of the input textures. We add one more constraint, to allow easy and correct filtering of k levels of lower resolution textures from the packed one: the active subregions cropped from the initial textures should have side lenght multiple of 2^k texels; active regions are therefore slightly enlarged to fulfill the above constraint.

Obviously, the technique used to produce the resulting RGB texture is also used to build a *bump-texture* from the initial normal maps produced by the un-shading phase.

A potential problem that can arise when mapping images taken from reality on a 3D geometry is the *perspective texture distortion* that may arise. This is because we have a non-linear perspective transformation and a corresponding bi-linear texture interpolation (according to most hardware and software texture mapping systems). This means that without *image rectification* of the texture (taking into account the respective Z coordinates), textures mapped on a rendered face in generally appear to bedistorted. The need to rectify textures to remove distortion also makes it difficult to build a merged texture from multiple images, as pointed out in other papers.
In our case, not performing rectification has a limited impact on visual quality because: (a) distortion decreases as the image viewpoint is closer to being orthogonal to the triangle face, and we generally choose one of the most orthogonal images for each triangle; (b) if the mesh is described by small triangles (in texel units) then the distortion is really small. Moreover, note that some recent hardware and API's allow the use of perspective texture coordinates (u, v, w). These could be used to remove all the inherent distortion in an image-based texture map, using the camera distance to the vertex as the third w coordinate.

7 Evaluation of results

Here we report the results obtained for two target objects. The first one was a ceramic *vase* of Albissola, with a height of around 40 cm. and a very complex painted surface (mostly thin painted lines). The *vase* geometry was acquired with a Cyberware 3030 range scanner. The mesh obtained originally consisted of 2M faces, and it was simplified down to 20,000 faces using the Jade 2.0 simplification code [5] with an accuracy of 0.17 mm. The second object was a small Capodimonte *statuette* (around 25 cm. tall), acquired using a CT scanner. The geometry, reconstructed via iso-surface fitting on the volume model, was around 1.5M faces, and was simplified down to 10,000 faces. In both cases, surface attributes registration and stitching were performed on the simplified models.

Texture images were acquired by using a KODAK DC290, a commercial digital still camera which can be driven by software and supports a sufficiently high acquisition resolution (up to 1792x1200) in true color. The detail of the *statuette*, which has a rather complex shape, was acquired by taking 14 different views, with six different lighting conditions per view. Conversely, only eight views were sufficient for the *vase*.

All images were processed on a 450MHz Pentium II PC. Data processing times of the initial *Image Registration* were obviously user-dependent. Each view was registered in less than three minutes (approximately). The *un-shading* phase took 32 seconds for each view (i.e. for each set of six images taken from the same view). After registration and un-shading of the input images, the automatic texture integration and patching process running time depends on the number of total images and on the complexity of the object shape (i.e. the number of faces and, to a lesser extent, the complexity of the mesh topology). The running times needed to process the two test objects were 89 seconds for the *vase* and 62 seconds for the *statuette*.

The detail texture produced for the *vase*, using the texture-resampling approach for the frontier faces, is shown in Figure 13. Three synthetic images of the *vase* mesh are in Figure 15, upper-most section; these images have been produced by using an interactive renderer supporting a basic local illumination model. A more sophisticated rendering is shown in the same figure (bottom-most picture), obtained using a commercial photo-realistic rendering system; the subtle differences in color hue are due to the very different lighting models used in the two different rendering tools.

A visual comparison between a synthetic rendering and a photographic image of the *statuette* object is in Figure 14.

The approach presented produces some good results, though not completely free from aliasing. The overall accuracy of our image-to-geometry mapping depends on multiple variables: the accuracy of the digital 3D model used (in our case, a simplified 3D scanned mesh); the accuracy of the initial image registration (i.e. the camera calibration procedure [33]); the accuracy of the local registration step performed on selected vertices of the 3D mesh; and finally, the visual accuracy produced by the texture blending approach. Tsai's registration method is very precise, but due to local inaccuracies of the 3D model or to some insufficiently accurate inputs given by the user (see the user-assisted selection of

corresponding pairs described in Section 5), the result of the initial registration could be slightly imprecise on selected locations. This is exactly why there needs to be a second registration step, the local one (see subsection 6.3), which allows the local correction of these potential small misalignment errors.

8 Conclusions

In this paper we have proposed a system for the semi-automatic acquisition of the surface attributes of 3D free-form objects, and its integration with the 3D shape through standard texture-mapping. In particular, surface attribute detail observed in a small set of photographic images is registered and patched onto the input mesh (acquired with any type of automatic acquisition methodology). The observed images are processed to remove lighting effects (shading, highlights and shadows), and the resulting *un-shaded* textures are then smoothly joined onto a standard texture map. The advantages of this process are the low user-intervention required (limited to the selection of the view set and to the initial rough registration) and the high quality results obtained, due to the innovative texture patching process proposed. In particular, the global texture is produced as a patchwork of un-shaded textures [or bump-textures] sections and gives a very precise representation of the observed detail due to the limited texture resampling (performed only on frontier faces) and the image-based local registration (which removes possible discontinuities).
Some of the methods proposed are now being adopted in the design and implementation of a low-cost 3D scanner based on structured light projection and distorted pattern processing.

Below are some possible extensions.

Given the high resolution given by current off-the-shelf digital cameras, packing all the texture detail in a single texture leads to huge textures. A better option is to divide the texture data into multiple images. In our case the problem has a rather simple, though non-optimal, solution. In the more general case the "skinning" problem (given a topologically complex mesh, produce a mapping of its faces to multiple planar textures which [hopefully] preserves topological adjacency) is a rather complex task [20, 23] which deserves further research.

In our approach, slight registration error or chromatic variations between different images are smoothed by cross-fading the texture images associated with each border face. Obviously, image fading depends on the relative size of the frontier faces with respect to the overall mesh size, and it may be insufficient in the case of high resolution meshes. The width of the frontier strip might be widened to produce a smoother join. Strip optimal width may depend on mesh characteristics (e.g. never smaller than a percentage of the mesh bounding box diameter) or on the characteristics of the corresponding adjacent target images (e.g. strip width may depend on the chromatic difference between the adjacent target images). The interpolation factor used to fade different adjacent images should be proportional to the geodesic distance [16] from the medial axis of the initial frontier strip.

A more complete evaluation of the object's reflectance properties should be performed, with the aim of acquiring a more accurate approximation of the surface BRDF. But this was not the aim of this paper. Although some more sophisticated techniques exist [29, 15, 34], acquiring the reflectance properties of a

complex object can be still considered an open issue.

Acknowledgements

This work was partially financed by the *Progetto Finalizzato Beni Culturali* of the Italian National Research Council (CNR). Special thanks to Giovanni Braccini and Simona del Corona for the tomographic acquisition of the Capodimonte statuette.

References

[1] R. Baribeau, M. Rioux, and G. Godin. Color reflectance modeling using a polychromatic laser sensor. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):263–269, 1992.

[2] J.E. Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. *Operation Research*, 33(1):49–64, 1985.

[3] R.G. Bogart. View correlation. In J. Arvo, editor, *Graphics Gems II*, pages 181–190. Academic Press, 1991.

[4] L. Gottesfeld Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, Dec. 1992.

[5] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *The Visual Computer*, 13(5):228–246, June 1997.

[6] P. Cignoni, C. Montani, C. Rocchini, and R. Scopigno. Multiple textures stitching and blending on 3d objects. In G. Ward and D. Lischinsky, editors, *10th Eurographics Workshop on Rendering (Granada, E, June 21-23)*, pages 127–138. SpringerVerlag, 1999.

[7] P. Cignoni, C. Montani, C. Rocchini, R. Scopigno, and M. Tarini. Preserving attribute values on simplified meshes by re-sampling detail textures. *The Visual Computer*, 15(10):519–539, 1999. (short version appeared in IEEE Visualization '98 Proceedings).

[8] P. Debevec, Y. Yu, and G. Borsukov. Efficient view-dependent image-based rendering with projective texture-mapping. In G. Drettakis and N. Max, editors, *Rendering Techniques '98*, page 14. Springer Wien, 1998.

[9] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In Turner Whitted, editor, *Comp. Graph. Proc., Annual Conf. Series (Siggraph '97)*, pages 369–378. ACM Siggraph, Aug. 1997.

[10] P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 11–20. ACM SIGGRAPH, Addison Wesley, August 1996.

[11] W.R. Franklin, N. Chandrasekhar, M. Kankanhalli, M. Seshan, and V. Akman. Efficiency of uniform grids for intersection detection on serial and parallel machines. In *New Trends in Computer Graphics – CGI '88*, pages 288–297, Geneva, Switzerland, 1988. Springer-Verlag.

[12] M. Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 209–216. Addison Wesley, August 1997.

[13] M. Gleicher and A. Witkin. Through-the-lens camera control. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):331–340, July 1992.

[14] G. Kay and T. Caelli. Inverting an illumination model from range and intensity maps. *Computer Vision, Graphics, and Image Processing. Image Understanding*, 59(2):183–201, March 1994.

[15] E.P.F. Lafortune, S.C. Foo, K.E. Torrance, and D.P. Greenberg. Non-linear approximation of reflectance functions. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 117–126. ACM SIGGRAPH, Addison Wesley, August 1997.

[16] M. Lanthier, A. Maheshwari, and J.R. Sack. Approximating weighted shortest paths on polyhedral surfaces. In *6th Annual Video Review of Computational Geometry, Proc. 13th ACM Symp. Computational Geometry*, pages 485–486. ACM Press, 4–6 June 1997.

[17] H.P. Lensch, W. Heidrich, and H.P. Seidel. Automated texture registration and stitching for real world models. In *Proc. Pacific Graphics 2000 Conf.*, page (in press), 2000.

[18] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The Digital Michelangelo Project: 3D scanning of large statues. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '00)*, pages 131–144. ACM SIGGRAPH, Addison Wesley, July 24-28 2000.

[19] J. Lu and J. Little. Reflectance function estimation and shape recovery from image sequence of a rotating object. In *Proc. of International Conference on Computer Vision*, pages 80–86, 1995.

[20] S.R. Marshner. *Inverse rendering for computer graphics*. PhD thesis, Cornelle University, 1998.

[21] Peter J. Neugebauer and Konrad Klein. Texturing 3d models of real worls objects from multiple unregistered photographic views. *Computer Graphics Forum (Eurographics'99 Proc.)*, 18(3):245–255, 1999.

[22] M. Petrov, A. Talapov, T. Robertson, A. Lebedev, A. Zhilyaev, and L. Polonsky. Optical 3D digitizers: Bringing life to the virtual world. *IEEE Computer Graphics & Applications*, 18(3):28–37, May – June 1998.

[23] D. Piponi and G. Borshukov. Seamless texture mapping of subdivision surfaces by modeling pelting and texture blending. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, Annual Conference Series, pages 471–478. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[24] K. Pulli. *Surface Reconstruction and Display from range and color data*. PhD thesis, Dept. of Computer Science and Engeneering, University of Washington, 1997.

[25] K. Pulli, M. Cohen, T. Duchamp, H.Hoppe, L. Shapiro, and W. Stuetzle. View-based rendering: Visualizing real objects from scanned range and color data. In Julie Dorsey and Philipp Slusallek, editors, *Eurographics Rendering Workshop 1997*, pages 23–34. Springer Wien, June 1997. ISBN 3-211-83001-4.

[26] H. Rushmeier and F. Bernardini. Computing consistent normals and colors from photometric data. In *Proc. Second Int. Conf. on 3D Digital Imaging and Modeling*, pages 99–108, Ottawa, Canada, 1999.

[27] H. Rushmeier, F. Bernardini, J. Mittleman, and G. Taubin. Acquiring input for rendering at appropriate levels of detail: digitizing a pietá. In G. Drettakis and N. Max, editors, *Eurographics Rendering Workshop 1998*. Springer Wien, June 1998.

[28] Y. Sato and K. Ikeuchi. Reflectance analysis for 3D computer graphics model generation. *Graphical models and image processing: GMIP*, 58(5):437–451, September 1996.

[29] Y. Sato, M.D. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. In Turner Whitted, editor, *Comp. Graph. Proc., Annual Conf. Series (Siggraph '97)*, pages 379–388. ACM Siggraph, Aug. 1997.

Figure 14: A comparison of the accuracy of the synthetic model (right) *wrt* one of the original object images (left).

[30] H.-Y. Shum and R. Szeliski. Construction and refinement of panoramic mosaics with global and local alignment. In *In Sixth International Conference on Computer Vision (ICCV'98), Bombay*, pages 953–958, 1998.

[31] M. Soucy, G. Godin, R. Baribeau, F. Blais, and M. Rioux. Sensors and algorithms for the construction of digital 3d colour models of real objects. In *Proceedings Intl. Conf. on Image Processing*, pages 409–412, 1996.

[32] A. State, G. Hirota, D. T. Chen, and W. Garrett. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '00)*, pages 429–438. ACM SIGGRAPH, Addison Wesley, 1996.

[33] R. Tsai. A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4), August 1987.

[34] Y. Yu, P. Debevec, J. Malik, and Hawkins T. Inverse global illumination: recovering reflectance models of real scenes from photographs. In A. Rockwood, editor, *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series, pages 215–224. ACM SIGGRAPH, Addison Wesley, July 1999.

[35] Y. Yu and J. Malik. Recovering photometric properties of architectural scenes from photographs. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 207–218. ACM SIGGRAPH, Addison Wesley, July 1998.

Figure 15: Three different views of the resulting *vase* mesh (rendered without shading using a standard OpenGL-based interactive renderer). The image on the bottom is a re-lighted image, obtained with a photo-realistic rendering software.