Multiple Textures Stitching and Blending on 3D Objects

C. Rocchini, P. Cignoni, C. Montani
Istituto di Elaborazione dell'Informazione – C.N.R.
Email: cignoni|montani|rocchini@iei.pi.cnr.it

R. Scopigno
CNUCE – C.N.R.
Email: r.scopigno@cnuce.cnr.it

Abstract. In this paper we propose a new approach for mapping and blending textures on 3D geometries. The system starts from a 3D mesh which represents a real object and improves this model with pictorial detail. Texture detail is acquired via a common photographic process directly from the real object. These images are then registered and stitched on the 3D mesh, by integrating them into a single standard texture map. An optimal correspondence between regions of the 3D mesh and sections of the acquired images is built. Then, a new approach is proposed to produce a smooth join between different images that map on adjacent sections of the surface, based on texture blending. For each mesh face which is on the adjacency border between different observed images, a corresponding triangular texture patch is resampled as a weighted blend of the corresponding adjacent images sections. The accuracy of the resampling and blending process is improved by computing an accurate piecewise local registration of the original images with respect to the current face vertices. Examples of the results obtained with sample Cultural Heritage objects are presented and discussed.

1 Introduction

An accurate digital representation of both the shape and the pictorial detail of 3D objects is mandatory in many applications. An example is certainly the acquisition of 3D objects in the Cultural Heritage field. The accuracy required is so high that standard CAD systems do not produce accurate enough models, and in any case accurate modeling is very time consuming. Automatic acquisition is therefore becoming the solution to the problem, and many different technologies have been proposed [15]. Accuracy and resolution of the available devices are in general very good, and allow the acquisition of highly detailed models.

One critical point is how do we acquire *pictorial detail* and how do we link it to the shape description [1, 11, 12, 9, 19, 20, 18, 16, 17, 26, 13] (hereafter, we assume that shape is represented by a standard triangle mesh encoding). The term *pictorial detail* can be used to represent different concepts: from texture color observed on the object surface, which strongly depends on the lighting conditions, to object surface reflectance properties, computed by removing highlights and diffuse shading or even by computing a bidirectional reflectance distribution function (BRDF).
Texture mapping has been used in graphics as an early approximation of *image-based modeling*. In our case, we want to acquire automatically the pictorial detail of a complex surface, by extracting texture-maps from real world [23] and by pasting them

on standard textured three-dimensional models. The same problem was investigated previously[1] by Soucy et al. [22].
Some problems we may find in current 3D scanning technology, with respect to detail acquisition, are: the existence of devices with no pictorial detail acquisition capability; possible insufficient accuracy or resolution of the pictorial detail acquired; complexity of the integration of multiple scans process, especially if we want the final mesh to integrate smoothly and accurately the acquired pictorial detail [9, 20, 17, 13].
In our approach, shape and pictorial detail acquisition are de-coupled. This to allow the use, out of the many available, of the 3D scanning technology which is more adequate to process the target object. In this paper we focus mainly on the acquisition and management of pictorial detail, and in particular on how we can map and integrate on a standard textured mesh the detail contained in a set of images taken from different view points. Our approach can be classified as a *hybrid image-based modeling* one. The construction of a panoramic image to be used in *image-based rendering* shows some similarities with the process proposed in this paper, where the seamless join of images taken from the real world is one of the goals [24, 14, 5, 16, 21].
We propose the integration of different techniques (range scanning, image processing, inverse texture-mapping and surface-based 3D graphics) to extract pictorial detail from images and to past it on a standard triangle-based mesh. An optimal correspondence between regions of the 3D mesh and sections of the acquired images is built. In particular, a new approach is proposed to produce a smooth join between different images that maps on adjacent sections of the surface (where it is crucial to preserve discontinuity of pictorial detail, e.g. edges or lines, and continuity of chromatic content). This is done by applying an accurate piecewise local registration of the observed images: an overlapping region 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 by resampling a new texture patch, obtained as a weighted blend of the corresponding adjacent images.

The paper is organized as follows. In the next section we introduce the four constituent phases of our approach, presented in detail in the following sections. The *vertex-to-image* binding phase is presented in Section 3. The determination of a potentially optimal decomposition of the mesh in contiguous areas that map to the same texture image is described in Section 4. The management of texture blending on the frontier faces is introduced in Section 5. Finally, all active texture patches are packed in a single rectangular texture, see Section 6. Issue related to the quality of the final texture mapping obtained are briefly discussed in Section 7. An evaluation of the proposed approach is presented in Section 8 and concluding remarks are in the last section.

2 Texture Stitching and Blending

Our approach de-couples shape and detail acquisition. We assume that the shape of the considered object has been acquired using whichever type of scanning device. A triangle mesh M becomes one of the inputs of our detail acquisition process; it can be either the original range-scan mesh or a partially simplified one.
The pipeline of the processing phases is as follows.

[1] A comparison between our approach and the one by Soucy et al. [22] is not easy, because not sufficient detail is given in their paper regarding the technique they use to integrate different texture images on a single mesh.

Pictorial detail acquisition and registration
Our detail acquisition methodology requires only inexpensive hardware: a digital still camera or a high-resolution video camera. To acquire pictorial detail we shot images from a set of different view points. The set of viewpoints is defined such that each area of the sample object is visible at least from one viewpoint.
Then, all the different views are registered with respect to the object shape M. This is a well known problem, and we adopt a classical approach [25] 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.

Texture Stitching
In this phase all the detail images are stitched on the shape mesh and partially fused, to generate a single textured triangle mesh. Intuitively, we built an optimal patchwork of images subsections such that all of the object surface is covered and adjacent image subsections join smoothly on the object surface (both in terms of color and pictorial detail continuity). The texture patching process works on the mesh vertices, and it is composed of four sub-phases (described in detail in the next sections):

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 valid images, we select initially as *target* image the one which is most orthogonal to surface M 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 mesh M that map to the same target image (i.e. such that all the 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 smoothing:**
   The previous step produces some faces that have vertices associated with different images. We call these faces *frontier* faces, i.e. faces which are on the border of adjacency between different target image sub-sections. Particular attention is paid to prevent the production of discontinuity in the representation of pictorial detail. Therefore, we first apply a *local registration* to all of the vertices of the frontier faces; second, we resample a new triangular texture patch for each of these faces. This new texture patch is computed as a weighted composition of the corresponding triangular sections in the associated target images;
4. **texture-patches packing:**
   All the target image sub-sections and the set of resampled triangular texture patches are packed into a single rectangular texture map, and texture coordinates in the triangle mesh are updated accordingly.

3 *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 it is a *non-silhouette* vertex.

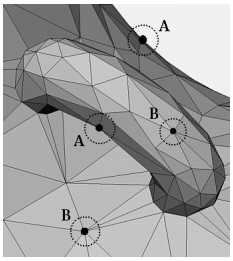

Fig. 1. Classification of the mesh vertices with respect to a given image i_k: (A) silhouette vertices, (B) non-silhouette vertices.

To verify if a vertex v is **visible** in a given image i_k we must check that: the projection of v on the image plane is contained in the image i_k, AND the normal vector in v is directed towards the viewpoint (i.e. the angle between the normal and the view direction has be lower than $\pi/2$), AND there are no other intersections of mesh M with the line that connect 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 [10]. The use of an hardware-accelerated z-buffer solution is also feasible (but will produce results dependent on the rasterization resolution).

Moreover, vertices are classified with respect to a given image as: **silhouette** and **non-silhouette** vertices. In the first case, at least one face oriented opposite to the view point should exist in the fan of triangles in M that are incident in v (see Figure 1). We avoid including in the *valid set* images in which v is classified as a silhouette vertex because possible small registration errors on v are easy to notice on these images.

If for a given a vertex v the valid image set is empty, then the *vertex–to–image binding* module will request that the user takes some more images, showing the area that has not been sampled.
If, conversely, we have multiple images in the valid set, we initially select as *target image* the one 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. See in Figure 2 an example of how much distortion can be introduced while mapping an image to a nearly orthogonal mesh section.

4 Patch growing

In the previous *vertex-to-image* binding phase we worked on each mesh vertex independently, without considering vertex and face adjacency. Mesh faces are mapped to a given texture and classified 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** face;
if, conversely, the vertices of f are linked to two (or even three) different target images, then face f is classified as a **frontier** face.

Because some aliasing might be introduced while resampling texture patch for *frontier* faces (multiple images composition might introduce some degradation and smooth-

Fig. 2. An example of two *target 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 may note how poor and distorted is the detail retrieved 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.

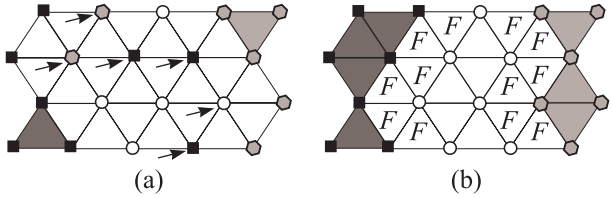

Fig. 3. Iterative local optimization of texture coverage on a sample mesh subsection: vertices are initially assigned to three target images (represented by an hexagon, a square and a circle). Then, we select a set of frontier vertices (indicated with arrows) and change their target images, obtaining configuration (b), which now correspond to a local minimum. Frontier faces are indicated with an *"F"* in (b).

ing, due to possible miss-alignment and color variation, see Subsection 5), we need to minimize the percentage of frontier faces and produce a mapping where contiguous areas of the mesh will be mapped, as much as possible, on contiguous areas 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 updates reduces the global number of frontier faces. During the iterative process a vertex can change multiple times its target image, until we get a minimum (see Figure 3). The results produced in our experiments were very good (see an example in Figure 10 in color plate and the results in Section 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. A similar approach has been adopted in [20] to optimize the face–to–image mapping, but it is based on a more simple single step evaluation phase.

A different approach proposed by Marschner builds texture patches by taking into account only the mesh topology [13]. A disadvantage of Marschner's approach is that a complete resampling is needed to build each circular texture maps; conversely, because in our approach resampling is limited to the frontier faces only (see Section 5), our output textures are noticeably less blurred.

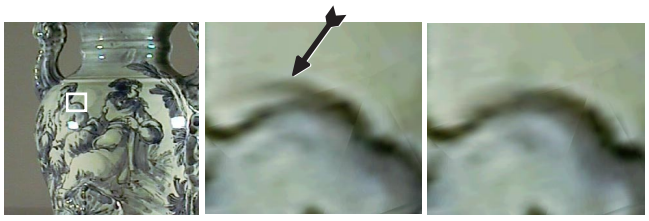

Fig. 4. Local registration phase: the section of the vase surface marked in image (a) is represented (magnified) in images (b) and (c); an example of the ghost effect generated by a slight miss-alignment is shown in the composite texture section shown in (b), while the more precise texture section generated after *local registration* is shown in (c).

5 Patch boundary smoothing

Other *view-dependent texture mapping* approaches blend at rendering time multiple weighted textures [8, 16]. Conversely, we resample a new triangular texture patch for each *frontier* face. This patch is computed as a weighted composition (or cross-fading) of the corresponding triangular sections of the target images associated with the face vertices.

Let us suppose that a set of three target images i_a, i_b, i_c is associated with the vertices v_1, v_2, v_3 of a given frontier face f. For each vertex of f we have the corresponding texture coordinates to the three target images. To build the corresponding triangular texture patch we have firstly to decide its resolution (in texel units); this depends on the corresponding resolution of the target images sections, to avoid to loose information while blending the images.

Then, the frontier face f is scan-convert (using the above sampling step). For each sampling point produced by the scan conversion, represented by its baricentric coordinates:
$p = \alpha\, v_1 + \beta\, v_2 + \gamma\, v_3\,, \quad \text{with } \alpha + \beta + \gamma = 1\,,$
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 composition of these three values, weighted using the barycentric coordinate factors:
$c_p = \alpha\, i_a[\bar{p_a}] + \beta\, i_b[\bar{p_b}] + \gamma\, i_c[\bar{p_c}]\,.$

The sampling point p might be not 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 images which correspond to a frontier face is critical, because we do not want to introduce discontinuity in the representation of detail. A potential problem can be an insufficient accuracy of the *Image Registration* phase, 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 limited numeric precision in the computations of the camera parameters. These misalignments may produce

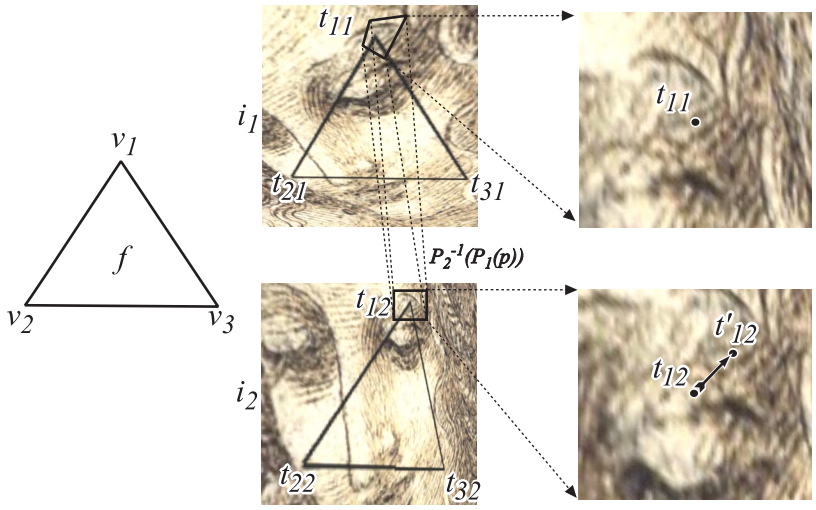

Fig. 5. Local registration of the texture coordinates of a frontier face vertices that maps on different target images.

ghost effects in the resampled triangular texture patches. Ghost effects are more easily perceivable if the pictorial detail contains very thin lines or abrupt changes of the color information (see an example in Figure 4). The vase used in our experimentation is a very challenging object, because the pictorial detail is composed by many thin painted lines.

To reduce drastically this possible aliasing we 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 of the *local registration* is to remove small miss-alignments that can be introduced during the different phases of our overall acquisition pipeline. Our solution is completely automatic and adopts an image-processing approach (see Figure 5).

Shum and Szelinski noticed a similar problem in the registration of multiple images for the construction of panoramic mosaics [21]. They propose a local alignment solution which removes ghosting effects by dividing the images in 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 area associated to frontier faces. For each frontier face we simply compute a local registration of the texture coordinates of its vertices with respect to [at most] the three target images onto which the frontier face maps. Let us see how our approach works.

For each frontier face f of the mesh we have: the three target images i_1, i_2, i_3 associated with the three vertices v_1, v_2, v_3 of f; 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 is defined also a projection function $\mathcal{P}_k : \mathbf{R}^3 \rightarrow \mathbf{R}^2$ which computes the texture coordinate corresponding to each mesh point p (and obviously $t_{hk} = \mathcal{P}_k(v_h)$).

During the *local registration* phase we maintain fixed the texture coordinates t_{kk} (that is, for each vertex the corresponding coordinates on its target image remain fixed; e.g.

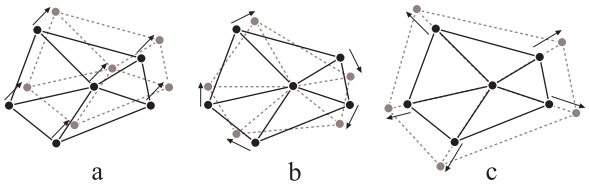

Fig. 6. Even if only translations are computed for each vertex, the combination of all of these transformations may result in a *affine transformation* of the corresponding target image section.

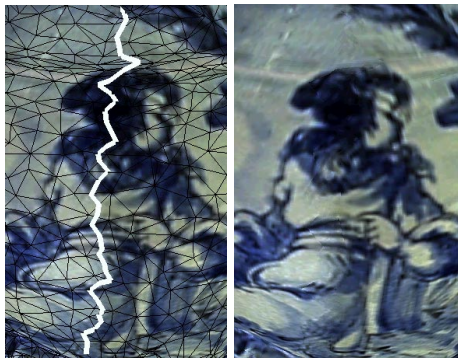

Fig. 7. As an example, a stripe of frontier faces is marked by the thick white polyline (on the left); the same *vase* section is rendered on the right.

vertex t_{11} in Figure 5). We process the vertices of face f one at the time. Starting from vertex v_1 of f, we compute an optimal translation of the texture coordinates t_{12} and t_{13} of the same vertex on the other two images. Each translation is computed by comparing independently a section of i_1 with the corresponding sections on i_2 and i_3. For example, to compute the optimal translation of t_{12}, we select on the target image i_2 a rectangular area centered on point t_{12} (see Figure 5); for each texel in this area, we compute the corresponding texel in i_1 (by projecting it firstly on 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 of i_2, that can be aligned by maximizing the cross-correlation between these two image sections [3].
This process is iterated for each vertex, and returns six new texture mapping coordinates. An example of the improved texture obtained after local registration is shown in Figures 4 and 7.
Please note that even though for each vertex we compute a simple translation of its texture coordinates, the result of the application of different translations on a set of nearby vertices can also correct empirically a slight rotation or scaling error introduced in the *Image Registration* phase. An example is in Figure 6.

Obviously, 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

Fig. 8. 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 cause some space overhead in the final texture T_M (shown on the right).

needed at all).

6 Texture-patches packing

For each triangle in M we therefore have texture coordinates to either one of the initial images or one of the triangular texture patches corresponding to frontier faces. In this last sub-phase, all the target image sub-sections (e.g. the areas of the initial images that have faces mapped into) and the set of triangular texture patches are packed into a single rectangular texture map T_M, and texture coordinates in the triangle mesh are updated consequently. We store all the detail in a single texture map to allow space/time efficient rendering on standard graphics systems (e.g. OpenGL, VRML).

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 8 in color plate, on the left). In case 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 (see an example in Figure 9).
As for the triangular texture patches, we join pairs of these patches to form rectangular areas (obviously, by joining pair of faces which have identical or similar size).
Then, all the rectangular texture sections are packed in a single texture T_M (see Figure 8 in color plate) by using a *cutting-stock* algorithm [2]. Even if more sophisticated solutions exist in literature for the polygonal area packing problem, this packing solution has been chosen because it is simple to implement and sufficiently efficient (at least, when the number of rectangular texture sections is not huge, as in the case of the meshes used in our experiments).

It is obvious that, once packed, a triangular texture has adjacent patches in the packed texture that are probably different from the ones assigned to the geo-topologically adjacent faces. A packed texture therefore might not produce a proper filtering across edges which are shared between all those triangle pairs that are now disjoint in the packed texture. But this problem can be simply overcome by a keen sampling of each

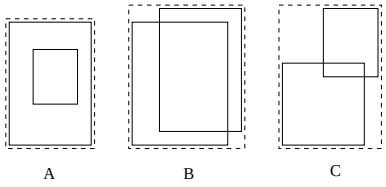

Fig. 9. The figure shows the possible combinatorial cases: in the case of configuration (A) and (B) we pack in T_M the bounding rectangle of the two sections (dashed area), because its size is smaller than the sum of the two areas; conversely, in case (C) it is more efficient to pack the two sections independently.

frontier face f. The associated triangular texture patch t_f is resampled few pixels larger than the required minimal size [7]. In this manner we ensure that at rendering time for each scan-converted pixel $p_i \in f$ we have that its texture coordinates are always internal to the associated discrete texture area t_f.

7 Perspective Distortion in Texture Mapping

One potential problem that can arise when mapping images taken from reality on a 3D geometry is the geometrical texture distortion that can be introduced. This because we have a non-linear perspective transformation and a corresponding bilinear 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 can appear distorted. This distortion decreases as the image viewpoint is closer to being orthogonal to the triangle face. The need to rectify textures to remove distortion also makes it difficult to merge rectangular textures, as pointed out in other papers.

In our case, the job is much simpler because we can assume to have a mesh described by small-area triangles (where triangle area is measured as the texel area of the corresponding texture subsection); in this case the distortion may be really small, and rectification may not be necessary.
Moreover, note that some recent hardware and API's now 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 distance from camera to vertex as the third w coordinate.

8 Evaluation of results

We report here the results obtained on two target objects. The first one is a ceramic *vase* of Albissola, with a height of around 40 cm. and a very complex pictorial detail (mostly thin painted lines). The *vase* geometry was acquired with a Cyberware 3030 range scanner. The mesh obtained was originally composed of 2M faces, and it was simplified down to 20,000 faces using the Jade 2.0 simplification code [6] with an accuracy of 0.17 mm. The second object is a small Capodimonte *statue* (around 25 cm. tall), whose geometry was acquired using a CT scanner. The geometry, reconstructed via iso-surface fitting on the volume model, is around 1.5M faces, and was simplified down to 10,000

faces. In both cases, detail mapping has been done starting from the simplified models.

Texture images have been acquired by using a KODAK DC210, a commercial digital still camera which can be driven by software and supports a sufficiently high acquisition resolution (1152x864) in true color. The detail of the *statue* model, that has a rather complex shape, was acquired by taking 14 different views. Conversely, only 8 views were sufficient for the *vase*.

Data processing times needed to perform the initial *Image Registration* are obviously user-dependent; in our case, the registration of each view is performed in less than 5 minutes. The automatic texture integration and patching process running time depends on the number of total images and the complexity of the object mesh (i.e. the number of faces and, only secondarily, the topologic complexity of the mesh). The running times needed to process the two test objects were 191 seconds for the *vase* and 132 seconds for the *statue* (on a SGI O2 R5000 180MHz).

The detail texture produced for the *vase* is shown in Figure 8 in color plate. A visual comparison of synthetic images and photographic images the real objects is in Figure 11 in color plate.

9 Conclusions

In this paper we have proposed a system for the semi-automatic acquisition of the pictorial detail of 3D free-form objects, and its integration with the 3D shape through standard texture-mapping. In particular, detail observed in a small set of photographic images is registered and patched on the input mesh (acquired with any automatic acquisition methodology). Advantages of the process are the very limited user-intervention required (i.e. selection of the view set, initial rough registration) and the high quality results obtained, due to the original texture patching process proposed. In particular, the global texture is produced as a patchwork of textures sections and gives a very precise representation of the observed detail because of the image-based local registration of the target images and the computation of resampled triangular texture patches for the frontier faces.

Some possible extensions on which we are now working are oriented to the reduction of the possible chromatic variations between overlapping images (possible either by increasing the width of the frontier region, or by adopting a multiresolution approach which blends only low frequency texture information [4]).

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 statue.

References

1. R. Baribeau, M. Rioux, and G. Godin. Color reflectance modeling using a polychromatic laser sensor. *IEEE Trans. on P.A.M.I.*, 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. L. Gottesfeld Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, Dec. 1992.
4. P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, October 1983.

5. S.E. Chen. Quicktime VR - an image-based approach to virtual environment navigation. *Comp. Graph. Proc., Annual Conf. Series (Siggraph '95)*, pp.29–38, 1995.
6. A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *The Visual Computer*, 13(5):228–246, June 1997.
7. P. Cignoni, C. Montani, C. Rocchini, R. Scopigno, and M. Tarini. Preserving attribute values on simplified meshes by re-sampling detail textures. Technical Report IEI-B4-37-12-98, IEI – C.N.R., Pisa, Italy, Dic. 1998.
8. P. Debevec, Y. Yu, and G. Borsukov. Efficient view-dependent image-based rendering with projective texture-mapping. *Rendering Techniques '98*, page 14. Springer Wien, 1998.
9. P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Comp. Graph. Proc., Annual Conf. Series (Siggraph '96)*, pp.11–20, Addison Wesley, 1996.
10. 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*, pp.288–297, Geneva, Switzerland, 1988. Springer-Verlag.
11. G. Kay and T. Caelli. Inverting an illumination model from range and intensity maps. *CVGIP - Image Understanding*, 59(2):183–201, March 1994.
12. J. Lu and J. Little. Reflectance function estimation and shape recovery from image sequence of a rotating object. In *Proc. of Int. Conference on Computer Vision*, pp.80–86, 1995.
13. S.R. Marshner. *Inverse rendering for computer graphics*. PhD thesis, Cornelle Univ., 1998.
14. L. McMillan and G. Bishop. Plenoptic modeling:an image-based rendering system. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '95)*, pp.39–46. ACM Siggraph, 1995.
15. M. Petrov, A. Talapov, T. Robertson, A. Lebedev, A. Zhilyaev, and L. Polonsky. Optical 3D digitizers: Bringing life to the virtual world. *IEEE CG&A*, 18(3):28–37, May – June 1998.
16. 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 *Proceedings of 8th Eurographics Workshop on Rendering (St. Etienne, France)*, pp.23–34, June 1997.
17. H. Rushmeier, F. Bernardini, J. Mittleman, and G. Taubin. Acquiring input for rendering at appropriate levels of detail: digitizing a pietá. *Rendering Techniques '98*, Springer Wien, 1998.
18. H. Rushmeier, G. Taubin, and A. Gueziec. Applying shape from lighting variation to bump map capture. In P. Slusallek J. Dorsey, editor, *Eurographics Rendering Workshop 1997*, pp.35–44. Springer Wien, June 1997.
19. 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.
20. Y. Sato, M.D. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. *Comp. Graph. Proc., Annual Conf. Series (Siggraph '97)*, pp.379–388, 1997.
21. 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*, pp.953–958, 1998.
22. 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*, pp.409–412, 1996.
23. R. Szeliski. From images to model (and beyond): a personal retrospective. In *Vision Interface '97*, pp.126–137. Canadian Image Processing and Pattern Recognition Society, 1997.
24. R. Szeliski and H.Y. Shum. Creating full view panoramic image mosaic and environment maps. *Comp. Graph. Proc., Annual Conf. Series (Siggraph '97)*, pp.251–257, 1997.
25. 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.
26. Y. Yu and J. Malik. Recovering photometric properties of architectural scenes from photographs. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pp.207–218. ACM SIGGRAPH, Addison Wesley, July 1998.

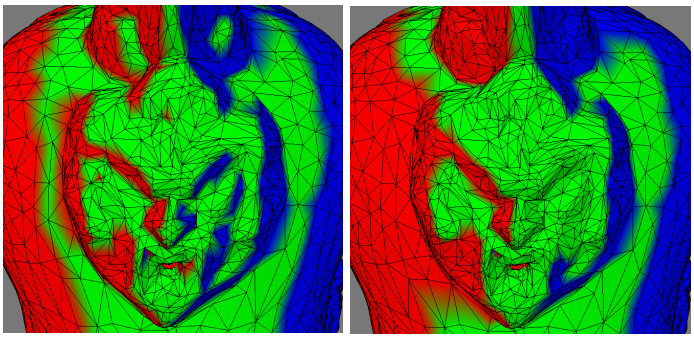

Fig. 10. An example of optimized frontier faces management: we have 1,137 frontier faces out of the total 10,600 faces in the initial configuration on the left; we get only 790 frontier faces after optimization (on the right).

Fig. 11. A comparison of the accuracy of the synthetic models (right) *wrt* the original object images (left).