A 3D Reconstruction Pipeline for Digital Preservation

Alexandre Vrubel Olga R. P. Bellon Luciano Silva
IMAGO Research Group (http://www.imago.ufpr.br)
Universidade Federal do Parana, Curitiba, Brazil *

Abstract

We present a new 3D reconstruction pipeline for digital preservation of natural and cultural assets. This application requires high quality results, making time and space constraints less important than the achievable precision. Besides the high quality models generated, our work allows an overview of the entire reconstruction process, from range image acquisition to texture generation. Several contributions are shown, which improve the overall quality of the obtained 3D models. We also identify and discuss many practical problems found during the pipeline implementation. Our objective is to help future works of other researchers facing the challenge of creating accurate 3D models of real objects.

1. Introduction

Digital reconstruction of 3D models from range and color images is a very active research field, but still includes many challenges. In this context, there are two main surveys [3, 13] presenting an entire reconstruction pipeline. Also, pioneer works focusing on digital preservation of cultural heritage are presented in [4, 15, 18]. These works contribute by highlighting the difficulties to be overcome in scanning complex objects.

In this paper, we show how we built a functional 3D reconstruction pipeline, aiming for high quality results required in the digital preservation of natural and cultural assets. Our contributions and the rationale behind the choices we make are meant to help future works in this area.

In our work we used a commercial 3D scanner, the Vivid 910 from Konica Minolta. However, the reconstruction software shipped with the scanner has several limitations: the alignment of views is an arduous process; the mesh integration sometimes generates incorrect surfaces that need to be manually corrected; the hole filling does not work in holes with complex topologies; and the generated texture is of low quality and incorrectly parametrized in all but the simplest objects. All these factors make inappropriate its use for digital preservation, and motivated our development of a **working** high-quality 3D reconstruction pipeline.

Several steps compose a complete 3D reconstruction pipeline, as shown in Fig. 1.

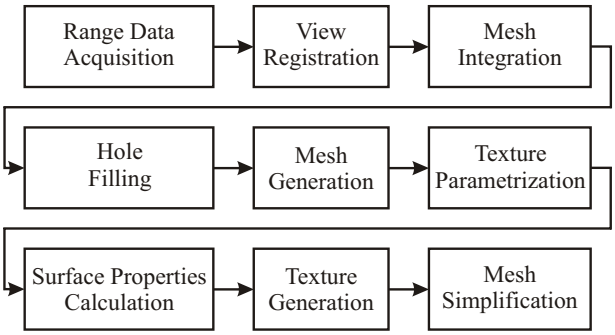


Figure 1. Overview of our 3D reconstruction pipeline.

First, the data is acquired from different and sufficient viewpoints. Next, the data is aligned into a common reference frame in a process known as registration. After alignment, follows the mesh integration stage where data from all acquired views are combined. Eventual holes due to incomplete data acquisition are usually filled after the integration step. Then, a 3D model with its textures (*i.e.* diffuse color, specular and normal maps) is generated. Finally, mesh simplification may be performed to improve rendering performance and storage costs. We follow this sequence of stages to present our solutions as well as other related works.

2. The 3D Reconstruction Pipeline

2.1. Acquisition

There are several types of acquisition devices to gather depth information from an object: laser scanning techniques, multi-view stereo, shape from structured light, shape from silhouette, contact digitizers, among others. From all these techniques, laser scanning is the most precise [22], being our choice.

One current avenue of research is dedicated to improving the quality of the acquired data. Nehab *et al.* [19] combine

*Thanks to CAPES, CNPq and FINEP for funding.

depth information from a triangulation scanner with normal information from photometric stereo. Park and Kak [21] proposed a technique to capture optically challenging objects through the usual laser triangulation technique with some modifications. Any improvement in the quality of acquired data is very welcome since it surely improves the fidelity of the 3D reconstruction.

The first obstacle we faced was how to separate the object from the background. The segmentation by color is problematic, because it needs a controlled acquisition background, and places restrictions on colors present on the scanned object. Scanning the object placed over a black surface (to avoid laser capture), is not a good option either, because dark regions of the object need higher laser power; therefore leading to the capture of the black surfaces used as support for the object.

Our contribution to this stage is a new method able to separate the object from the background data. In our approach, we assume that the object is always scanned over a support plane (*e.g*. a table). Then, we automatically detect and remove this plane, since we know that there is no information **below** the plane, and the object data is **above**. If we project all points into the support plane normal direction (using dot product), the points on the plane would project at the same value, the points on the object would be spread with values larger than the plane, and there would be very few values below the plane value (due to noise). If we build a histogram of the distribution of these values, one can find a distinct profile (see Fig. 2).

Our method tries different normal directions (sorted by occurrence on the data points), searching for a histogram that matches the presented profile. When one is found, we select the points on the peak, and refine the plane parameters with MSAC [29]. The refined plane is accepted and found if it conforms to a plane thickness threshold (usually around 2 mm), minimum plane percentage of total points (usually around 5%), maximum below plane percentage of total points (usually around 2%) and minimum above plane percentage of total points (usually around 10%). If any of these conditions fail, a new plane normal direction is tried. The thresholds were obtained empirically, and can be adjusted to different compromises between precision and detection. Once the plane is obtained, only the points above the plane are marked as belonging to the object.

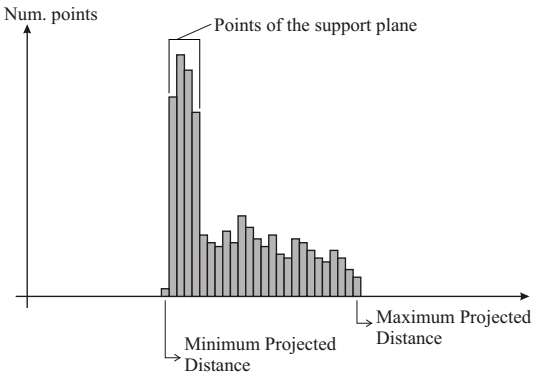
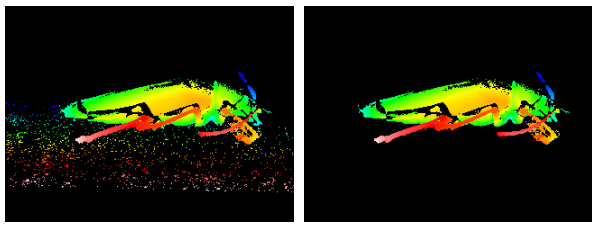


Figure 2. Example of a distribution of range points over a candidate support plane normal direction.

The advantage of our method is that the control of the background colors is unnecessary, what is very useful for acquisitions made outside a controlled environment. Besides, our method does not place any restrictions on scanning parameters (*e.g*. focus distance or laser power). For example, when scanning using a turntable, the technique consists of scanning only the turntable, and then discarding these data from each captured view. This requires that the scanner position and focus be constant on all captured views, an awkward limitation when scanning complex objects. Such a limitation does not exist in our technique. Finally, the detected plane can be helpful in the mesh integration stage, because it defines a half-space that is known to be outside the scanned object. Fig. 3 shows the automatic support plane detection.



(a) (b)

Figure 3. Range image of a real insect (beetle), showing a plane difficult to detect: (a) original range image, with several disconnected patches belonging to the support plane; (b) result from our automatic support plane detection and removal algorithm.

2.2. Registration

The objective of this stage is to find a 4×4 transformation matrix for each captured view to achieve the alignment into a common object coordinate system. Rusinkiewicz and Levoy [25] and Salvi *et al*. [28] present several algorithms that can be used in the registration stage.

In our pipeline, we use a pairwise ICP alignment [25], followed by a global registration step using Pulli's algorithm [23]. For each pair of neighboring views with sufficient overlap, we find the transformation matrix that aligns the second view with the first, using a modified version of the ICP algorithm, presented below. Currently, we manually pre-align the views; however, automatic pre-alignment techniques like in [1] can be used to improve this task.

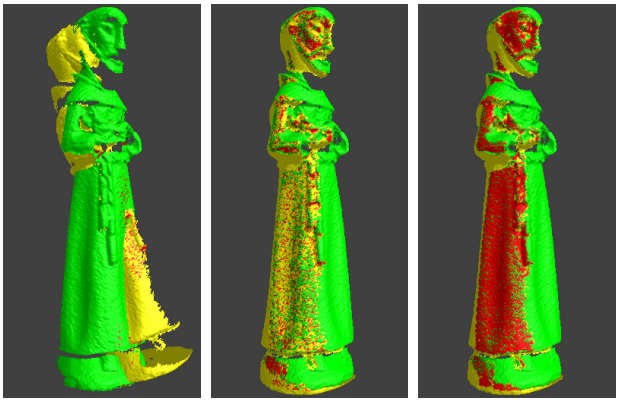
Our contribution regarding this stage is a new two-phase ICP algorithm. We needed an algorithm with good convergence properties (to reach the correct alignment), and with maximum precision. To achieve this, the first phase uses an

ICP variant with the point-to-plane error metric, a closest-compatible approach for pair generation, normal space sampling with random selection of points on both views, and rejection of the farthest pairs [25]. This promotes excellent convergence, but with limited precision.

When this first phase converges, we move on to the second phase of the ICP algorithm. Now, we use all points on both views, adding a maximum pair distance into the pair compatibility test. This distance is related to the scanner error, and is usually very small (*e.g.* around $0.7\ mm$). We still use a point-to-plane error metric during error minimization. This version of ICP has limited convergence, but excellent precision. As the first phase already reached an almost optimal alignment, the second phase just improves the precision of the result. As the compatibility threshold is very restrictive, we achieve good outlier elimination, which is essential for a precise alignment.

Fig. 4 shows an experimental result from this two-step ICP-based approach; even with the low overlap and bad initial position, the result converged to a precise alignment.

After the pairwise alignments, the global registration algorithm of Pulli [23] improves the final alignment, spreading the errors equally between all view pairs.



(a) (b) (c)

Figure 4. Our two-phase ICP-based approach: (a) initial position of two views, far from being aligned; (b) result from the first phase of our algorithm; (c) final result, with alignment precision enhanced by the second phase of our algorithm. Red is used to represent the matching pairs.

2.3. Mesh Integration

After the registration, we have several overlapping partial meshes, one for each captured view. The next stage of the reconstruction pipeline must integrate them to build a single triangle mesh for the object. There are several approaches for mesh integration: Delaunay-based methods, surface-based methods, parametric surfaces and volumetric methods [3], all of them presenting limitations. Delaunay-based methods are costly, and require pre-processing techniques to eliminate incorrect data. Surface-based methods may have problems with complex topologies and are sensitive to noisy data. Most parametric surfaces algorithms cannot handle sharp corners properly and surface fitting can be problematic due to the outliers. Finally, volumetric methods are costly regarding both time and space. In addition, Kazhdan *et al.* [14] compared several recent algorithms, neither one was able to ensure high fidelity reconstructions, especially at small-scale details.

We chose volumetric methods because they impose fewer restrictions to reconstructed objects; offer an easy way to change the precision of the output (by varying the voxel size); can easily support the space carving technique [6] to help outlier elimination; and can work in the presence of low quality input data. Besides, they present better results compared to other recent techniques [14].

We exhaustively implemented, tested and modified three algorithms: The VRIP from Curless and Levoy [6], used in *"The Digital Michelangelo Project"* [15]; Consensus Surfaces from Wheeler *et al.* [30], used in *"The Great Buddha Project"* [18]; and our new algorithm, developed to solve the limitations present in the two previous methods.

The VRIP in general achieves good results, but presents some artifacts near corners and thin surfaces. Consensus Surfaces, even with the improvements by Sagawa *et al.* [26, 27] and some of our own, still generates incorrect results in regions near occlusions, as these regions rarely can achieve consensus.

To solve these drawbacks, we developed a new algorithm that combines elements from both VRIP and Consensus Surfaces. Our new algorithm is based on two phases. In the first one, we use a slightly modified version of VRIP, together with a space carving method, to generate an initial volumetric representation. Our modification on VRIP is a new weight curve (see Fig. 5), that gives more weight to outside voxels than to the ones inside the objects. This attenuates the artifacts of VRIP in corners and thin surfaces, at the cost of a small misplacement of the surfaces in the first phase. Our space carving takes into consideration only the object data, and optionally the support planes detected in

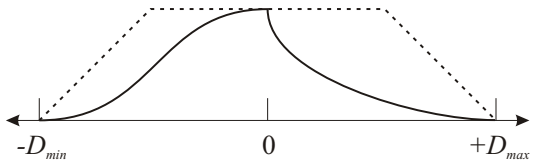
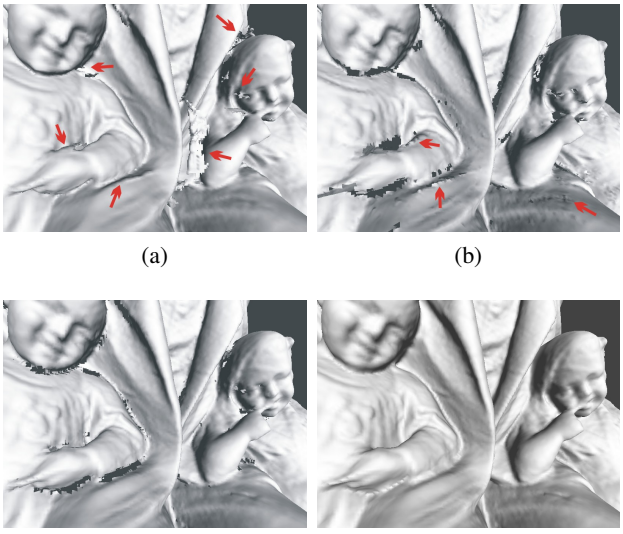


Figure 5. Distance weight curves for VRIP. The original curve [6] is shown with the dashed line, and our new curve with the solid one. Our new curve is a simple concatenation of two bezier segments. It reduces both creases and the effect of outliers on the integrated surface. This factor ranges from 0.0 to 1.0 (according to the signed distance), and is multiplied by the other weight factors. Negative values of distance are outside the object.

the acquisition stage, having as main goal the outlier elimination. The volumetric result of this first phase works as a consensual basis for the second phase of the algorithm.

The second phase builds the definitive volumetric representation, integrating only measurements in consensus with the result obtained in the first phase. The consensus is tested at each candidate voxel, between the normal on the closest surface point of each view and the gradient of the volumetric result from the first phase. The space carving performed on the first phase is also used to eliminate outliers, here standing for the incorrect data outside the object. We must note that we use line-of-sight signed distances on the first phase (VRIP) for performance, and Euclidian distances on the second phase (Consensus) for precision and correction of the hole filling later on.

With our algorithm, we eliminate the artifacts of VRIP near corners and thin surfaces, and generate good results near occluded regions. Fig. 6 shows a comparison of results from the integration algorithms discussed previously.



(a) (b)

(c) (d)

Figure 6. Comparison of integration algorithms: (a) VRIP; (b) Consensus Surfaces; (c) our new integration algorithm, which successfully removed all outliers; (d) our algorithm with holes filled with the Volumetric Diffusion algorithm [7]. Both VRIP and Consensus Surfaces generate incorrect surfaces due to outliers (indicated by the red arrows), spoiling the subsequent hole filling. Our new algorithm, in contrast, eliminates all outliers and keeps the space carving information, allowing the hole filling stage to generate very good results.

2.4. Hole Filling

The acquisition process is usually incomplete. Deep recesses and occlusions prevent the capture of some parts of the objects. This requires some efforts to complete the captured data to allow the generation of a "watertight" model, necessary for several applications such as user visualization and creation of replicas.

Some integration algorithms fill holes automatically, like the ones based on parametric surfaces [5, 20]; however, the results are not always topologically correct. Some simple techniques catastrophically fail in holes with complex topology, common in real objects, when they assume that the holes have a disc topology.

In our pipeline we chose the volumetric diffusion algorithm by Davis *et al*. [7], because it can handle complex topologies satisfactorily. Besides, it is a volumetric technique that works well with our mesh integration stage. The idea of the algorithm is to diffuse the values on observed voxels into voxels without data, similar to a 3D blurring operation. Space-carving information, although not necessary, usually helps the algorithm to produce a more faithful reconstruction.

The volumetric diffusion algorithm suffered some criticism by Sagawa and Ikeuchi [26], but we disagree with their assessment. In our experiments, the Volumetric Diffusion generated excellent results, mainly due to the quality of our integration method. The explanation lies on the characteristics of the integration algorithms used. Sagawa and Ikeuchi [26] use the Consensus Surfaces, which usually generates incorrect results near holes. So, it is natural that when propagating this incorrect information to fill holes, bad results are expected. Since our new integration method eliminates incorrect data near holes, and space carving data from the first integration phase is available, Davis' method is able to generate good results in these challenging cases. Therefore, we can say that Volumetric Diffusion is a good technique, but depends on a good mesh integration to work successfully. Fig. 6(d) shows Davis' method result after our integration algorithm was performed.

2.5. Mesh Generation

We use the well established Marching Cubes algorithm [17] to generate a triangle mesh from the volumetric representation of the previous stages. We use the disambiguation method of Chernyaev [16] to ensure the generation of manifold topologies.

The only drawback of this approach is the generation of very thin triangles (a.k.a. *slivers*) in some parts of the generated model. A mesh simplification technique like [11] can eliminate these triangles, resulting in a more homogeneous mesh, useful for the next stages of the pipeline.

2.6. Texture Parametrization

The mesh generation concluded the geometric part of the reconstruction problem. However, we still needed to calculate the surface properties (*i.e*. color and specularity). These properties are usually represented by textures. Therefore, we need to be able to apply textures to the generated model.

The goal of this stage is to generate a mapping between the 3D coordinates (x, y, z) of each vertex and a 2D texture coordinate (u, v). These 2D coordinates represent a position into the texture image. This process can be seen as "skinning" the model through cuts and planifications.

There are several methods to perform texture parametrization [12]. For our purposes, we needed a parametrization that minimized distortion, being at the same time as homogeneous as possible. In our implementation, we used a simple texture atlas approach. We segmented the model into almost planar regions, starting from a seed triangle and growing the region while the normals of the faces are within a threshold (usually $30°$, to prevent the generation of too many small regions) from the average normal of the region. Each region is then planified; this is done by calculating the principal axis of the vertices in question [31]. The axis closest to the average normal of the region is then used as the normal of the plane, and the other two axes define the u and v directions in the texture space. The result is a 2D projection (in mm) of each region. After all regions are planified, a texture atlas is generated, packing all regions into a single parametrization space (see Fig. 7). As we know the size of each region in millimeters, it is easy to define the image size in pixels necessary to achieve a desired resolution in pixels per millimeter.

We must notice that any parametrization can be used with our pipeline. An extensive review of more complex parametrization alternatives is presented in [12].

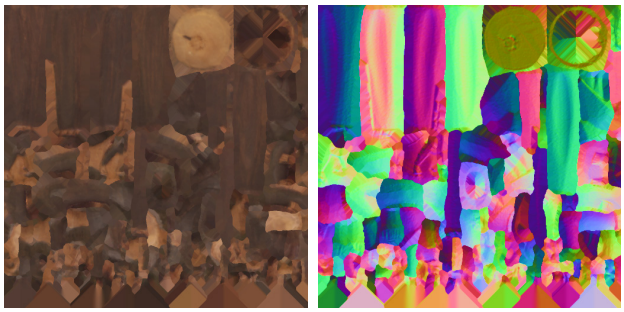
It is important to note that the trivial solution of generating an atlas with each triangle being a region is a really bad choice of texture parametrization. The performance of real time rendering suffers greatly due to lack of spatial coherence, and mip-mapping [8] becomes almost impossible to accomplish, consequently reducing the quality of the renderings. Therefore, minimizing the number of regions on the atlas is also an important criterion when choosing a good texture parametrization scheme.

2.7. Surface Properties Calculation

The main surface property we need to calculate is the reflectance or surface color. Additional properties, like specularity, are also useful in high fidelity reconstructions.

Acquiring accurate color information from the object is more challenging than it appears. Usually, we do not have complete control over the incident illumination on the scanned object. Even when this is tried, the simple illumination models commonly used in Computer Graphics (*e.g.* Phong, Blinn, Torrance-Sparrow [8]) are not physically realistic, since they ignore indirect illumination and object inter-reflections. Bernardini and Rushmeier [3] present the main techniques used to estimate the surface properties, including the compensation of the illumination parameters.

Another practical difficulty is that the commercial 3D scanners available usually return color information in low resolution. For example, the Vivid 910 we used returned images with 640×480 pixels of resolution, and the color is not reliable. This leads to the use of a different high resolution camera to acquire color images, and the need of calibration between this camera and the scanner data, another source of imprecision on the final result.



(a) (b)

Figure 7. Example of automatically generated textures for the object presented in Fig. 8: (a) diffuse color texture; (b) object-space normal map. We used our chart parametrization and calculated vertex colors and normals to generate both textures. As explained in section 2.8, we diffused the border of each chart into its neighboring pixels to prevent problems with mip-mapping.

There are two approaches to generate the surface properties. We can either generate color and illumination per vertex, as the models are usually high-poly; or we can generate them directly into the texture space, using the parametrization of the previous pipeline stage. The former is used when only data from the 3D scanner is available, while the latter is needed when using high resolution cameras.

Our current implementation still does not calculate an accurate photometric modeling. We use the simple vertex color approach, and we have been improving it using high resolution cameras. Specularity is not yet estimated, too. To generate the vertex colors, we calculate a weighted average of the colors on all views that observe each vertex. The weight we adopted is the angle between the scanner line-of-sight and the vertex normal. This is done because the data observed at an angle are less reliable than the data facing directly the scanner. Although simple, our method generates good results, as shown in Section 3.

2.8. Texture Generation

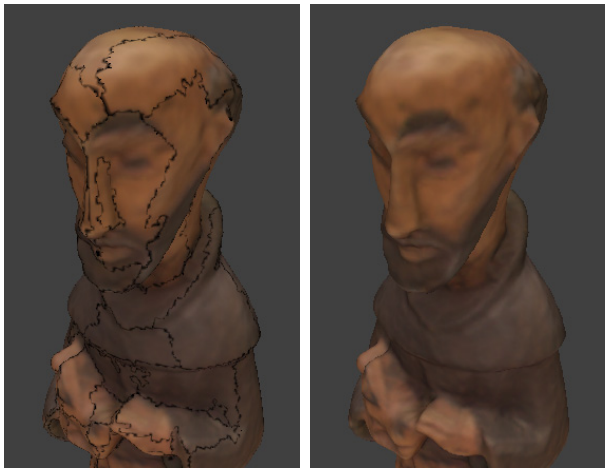
Texture generation combines the results of the two previous stages of the pipeline: texture parametrization and surface properties. Our objective is to encode the surface properties into one or more images (*i.e.* textures). These will be used when rendering the reconstructed model (see Fig. 7).

This stage and the previous two are very dependent on the algorithm used. Sometimes, they are condensed in a single stage [24]; in other cases, they are strongly related to

the acquisition devices used [2]. We prefer to separate these three stages, so that different techniques can be tested, and at the same time easily integrated in our pipeline.

As we explicitly generate the parametrization and vertex color, the texture generation is straightforward. We render the model using the texture coordinates $(u, v, 0.0)$ as the (x, y, z) coordinates of the vertices, and using the calculated vertex colors. The 3D graphics card interpolates the color across each face using Gouraud shading [8]. We use an orthogonal projection matrix, and a render target of the size of the desired texture. The same rendering technique can be used to generate other textures, like a *normal map* (encoding each vertex normal as a RGB triplet), or a *specular map* (encoding each vertex specular color and exponent as a RGBA tuple).

We found another practical problem when automatically generating textures: the perimeters of each parametrized region usually causes problems with mip-mapping [8]. This occurs due to the bilinear interpolation made when accessing texture maps. This appears as "cracks" on the final model that highlights the boundaries of the segmented regions. To solve this, we expanded the colors from the regions into the unused texture spaces, using a diffusion technique. We created this technique inspired on the Volumetric Diffusion [7] used for hole-filling, but here the diffusion is 2D and we propagate color instead of distance values. This works like a blurring filter, but only affecting the unused pixels of the texture, and using only the colors propagated from the regions. This technique can be used on any chart-based parametrization scheme, and is usually effective to solve the "crack" problem. Fig. 8 shows the problem and the solution with our method, while Fig. 7 shows an example of textures with diffused regions.



(a) (b)

Figure 8. Example of the "cracking" problem due to the mip-mapping of automatically generated textures: (a) rendered result with the original texture map; (b) rendered result with colors diffused into the unused texture space, with "cracks" eliminated.

2.9. Mesh Simplification

An optional pipeline stage consists of reducing the triangle count on the model to improve its rendering performance and storage costs. After capturing the geometric, color and eventually specular properties into textures, we can perform mesh simplification and still mantain the visually high accuracy of the source model.

When dealing with digital preservation, this step is not essential, since we want precise results. However, the Marching Cubes algorithm used in the pipeline can generate much more triangles than necessary to accurately represent the model, mainly in almost planar regions. So, a mesh simplification procedure can improve the performance keeping high accuracy. Another important fact is that we are able to generate a normal map for the model that helps preserve the visual accuracy even when low-poly models are used.

There are several approaches for mesh simplification. The technique of Garland and Heckbert [9], improved by Hoppe [11] is fast and generates accurate results when reducing moderately the polygon count, which is the goal of digital preservation. Using a progressive mesh representation [10] is also useful to allow the generation of different levels of detail for each object.

We prefer to perform the mesh simplification after the texture generation, because we are able to generate maximum quality normal maps, and the textures can guide the mesh simplification, thus minimizing texture distortion.

3. Results and Future Works

We used our pipeline to reconstruct several objects, ranging from artworks to fossils. The objects were selected to stress test the pipeline, with complex topologies and optically uncooperative materials. Table 1 shows characteristics of some reconstructed objects, presented in Fig. 9. In general, we are able to generate good quality reconstructions, even on complicated objects. Other examples can be found on our research group website (`www.imago.ufpr.br/Museu`).

Our implementation of a reconstruction pipeline had to overcome several practical problems:

Object Characteristics	Beetle	Rooster	Protocyon
Number of views	43	32	56
Number of ICP pairs	69	36	104
Data size	979 MB	671 MB	1.22 GB
Voxel size	0.3 mm	0.5 mm	0.5 mm
Dimensions (cm)	6×4×2	22×19×12	23×15×10
Number of vertices	136,706	296,786	436,863
Number of faces	273,424	593,584	873,746
Reconstruction time	322 s	963 s	1,348 s

Table 1. Some reconstructed objects with the proposed pipeline. We used a 2.2GHz Core2 Duo PC, with 4 GByte of RAM.

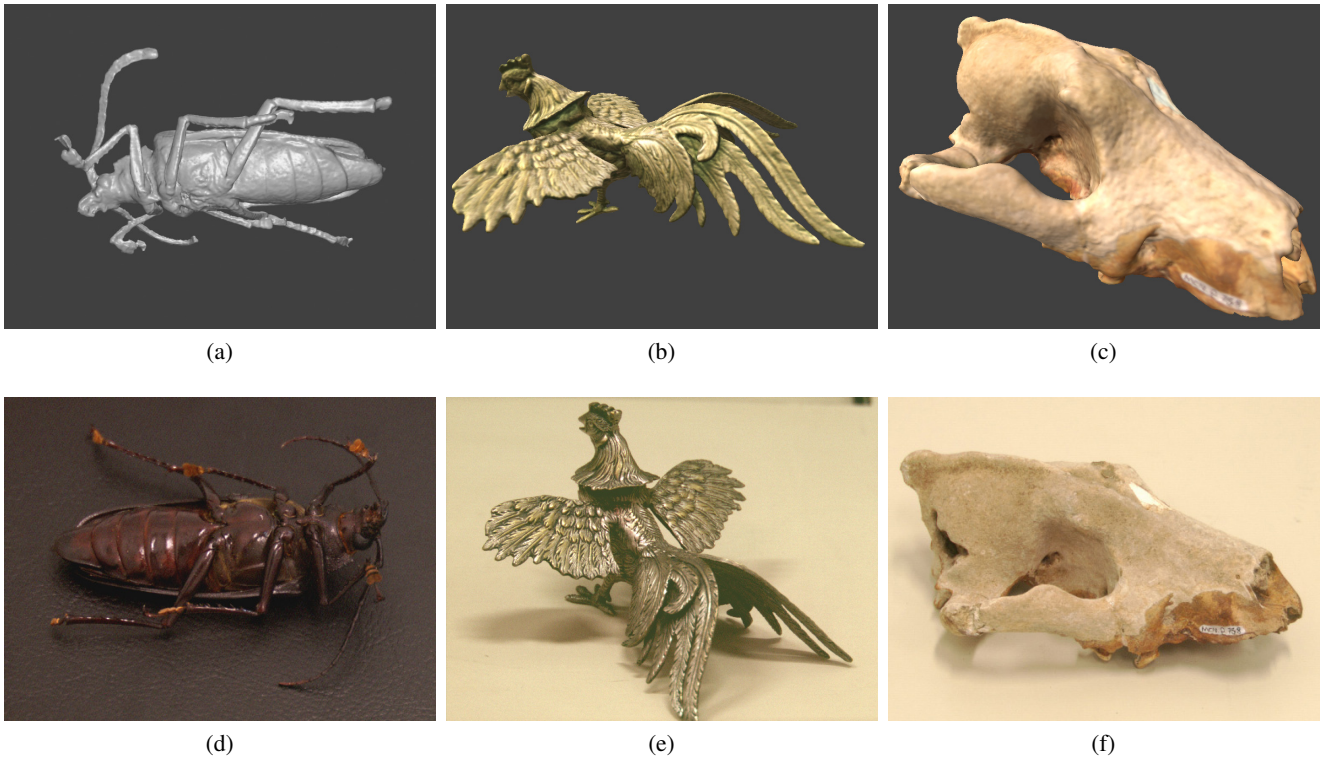


Figure 9. Results from our 3D reconstruction pipeline: (a) reconstruction of a beetle, challenging because the small scale of the details and dark colors; (b) reconstruction of a metal statue of a rooster, challenging because the specularity of the object material and thin gaps between the feathers; (c) reconstruction of a *protocyon* fossil (an ancient American wolf), challenging because the complex topologies, occlusions and thin surfaces; (d), (e) and (f) color images of the real objects. In all cases, high quality reconstructions were achieved.

- Even moderately sized objects generate large data sets. Keeping all these data loaded into memory is unfeasible, therefore temporary files, a cache mechanism, and cache-friendly algorithms are necessary;
- Current mesh integration algorithms still generate false or incorrect surfaces. This directly impacts the accuracy of the final models;
- Mathematically evaluating accuracy of the results is still difficult. When reconstructing real objects, we do not have a "ground truth" to compare the generated model to. Even producing a test object from a previous 3D model, to scan and reconstruct it later, does not solve the problem. This happens because any production process introduces inaccuracies that make the real object different from the source 3D model, making a simple comparison between the source and reconstructed model incorrect;
- Color images acquired with laser triangulating 3D scanners are usually of low resolution. This limits the generation of accurate textures for the 3D model.

Some future works can focus on improving several stages of the pipeline. An interactive tool to help planning the capture would be useful to minimize the effort during acquisition. Using some automatic pre-alignment for each pair of views would reduce the amount of human labor to generate the models. Improving the quality of the alignment would improve the precision of the resulting models. The development of better integration algorithms is another important avenue of research, so that only precise surfaces are generated. Using camera calibration techniques to combine the acquired geometry with high resolution photographs of the object would improve the quality of the textures.

4. Conclusion

The purpose of this work is to show a complete and functional solution for the 3D reconstruction problem applied to digital preservation. There are lots of algorithms and possibilities to build such a pipeline; we present our particular solution, and the reasoning behind the selection of the algorithms for each stage of the pipeline. We are able to reconstruct complex objects with good accuracy, proving that our approach is functional.

Our main contributions are: a new support plane detector to automatically separate the object from the background in the acquired range data; a new two-phase ICP-based al-

gorithm to achieve at the same time good convergence and good precision; a new volumetric integration algorithm that overcomes drawbacks from both VRIP and Consensus Surfaces, working nicely with the Volumetric Diffusion algorithm for hole-filling; a functional way to generate textures for the geometric models using a simple texture atlas approach combined with a per-vertex calculation of surface color; and an automatic rendering of the texture image using common 3D graphic accelerators. Another important contribution is a general overview of the entire pipeline, and how each stage interacts with the other stages; attention to this is paramount when building a functional 3D reconstruction pipeline.

We hope our work helps other researchers facing the daunting task of building a 3D reconstruction pipeline for digital preservation, facilitating its achievement.

References

[1] D. Aiger, N. J. Mitra, and D. Cohen-Or. 4-points congruent sets for robust pairwise surface registration. *ACM Trans. Graphics*, 27(3):1–10, 2008.

[2] F. Bernardini, I. Martin, and H. Rushmeier. High quality texture reconstruction. *IEEE TVCG*, 7:318–332, 2001.

[3] F. Bernardini and H. Rushmeier. The 3D model acquisition pipeline. *Comp. Graphics Forum*, 21(2):149–172, 2002.

[4] F. Bernardini, H. Rushmeier, I. M. Martin, J. Mittleman, and G. Taubin. Building a digital model of Michelangelo's Florentine Pieta. *IEEE Comp. Graph. and Appl.*, 22(1):59–67, 2002.

[5] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. SIGGRAPH*, pages 67–76, 2001.

[6] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH*, pages 303–312, 1996.

[7] J. Davis, S. R. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. *Proc. 3DPVT*, pages 428–438, 2002.

[8] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics (2nd ed. in C): Principles and Practice*. Addison-Wesley Publishing, 1996.

[9] M. Garland and P. Heckbert. Simplification using quadric error metrics. In *Proc. SIGGRAPH*, volume 31, pages 209–216, 1997.

[10] H. Hoppe. Progressive meshes. In *Proc. SIGGRAPH*, pages 99–108, 1996.

[11] H. Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *Proc. IEEE Visualization*, pages 59–66, 1999.

[12] K. Hormann, B. Levy, and A. Sheffer. Mesh parameterization: Theory and practice. In *ACM SIGGRAPH Course Notes*, 2007.

[13] K. Ikeuchi, T. Oishi, J. Takamatsu, R. Sagawa, A. Nakazawa, R. Kurazume, K. Nishino, M. Kamakura, and Y. Okamoto. The Great Buddha project: Digitally archiving, restoring, and analyzing cultural heritage objects. *IJCV*, 75(1):189–208, 2007.

[14] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proc. Eurographics SGP*, pages 61–70, 2006.

[15] 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 *Proc. SIGGRAPH*, pages 131–144, 2000.

[16] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.

[17] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proc. SIGGRAPH*, pages 163–169, 1987.

[18] D. Miyazaki, T. Oishi, T. Nishikawa, R. Sagawa, K. Nishino, T. Tomomatsu, Y. Takase, and K. Ikeuchi. The Great Buddha project: Modelling cultural heritage through observation. In *Proc. VSMM*, pages 138–145, 2000.

[19] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi. Efficiently combining positions and normals for precise 3D geometry. *ACM Trans. on Graphics*, 24(3):536–543, 2005.

[20] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H. P. Seidel. Multi-level partition of unity implicits. *ACM Trans. on Graphics*, 22(3):463–470, 2003.

[21] J. Park and A. C. Kak. 3D modeling of optically challenging objects. *IEEE TVCG*, 14(2):246–262, 2008.

[22] G. Pavlidis, A. Koutsoudis, F. Arnaoutoglou, V. Tsioukas, and C. Chamzas. Methods for 3D digitization of cultural heritage. *Journal of Cultural Heritage*, 8(1):93–98, 2007.

[23] K. Pulli. Multiview registration for large data sets. In *Proc. 3DIM*, pages 160–168, 1999.

[24] 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 *Proc. 8th Eurographics Workshop on Rendering*, pages 23–34, 1997.

[25] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. 3DIM*, pages 145–152, 2001.

[26] R. Sagawa and K. Ikeuchi. Hole filling of a 3D model by flipping signs of a signed distance field in adaptive resolution. *IEEE TPAMI*, 30(4):686–699, 2008.

[27] R. Sagawa, K. Nishino, and K. Ikeuchi. Adaptively merging large-scale range data with reflectance properties. *IEEE TPAMI*, 27(3):392–405, 2005.

[28] J. Salvi, C. Matabosch, D. Fofi, and J. Forest. A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing*, 25(5):578–596, 2007.

[29] P. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[30] M. D. Wheeler, Y. Sato, and K. Ikeuchi. Consensus surfaces for modeling 3D objects from multiple range images. In *Proc. ICCV*, pages 917–924, 1998.

[31] X. Wu. A linear-time simple bounding volume algorithm. *Graphics Gems III*, pages 301–306, 1992.