Eurographics Italian Chapter Conference (2008)
V. Scarano, R. De Chiara, and U. Erra (Editors)

MeshLab: an Open-Source Mesh Processing Tool

P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia

Visual Computing Lab, ISTI - CNR, Pisa Italy

Abstract

The paper presents MeshLab, an open source, extensible, mesh processing system that has been developed at the Visual Computing Lab of the ISTI-CNR with the helps of tens of students. We will describe the MeshLab architecture, its main features and design objectives discussing what strategies have been used to support its development. Various examples of the practical uses of MeshLab in research and professional frameworks are reported to show the various capabilities of the presented system.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation

1. Introduction

3D modeling and editing tools are a quite common nowadays, and the software market offers a very vast set of solutions for many different purpose, targeting the most variable audience, so the idea of developing yet another tool should be evaluated with a bit of care. On the other hand there is still space for specific, sharp tools that offers functionalities rarely available (particularly when they are free). In this paper we present MeshLab the 3D mesh processing system that we have developed with the help of many students in the last two years and that fits in the peculiar niche of advanced research tools useful for a wider audience.

1.1. Motivations

There are many basic motivations for a research institution to start a free or open source software(F/OSS) project. Commonly the most frequent reason is to allow the so called *reproducibility of research results*; in this case a research group releases, under a more or less liberal license, a frozen version of the software system behind a given published work; while this kind of approach should be strongly encouraged by journal and conferences because allows to the reviewer to get a better assessment of quality of the proposed solutions, from a software engineering point of view the release software is often far from being a clean and robust software product. This is not a negative point of view from a scientific point of view, because the purpose of the released software is well different from the one of being a real product.

Another more serious issue of these system is their very short life expectation: maintenance is neither a requirement nor a rewarding activity for a researcher, so in most of the cases these tools, after a few months, often lie in a rather abandoned state.

Another source of research-level F/OSS systems are government founded projects where, in some cases, the funding organization encourages the dissemination of the results under a liberal licensing scheme. In these cases stability and maintenance are guaranteed for the project duration, but at the end of the project the developed software systems face again the risk of being abandoned.

For our system we have decided try to reduce our commitment in the project as much as possible and to leverage on students workforce for the more practical implementing duties. This approach as many positive effects: reducing our commitment left us more time for classical research activity (usually more rewarding from a professional point of view), binding the development of the project with a university course has the consequence of guaranteeing willing work force (the students *has* to complete a piece of MeshLab as part of their course jobs) and it forces us to keep the project updated from year to year.

2. MeshLab general architecture

MeshLab was designed as a general 3D mesh processing system tool with the following primary objectives in mind:

- **ease of use**. The tool should be designed so that users without high 3D modeling skills could use it (at least for the most basic functionalities)
- **deep of use**. The tool should be designed so that advanced users can tweak and extend it by adding functionality and or by modifying all the involved parameters.
- **single mesh processing oriented**. The system should try to stay focused on mesh processing instead of mesh editing and mesh design where a number of fierce competitors already crowd the software arena (notably blender, 3D Max, Maya, and many others).
- **Efficiency.** 3D scanning meshes can easily be composed by millions of primitives, so the tool should be able to manage them.
- **Sustainability.** The project should be able to grow and self sustain itself for at least some years.

As a result MeshLab presents itself a *mesh viewer* application, where a 3D object, stored in a variety of formats can be loaded and interactively inspected in a easy way, by simply dragging and clicking on the mesh itself. MeshLab supports a ever growing variety of 3D formats (all the most commons format are supported) to accommodate the broadest set of users. Once loaded a mesh the user can work on it by mean of a large set of direct parametric filters, that performs unattended automatic task like smoothing, re-meshing or simplifying, or by mean of interactive tools. Figure 1 shows an example of an interactive filter when the user drag the mouse over the mesh a local smoothing is performed on the touched portion of the mesh in real time; in this case the result is that the user is washing out some features of the object. No classical design-oriented features are provided, structured editing of complex scene graphs is not supported by design. Multiple meshes can be loaded together and processed separately or together following a *flat* approach based on *layers*. This approach is, in some sense, a bit similar to the classical raster processing tools that focus on the processing of a bitmap and their "global" composition rather focusing on the pure painting operations.

The mesh processing functionalities that MeshLab currently provide are many and a short, incomplete, high level list of MeshLab characteristic is presented in the following:

- Interactive selection and deletion of portion of the mesh. Even for large models.
- Painting interface for selecting, smoothing and coloring meshes.
- Input/output in many formats:
  - import: PLY, STL, OFF, OBJ, 3DS, COLLADA, PTX, X3D, VRML
  - export: PLY, STL, OFF, OBJ, 3DS, COLLADA, X3D, VRML, DXF
  - Point Clouds support. 3D files that are composed only by points are well supported in PLY and OBJ format.
  - U3D support; MeshLab is the first open source tool to provide direct conversion of 3D meshes into the U3D format. with this format users can create, with just MeshLab and LaTeX, pdf files with 3D objects that can be seen with the current free version of Acrobat Reader.
- Mesh Cleaning Filters:
  - removal of duplicated, unreferenced vertices, null faces, small isolated components
  - coherent normal unification and flipping
  - erasing of non manifold faces
  - automatic filling of holes
- Remeshing filters:
  - High quality edge collapse simplification (even with texture coords preservation)
  - Ball Pivoting surface reconstruction
  - Subdivision surfaces (loop and butterfly)
  - Feature preserving smoothing and fairing filters
- Various Colorization/Inspection filters
  - Gaussian and mean curvature
  - Border edges, geodesic distance, from borders
  - Non two-manifold edges and vertices
  - Self intersecting faces
  - Ambient Occlusion. An ambient occlusion field can be computed and stored per vertex
- Measuring tool. You can take linear measures between points of the displayed meshes
- Slicing tool. A new tool that allows to export planar sections of a mesh in SVG format
- 3D Scanning tools
  - Alignment: ICP based range map registration tool, for putting meshes into the same reference space.
  - Merging of multiple meshes the Poisson surface reconstruction source code
- OpenGL Shader based rendering
- Large rendering. Images up to 16k x 16k pixels can be created for high quality poster printing

2.1. MeshLab Development

One of the MeshLab ambitious goals was to create an open source application that is backed up by a sustainable development model. As previously discussed a common problem of many open source tool is that either they are small enough to be created and maintained by a small team, or, if they require a considerable development effort, the developer team need to be economically supported in some way. Many times the projects originates as research projects that are freed/opened (or simply released under some OSI approved license) at the end (or during) the (financially supported) research project. In these case the big incognito is the long term maintenance of the tool. In some cases the original developers, mostly for ethical and sentimental reasons, continue a low priority maintenance the software correcting small bugs and applying minor modifications, but great improvements are difficult to carry on. A direct consequence of this situation is that many many interesting projects lie in a rather abandoned state.

While we could base the development of the system on a

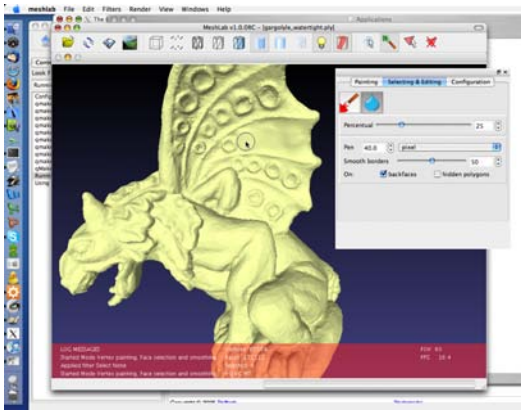

Figure 1: *A typical snapshot of MeshLab in action: the mesh under processing is interactively displayed and the user can work on it by mean of a large set of unattended parametric filters, or by mean of interactive tools, like the one shown in the above image, where the user is smoothing out some features of the object with some simple mouse strokes.*

large stable C++ library for mesh processing task [The04], the whole task of building a consistent interface to present the library functionalities to the user (and add new missing pieces) was considerable.

Very large projects that can afford the support of a large community can rely on the loose sustain of many individuals, but large communities are difficult to be constructed. We decided for the above reasons for a very modular architecture and to subdivide the whole system into as small and as independent as possible pieces. These final pieces were small enough to be assigned, as course tasks, to students of FGT a Computer Graphics course at the Computer Science Faculty of the University of Pisa.

From a practitioner point of view, MeshLab is assembled by a central skeleton framework with almost no capability and a large set of independent plugins that implement all the functionalities of meshlab, from loading the many different file formats, to simple filtering like cleaning and smoothing algorithms to complex interactive tools that directly act over the mesh under the control of the user like the painting tools or the alignment subsystem.

The idea proved successful, motivating the students much more than classical 'mock-up' projects usually faced during courses, moreover the fact that their code would have been actually used in a real project used by real people (we usually show to the students people around the world that discusses MeshLab bugs and workaround) stimulated them to take their task with greater serious attention.

On the other hand not all the students are equal, and for tasks that where assigned to low scoring students, the results are pieces of code that not always was decided that they should be included in the core of the system. And this can cause the some unpredictable delays in the development of some (possibly very important) features.

As a result MeshLab started out in the autumn of 2005 and most of the 60k lines of code composing its core, have been written by willing students as part of their course work; each year students are adding new and new features to the system and they are actually making it to grow at a stead rate. The long list of many contributors to the MeshLab base code is well shown in the home page of the system.

3. MeshLab and Arc3D web service

As an example of the capabilities of MeshLab we briefly report how MeshLab covered a critical role in a publicly accessible framework for the 3d reconstruction from sequence of photos. Two partners of the Epoch Network of Excellence [Epo06], the ESAT-PSI lab of K.U.Leuven (Belgium) and the Visual Computing Lab of CNR-ISTI (Italy) have set up a low-cost 3D reconstruction pipeline to be used in the cultural heritage field. The idea of the designed system is that only a digital photocamera, a pc, and an Internet connection are necessary for a user to reconstruct scenes in 3D. The central part of this approach is a web accessible service, called Arc3D that allows to reconstruct raw 3D data directly from photo sequences, whose results are then processed by MeshLab into clean usable 3D models.

A more detailed description of the complex inner working of this system, and how it *magically* succeed to reconstruct the basic 3D data from simple sequences of photos, can be found in [VVG05]; in this section we mostly describe the second part of the whole pipeline the one that brings to raw reconstructed data to 3D objects. For accessing to the service, downloading the uploading software add trying the reconstruction service visit [Arc07].

3.1. MeshLab processing of Arc3D data

The output of the Arc3D web service is constituted by a set of range maps with texture and a per-pixel quality measure that denotes the number of found matches among all the images. For each range map all the intrinsic and extrinsic camera parameter are available, that makes easy to build, for each range map a simple point clouds with the points placed in the right spatial position.

Like all the data coming from digital acquisition devices they are affected in some measure by various kind of noise and errors. Knowing the kind of issues that are in the acquired data can help to remove or at least reduce them.

MeshLab provide a specific tool for importing the data produced by the Arc3D service. The tool is integrated inside meshlab as an input filter, when the author try to open a ".v3d" file created from the Arc3D the interface shown in Figure 2 is displayed to the user. All the images used for the

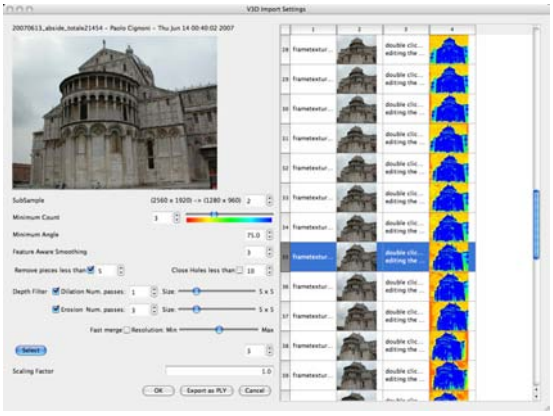

Figure 2: *The MeshLab interface of the importer for photo-reconstrcuted data from the Arc3D service. A synthetic color map depicting the quality of each range map is shown for each reconstructed photo. A number of option allow to fine tune the processing of the chosen range maps.*

reconstruction process are shown on the right with a color coded measure of the quality of the reconstruction process. The user can select some of the images (usually the best ones and /or the ones that cover in the better way the targeted object) and the 3D data generated from the Arc3D is used to build a set of range maps ready to be utterly cleaned inside MeshLab and then integrated in a single clean object. The user can also paint individual masks over the image for discarding portions of the images. A number of parameters are present in the left side to carefully drive the process of converting the raw 3D data into clean high quality range maps. Once the range maps have been imported a set of different operations have to be performed on them (cleaning, merging, aligning) in order to obtain a nice, ready to be used 3D model. In the next paragraphs we discuss some aspects of these processing steps

3.2. Weighted averaging and depth aware triangulation

First of all, some subsampling steps are usually necessary, current digital cameras has very resolute CCD an can easily generate tens of megapixel images. While this kind of definition is useful from the reconstruction point of view because it helps the robustness of the feature matching parts (at high definition even flat surfaces have some kind of texture), when converting these maps to 3D meshes this resolution is often not very useful. Infact the spatial accuracy of the reconstructed data is not uniform along the axis: along the xy image plane there is much more precision than along z depth direction. When you will integrate the various meshes the final quality will be affected by an error dependent from the maximal erorr of the single range-[maps/meshes so it is a good suggestion to subsample the images at the point that the error is more or less uniform along all the directions. Moreover the subsampling process is done

3.3. Outlier management

From a practical point of view it is convenient to consider also procedures for the management of outliers. For a variety of reasons (wrong matching, non correct lighting setup, difficult surface reflectance properties) some of the acquired points can represent ghost surfaces that do not belong at all to the original surface. Luckily enough these points are usually rather isolated, or in other words they do forms small floating pieces far form the main surface. MeshLab provides a couple of automatic filters that allows to cope with these issues, these filters removes the all the small isolated connected components that are smaller than a chosen size expressed either spatially or in terms of primitives composing the component. Figure 3 shows an example of the filters that automatically removes the small isolated pieces that float around each single range map. The same range maps is shown before (left) and after (right) the processing.

3.4. Noise management

Image based acquisition methods suffers of noise problems that are in some way the opposite of the one that happens in traditional active scanning technologies. In fact the most difficult surfaces for passive techniques are the ones that are very flat, uniform and smooth. where the absence of prominent visible distinctive features, make difficult any matching procedure. The direct consequence of this behavior is that flat smooth surfaces can be affected by a considerable amount of noise. On the other hand it is possible to partially cure this problems by analyzing the original images used for the reconstruction process. In fact by detecting the portions of the image that do not posses enough features and marking them it is possible to apply them in a latter stage a heavier smoothing pass.

Moreover sometimes, typically when the photos are not taken in the perfect setting, the boundaries of high jumps region can present larger errors with eventually large and coherent outliers. Also these situations can be detected and corrected almost automatically by searching the boundary regions and applying a erosion-dialation approach that removes one or more strip of pixels from the dangerous jump zones.

Once you have removed all the outliers you have to choose if you should fill all the created holes and interpolate in some way the original mesh, This can be done inside meshlab, that provides some hole-filling filters that can close all the small holes under a given size. Thy applies a variant of the algorithm presented in [Lie03], with various different heuristics for choosing the

Figure 4 shows the combined application of the techniques here described, large outliers and dangerous borders

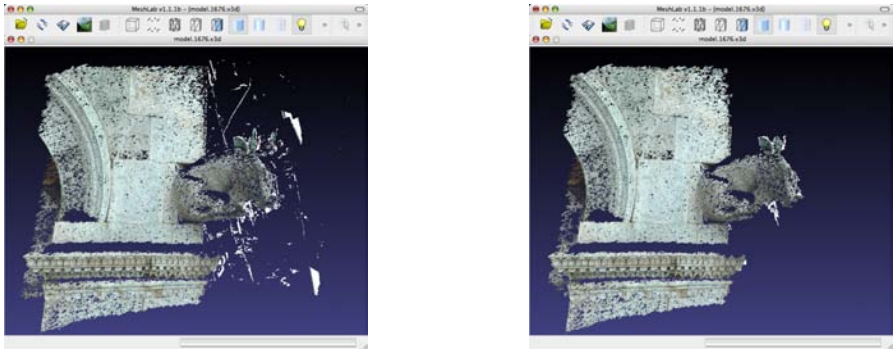

Figure 3: *MeshLab provides filters that automatically removes the small isolated pieces that float around each single range map. The same range maps is shown before (left) and after (right) the processing.*

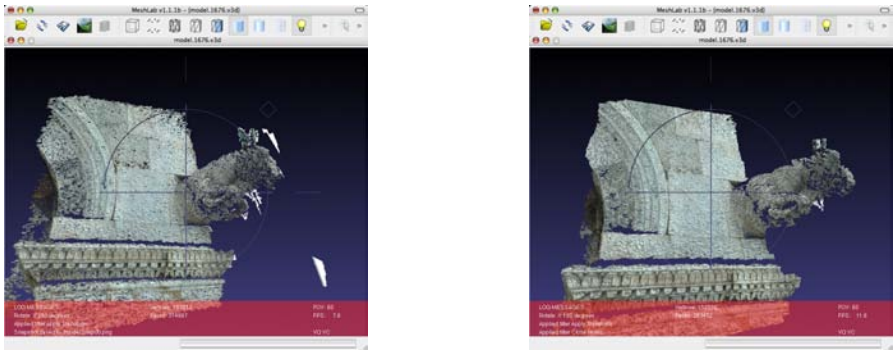

Figure 4: *The figure shows the combined application of erosion dilation approach for cleaning out the depth jumps outliers that sometimes appears. A set of small holes, that remains from outliers removal, is covered automatically by robust holefilling technique.*

(look above and around the bull statue) are removed. A set of small holes (all over the wall), that remains from previous outliers removal, is covered automatically too.

4. MeshLab as a general range map processing tool

One of the interesting characteristic of MeshLab is the presence of all the basic pieces of the 3D scanning software pipeline. Infact beside the rangemap cleaning tools described in the previous sections, that were tailored for the noisy meshes output from multistereo reconstruction techniques, but works quite well for any kind of range maps, inside meshlab there are tools for the subsequent required alignment and merging processing steps.

4.1. Aligning

The aligning process it the step in which you take a set of different range maps of a same object, each one in its onw reference space and you rototranslate them in a single consistently aligned space. Usually in the traditional scanning pipeline [BR00], the aligning step, come before the merging, step but range map from each sequence get out already well aligned from the Arc3D service. The aligning issue arise when you wont to join two or more different sequences. In this case you have to align them together.

The Alignment process adopts a classical approach based on first, a pairwise local and then a global alignment [LR01, Pul99]. The initial placement placement of each mesh is driven by the user and require the selection of four or more points on both the meshes. This initial step is followed by an automatic process that fine tunes the whole alignment . The alignment code used in meshlab is a derivation of the one used in Scanning Tools of the Visual Computing Lab [CCG*03], that has been used in a number of projects.

4.2. Merging

The individual range maps, after having been carefully cleaned and prepared, are ready to be integrated and fused in a single coherent model. All the range maps coming from a same sequence are already well aligned, so the next step is the one of to apply a surface reconstruction filter that fuses

Figure 5: *Three samples of the meshes that can be obtained by using the Arc3D web service reconstruction and the MeshLab system. On the Left, a textured mesh representing one of the two lions in the Loggia della Signoria in Florence. On the center an untextured statue from the Portalada in Ripoll, a large portal in Ripoll, near Barcelona, on the right a reconstruction of the Paris arc du triomphe, untextured for better evaluation of the geometric quality of the geometric shape. The ambient occlusion lighting used to better enhance the shape features was computed with MeshLab.*

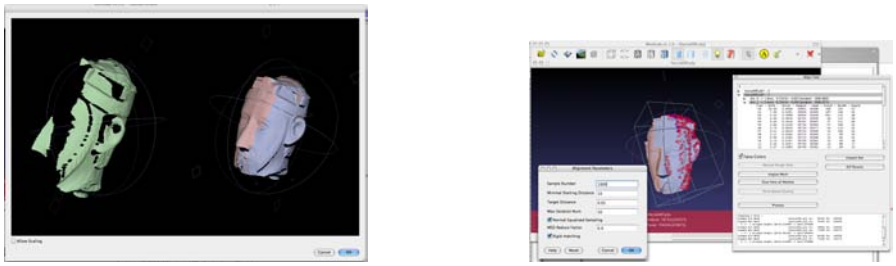

Figure 6: *A snapshot of the MeshLab Aligning tool, that allow to register different range maps, or in the case of data coming from the Arc3D web service, portions of a same object that have been reconstructed from different photo sequences.*

all the meshes in a single new mesh integrating the various parts together in a seamless way. MeshLab offers three different surface reconstruction algorithms. The first one is a interpolatory triangulation filter based on the Ball-Pivoting algorithm [BMR*99], that tries to build a surface connecting all the input points. This kind of algorithms do not work very well in presence of very noisy input data like the one coming from the Arc3D service. The other two are implicit surface approaches that uses the input data to build a implicit representation and then polygonalize it using marching cubes [LC87] variants. One is based on the Poisson surface reconstruction algorithm [KBH06], and it uses part of the original code provided by the authors themselves, and the second one, called *plymc*, developed at the Visual Computing Lab and described in [CCG*03], is an extension of the approach of [CL96]. The Poisson based surface reconstruction algorithm has the very nice characteristic that always build a watertight hole-free surface filling with an interpolatory surface all the missing parts, but currently do not support the color preservation. On the other hand the plymc approach preserves the color during the processing but leaves the unsampled areas as holes in the mesh; from a Cultural Heritage point of view this behavior could be considered a more *safe approach*. Figure 5 shows three samples of the reconstruction process. On the Left, a textured mesh representing one of the two lions in the Loggia della Signoria in Florence, the color was integrated from the many photos directly during the reconstruction. On the center an untextured statue from the Portalada in Ripoll, a large portal in Ripoll, near Barcelona reconstructed using the Poisson surface reconstruction algorithm; a watertight surface was robustly constructed, even if the input data contained only information on the front of the statue, building an interpolating surface even for the back of the model. On the right a reconstruction of the Paris arc du triomphe, done again with the plymc apprach untextured for better evaluation of the geometric quality of the geometric shape. The ambient occlusion shading that is used to better enhance the shape features was computed with MeshLab.

5. MeshLab as an inspection, verification, and assessment system

Last but not least we would remember the many visualization, inspection and healing features of MeshLab. One of the common problem in managing real world 3d meshes is that

common models are often plagued by many different issues, and inconsistencies both topological and geometrical. Typical example of topological issues are the presence of vertices that are not referenced, or duplicated among faces, inconsistency in the face orientation, edge and vertex non manifoldness. For all of them meshlab provide tools for detecting and in many cases removing them. Similarly there are many geometrical issues that can arise frequently, like the plain presence of self-intersecting faces, ghost geometry inside the object, small floating pieces around the mesh, badly shaped triangles, fold-over and other high curvature configurations. Again MeshLab provide tools for detecting these situations. In figure 8 two example of the visualization tools are shown. On the left the mesh is transparently rendered with a color depending on the surface curvature; rendering the mesh with an x-ray effect allow to easily detect the presence of extraneous geometry inside the object itself. On the right, self intersecting face are automatically detected and colored in red.

An interesting unique feature of MeshLab is that it can export meshes in the U3D format. Meshes in this format can easily included in PDF files using pdfLaTeX and the movie15 latex package. Acrobat Reader supports the interactive viewing of these objects since version 7; click on the figure 9 to interactively rotate and zoom the Gargoyle. It is worth noting that the u3d format is quite compact: the u3d model of the displayed gargoyle is 130k, less than the png file used as a placeholder before activating the 3D model or for very old pdf browsers that do not support 3D objects.

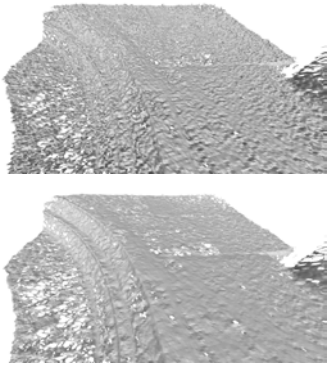

Figure 7: *Smoothing filters can help the process of noise removal. Small bumps due to noise from the acquisition device can be automatically reduced.*

6. Conclusions and Future Works

The system has proved a success over any initial prediction. The last version has been downloaded 10000 times in the first three weeks, there are thousands of users from all the world with at least 600 users that have used it for opening more than one hundred of meshes, Users come from hundreds of universities and renowned commercial firms that have found MeshLab useful in contexts different from the original one of Cultural Heritages.

Acknowledgments We acknowledge the financial support of “EPOCH" (EU Network of Excellence, IST-2002-507382). A warm thank you to the many developers and code contributors that have helped to build the current MeshLab system, to the Arc3D team, and, particularly, to M. Kazhdan and M. Bolitho for the Poisson surface reconstruction code.

References

[Arc07] Arc3d: Automatic reconstruction conduit. More info on: `http://www.arc3d.be/`, 2007.

[BMR*99] BERNARDINI F., MITTLEMAN J., RUSHMEIER H., SILVA C., TAUBIN G.: The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (Oct.-Dec. 1999), 349–359.

[BR00] BERNARDINI F., RUSHMEIER H. E.: 3D Model Acquisition. In *Eurographics 2000, State of the Art Reports Proceedings* (Aug. 24–25 2000), Eurographics Association, pp. 41–62.

[CCG*03] CALLIERI M., CIGNONI P., GANOVELLI F., MONTANI C., PINGI P., SCOPIGNO R.: VCLab's tools for 3D range data processing. In *VAST 2003* (Bighton, UK, Nov. 5-7 2003), D. Arnold A. C., Niccolucci F., (Eds.), Eurographics, pp. 13–22.

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. In *Comp. Graph. Proc., Annual Conf. Series (SIGGRAPH 96)* (1996), ACM Press, pp. 303–312.

[Epo06] EPOCH: The European Network of Excellence on ICT Applications to Cultural Heritage (IST-2002-507382). More info on: `http://www.epoch-net.org/`, 2006.

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. *Proceedings of the fourth Eurographics symposium on Geometry processing* (2006), 61–70.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. In *ACM Computer Graphics (SIGGRAPH 87 Proceedings)* (1987), vol. 21, pp. 163–170.

[Lie03] LIEPA P.: Filling holes in meshes. In *EG/ACM Symposium on Geometry Processing (SGP)* (June 2003), Eurographics, pp. 200–206.

[LR01] LEVOY M., RUSINKIEWICZ S.: Efficient variants of the ICP algorithm. In *Third Int. Conf. on 3D Digital Imaging and Modeling (3DIM 2001)* (May 28th - June 1st 2001), IEEE Comp. Soc., pp. 145–152.

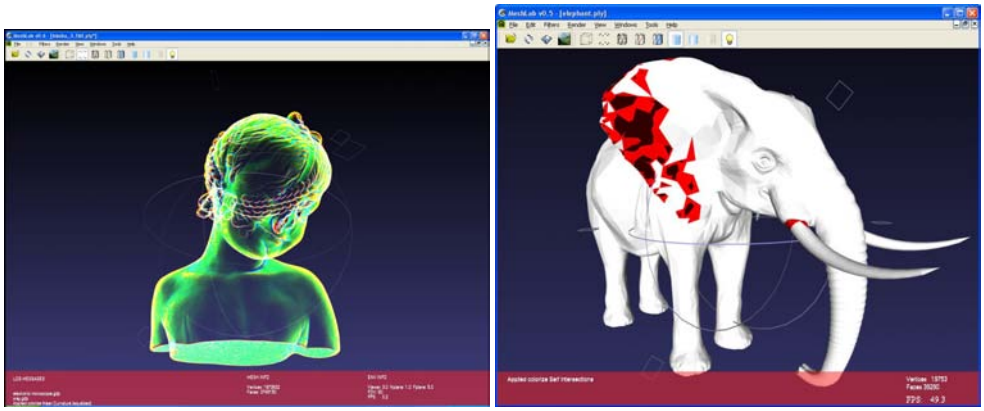

Figure 8: *Some of the many inspection tools inside meshlab. On the left the mesh is transparently rendered with a color depending on the surface curvature; on the right, self intersecting face are automatically detected.*

Figure 9: *MeshLab can export meshes in the U3D format. Meshes in this format can easily included in PDF files using pdfLaTeX. Acrobat Reader supports the interactive viewing of these objects; click on the above picture to interactively rotate and zoom the Gargoyle.*

[Pul99] PULLI K.: Multiview registration for large datasets. In *Proc 2nd Int.l Conf. on 3D Digital Imaging and Modeling* (1999), IEEE, pp. 160–168.

[The04] THE VISUAL COMPUTING LAB - ISTI - CNR: Visual computing library: a gpl c++ library for mesh processing. More info on: `http://vcg.isti.cnr.it/`, 2004.

[VVG05] VERGAUWEN M., VAN GOOL L.: Web-based 3D Reconstruction Service. *Machine Vision and Applications 17*, 6 (2005), 411–426.