Alignment, Merging and Hole Filling Experiments with 3D Range Scans

F. B. ter Haar

R. C. Veltkamp

institute of information and computing sciences, utrecht university

technical report UU-CS-2005-047

www.cs.uu.nl

Alignment, Merging and Hole Filling Experiments with 3D Range Scans

F.B. ter Haar and R.C. Veltkamp
Institute of Information and Computing Sciences
Utrecht University
the Netherlands

February 3, 2006

Abstract

During the last decade, a number of systems and methods for the alignment, merging and hole filling of 3D range scans have been developed. However, a quantitative comparison of the accuracy of resulting 3D models is still missing. In order to make a fair comparison between these systems, we need to determine the parameter settings for each system that optimizes its performance. This work describes the experiments that determine the settings of three systems for the alignment and four systems for the merging of range scans. We also look at the hole filling capabilities of four different systems, with a total amount of six hole filling techniques. During the experiments one set of actual range scans (from a physical object) and one set of 'virtual' range scans (from a 3D model) were used.

Keywords: Range scanning, Registration, Alignment, Merging, Hole Filling

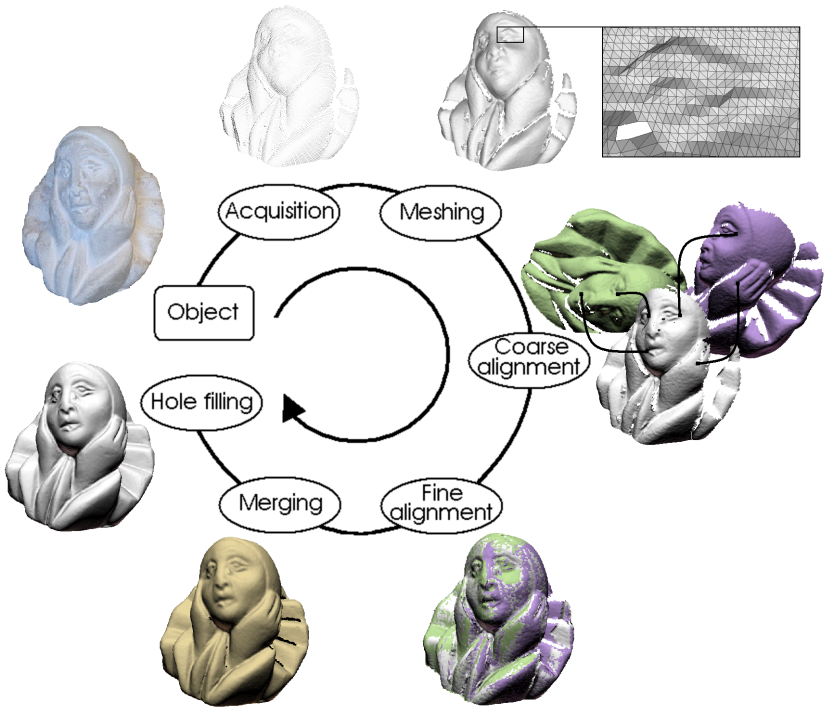


Figure 1: The acquisition and reconstruction pipeline. Systems for the fine alignment, merging, and hole filling of range scans are inspected.

1 Introduction

Various techniques and software systems have been developed to aid the alignment and merging of range scans. Most systems use one of the following types of construction sequences: (a) convert the range scans directly into meshes (which

is often done automatically by the range scanning software) and then perform the alignment and merging of these meshes in order to construct a 3D model, and (b) align the range scans first and then reconstruct the complete surface from the unorganized set of 3D points.

This paper presents parameter setting experiments of systems based on the first type of reconstruction sequence. For the alignment of meshes the following systems are considered: *MeshAlign* [23], *RapidForm* [22] and *Scanalyze* [39]. The systems that are used in the experiments with respect to the merging process are *MeshMerge* [23], *RapidForm* [22], *VripPack* [39] and *Octree Merger (OM)* [23]. Most of these merging system provide a method to fill holes during the merging of the range scans. Hole filling capabilities of the following systems are considered: *MeshMerge* [23], *RapidForm* [22], *VolFill* [39] and *VripPack* [39].

The *alignment* of meshes includes the coarse alignment and the fine alignment. During the *coarse alignment*, a transformation for each of the meshes is found to place them in a common coordinate system in which they are coarsely aligned to each other. During the *fine alignment*, the relative positions of the coarsely aligned meshes are optimized. The coarse and fine alignment both distinguish a *pairwise* and a *multiview* approach. A pairwise approach finds a transformation for one pair of meshes only, while the multiview approach is characterized by finding transformations for all meshes simultaneously.

While the fine alignment is always performed automatically, the *coarse alignment* is performed either interactively or automatically. When the coarse alignment is performed *interactively*, a user decides which pairs of meshes have parts of their surface in common (pairwise). He or she then either selects a few corresponding points on the common surface of two meshes, or manually rotates and translates one mesh towards the other, to bring them into alignment. An *automatic pairwise* approach will try to find a set of corresponding points on two meshes automatically. When enough correspondences are found the two meshes are brought into alignment. A problem with the automatic pairwise approach occurs when incorrect correspondences are selected to align meshes, with an unsuccessful coarse alignment as a result. An *automatic multiview* approach will try to solve this with the use of a global consistency check for all pairs of meshes with high correspondence.

Several techniques have been developed to perform the *coarse alignment* of pairs of meshes *automatically*. The *pairwise* approach includes the exhaustive search for corresponding points [12, 14] and the use of surface signatures such as spin-images [24], point signatures [15], bitangent curves [42] and spherical attribute images [19]. Methods to perform the coarse alignment according to the *multiview* approach include work of Huber et al. [21] and Mian et al. [29]. Huber describes a framework in which spin-images are applied on all possible pairs of meshes followed by the construction of a minimal spanning graph. This spanning graph determines which pairs of meshes should be aligned to construct a globally correct alignment. Mian extracts 3D tensors (grid representations of local surface) from all meshes and constructs a correspondence tree with the mesh having the most tensors as root. The remaining meshes may only be added to the leafs of the tree, and only when the mesh has enough corresponding tensors and passes global verification.

The most popular method for the *fine alignment* of coarsely aligned meshes is the Iterative Closest Point (ICP) algorithm, which was introduced by Chen and Medioni [13], and Besl and McKay [9]. It starts with an initial guess for the relative rigid-body transformation of two meshes obtained from the coarse alignment. Then the algorithm iteratively refines this transformation by repeatedly selecting pairs of corresponding points on the meshes while minimizing an error metric. Because the method operates on one pair of meshes only, we will refer to it as the *pairwise ICP* algorithm. Many variants of the pairwise ICP algorithm have been introduced [36]. The alignment of several pairs of meshes can be performed by applying the pairwise ICP algorithm sequentially to all pairs of overlapping meshes, which may result in the accumulation of alignment errors. To avoid the accumulation of errors, several techniques have been developed to finely align multiple pairs of meshes at once rather than single pairs [7, 30, 33, 40]. The basic goal of these *multiview ICP* methods is to spread the alignment error evenly across the available mesh pairs.

The *merging* of a set of aligned range scans remains a challenging problem. Many techniques have been developed based on either the aligned meshes obtained from the range scans, or based on the point cloud defined by the aligned range scans. The two main approaches for the merging of aligned meshes are surface zippering [41], and the volumetric merge based on a discrete distance field [16]. The volumetric approach has several variants (summarized in [34]). For the construction of a surface out of point clouds (e.g. aligned range scans) popular techniques include the moving least-squares (MLS) surface [2, 4, 26], the use of radial basis functions [11], and ball-pivoting [8].

Portions of the surface that remain unseen by the laser range scanner during acquisition will appear as holes in the merged surface. A large number of methods were developed to fill such holes. These *hole filling* methods can be categorized by the data representation they operate on, defining: the point cloud, the volumetric distance field grid, or the holes and their surrounding surface in a mesh. Methods using a point cloud (e.g. aligned range data) will attempt to create a watertight surface out of the points [3, 5, 11]. Volumetric merge methods construct a discrete distance field from which a merged surface is created. Many hole filling techniques operate on such a volumetric grid [16, 17, 25, 31, 32]. Hole filling techniques based on various properties of the reconstructed surface mesh include [6, 10, 20, 27, 37].

In this paper, we perform parameter setting experiments of alignment and merging systems, using two test objects.

One of the objects is a physical object of which we acquired range scans using the *Roland LPX-250* laser range scanner [35] at a resolution of 0.4×0.4 mm. The other object is a 3D surface model that was reconstructed from 3D range scans. For this model, the range scans were generated by simulating the *Roland LPX-250* in software, with the same resolution but without scan inaccuracies (noise). All range scans were converted into meshes. For the coarse alignment we used an interactive technique. The coarse alignments of meshes were refined using three different systems for the fine alignment (*MeshAlign*, *RapidForm* and *Scanalyze*). The output of *MeshAlign* was then used as input to all four merging systems (*MeshMerge*, *RapidForm*, *VripPack* and *OM*) to reconstruct a surface model. Parameter settings of the alignment and the merging systems were experimentally determined varying the default values and comparing the results visually with the original objects. Results of the hole filling process using *MeshMerge* [23], *RapidForm* [22], *VolFill* [39], and *VripPack* [39] are compared visually as well.

2 Range scan processing

2.1 Objects

We reconstruct the surface of one physical object (*UU-memento*) and one 'synthetic' models (*armadillo*), shown in Figure 2. Models and range scans of the, *UU-memento* are available for download in the AIM@SHAPE Shape Repository [1].
The *UU-memento* is a 110 mm high white painted sculpture. The object has many protrusions (arms and one leg) and inner parts (backs) which are difficult to scan.
The *armadillo* is a reconstructed model of 60 to 70 range scans using techniques described in [16] (345,944 faces). It has many small details, and sharp features such as the ears. This model was downloaded from the Stanford 3D Scanning Repository [39].

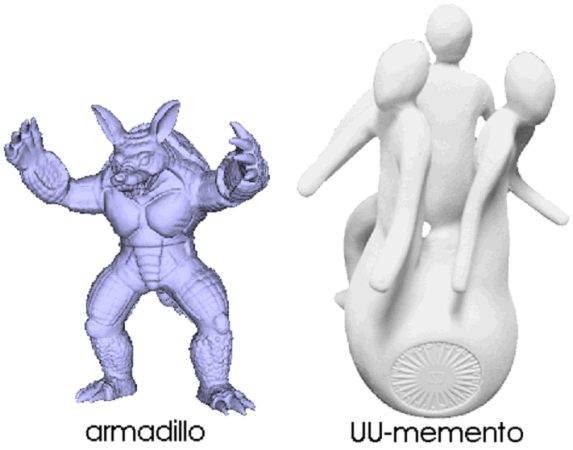


Figure 2: The synthetic model (left) and the physical object (right) used in the experiments.

2.2 Acquisition

To acquire range data for the *UU-memento* we used the *Roland LPX-250* laser range scanner. During the acquisition this scanner projects a laser dot on the object's surface and when the illuminated dot is sensed by the scanner's sensor, accurate 3D point localization can be performed. This 3D point location is determined using an optical triangulation system based on the direction of the laser light, the line of sight of the sensor and the position of the laser with respect to the sensor. Missing data, outliers and noise may occur when the sensor can not properly detect the laser dot due to absorption, reflection, or occlusion. The *Roland LPX-250* generates orthogonal range scans by rotating the laser head and its rotation table both clockwise for a small scope, keeping an equal distance between the laser and a virtual plane perpendicular to the laser direction.

Using the rotation table of the scanner, four range scans are generated rotating the object 90° after each scan, obtaining the front, back, left, and right side of the object. Objects are scanned at a resolution of 0.4 mm in both horizontal and vertical direction. As a result we have a set of four range scans that can be trivially transformed into the same coordinate system. For five different poses of the object, a set of four range scans is generated in a similar way per pose. These twenty range scans will cover most of the object's surface. This process is simulated for the *armadillo* model using the

Parameter description	Parameter	Value
Scan resolution	*res*	0.4×0.4 mm
Threshold angle	t_α	80°
Threshold edge length	t_e	4 × *res*
Threshold patch size	t_p	100 faces

Table 1: Parameter settings used during acquisition and meshing.

same resolution after resizing this models to a height similar to the physical object (100 mm). We did not simulate scanner noise when we created the virtual scans.

2.3 Meshing

In the meshing step, the range scans are converted to triangular meshes by connecting adjacent sample points. Typically the range scans contain noisy data and outliers. These outliers cause incorrect faces in the reconstructed mesh. Other incorrect faces are due to connecting adjacent sample points that should not be connected. To remove most of the incorrect faces, the meshes are cleaned in a similar way as described by Johnson [24]: faces with a normal almost perpendicular to the scan direction are likely to be wrong, so when the angle between a face's normal and the scan direction is larger than a threshold t_α the face is removed. Faces that have an edge longer than a threshold t_e are removed as well. Finally, disconnected vertices and small patches with less than t_p faces are removed. The final meshes are shown in Figures 3 and 4 and values for the parameters used during meshing are listed in Table 1.

2.4 Coarse alignment

The systems for the fine alignment considered in this paper are all based on a variant of the *multiview ICP* algorithm and require an initial coarse alignment of all meshes. As described in the introduction, the coarse alignment can be performed both interactively and automatically. In this experiment we have used an interactive method provided by *MeshAlign* [23] to obtain the coarse alignment of meshes show in Figure 5. We start with the mesh of the object's front view and align overlapping meshes sequentially, using four manually selected correspondence points for the overlapping surface. As a result, all meshes are transformed to the coordinate system of the first mesh, resulting in the coarse alignment shown in Figure 6.

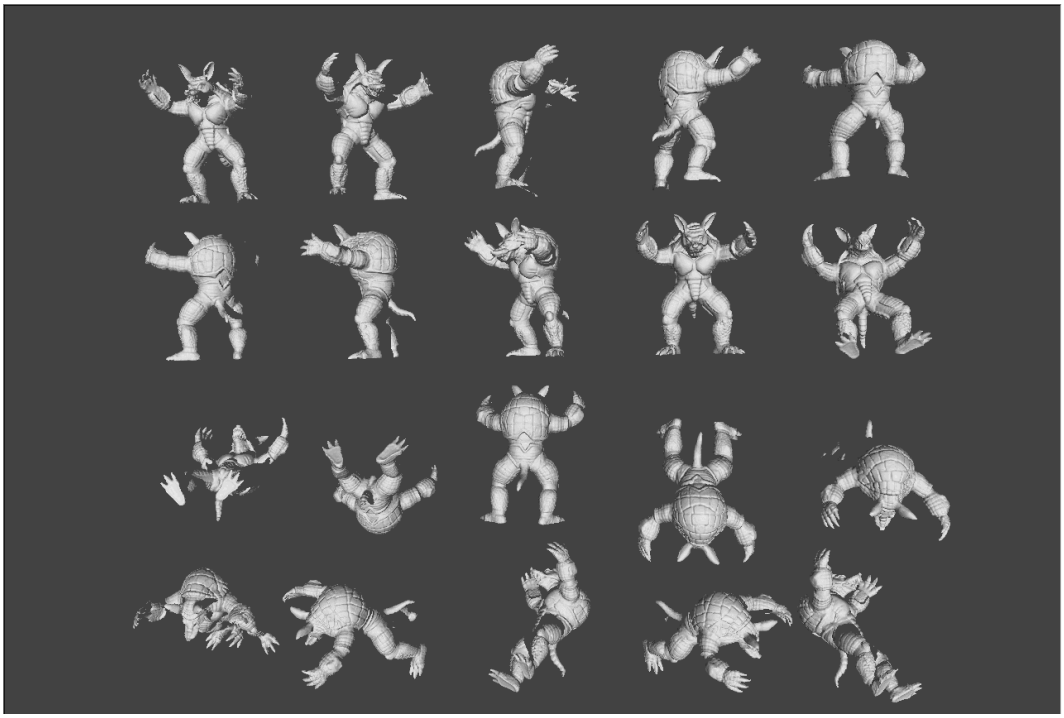


Figure 3: Meshes of the armadillo used in the experiments.

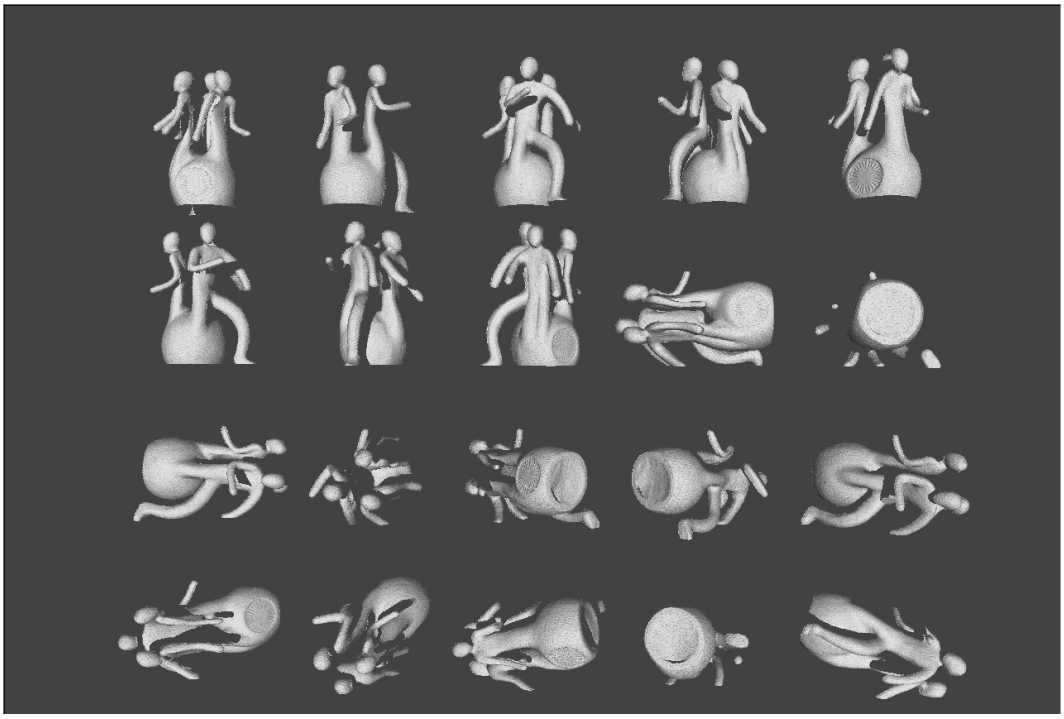


Figure 4: Meshes of the UU-memento used in the experiments.

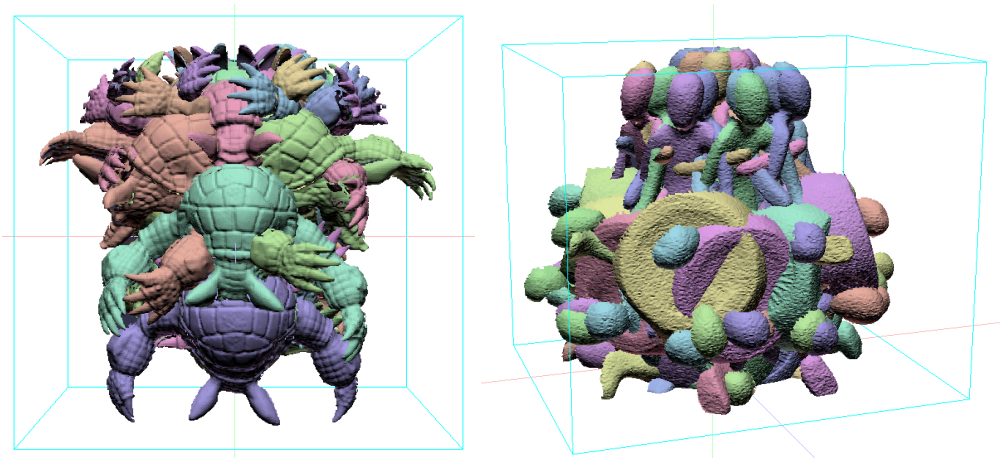


Figure 5: Un-aligned meshes of the armadillo and the UU-memento.

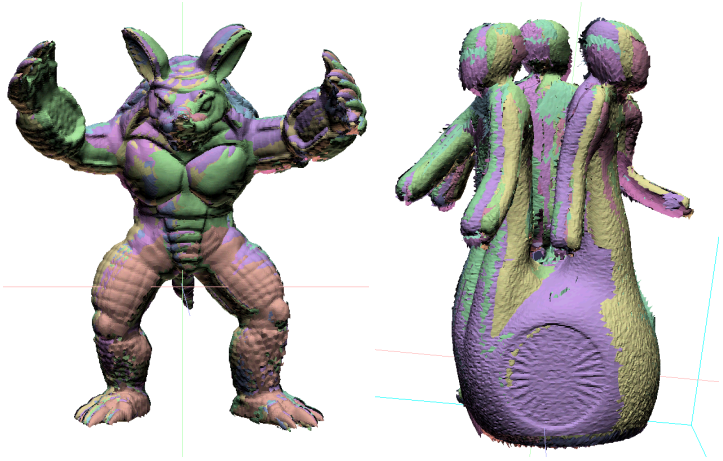


Figure 6: The coarsely aligned meshes of the UU-memento and the armadillo using MeshAlign's interactive method.

3 Experiment: Fine alignment

The coarsely aligned meshes are fine aligned using *MeshAlign*, *RapidForm*, and *Scanalyze*. These systems all use a variant of the multiview ICP algorithm that require a number of parameter settings. These parameters can be classified into (1) parameters that determine the stopping criteria of the ICP algorithm, like a target alignment error or the number of applied iterations and (2) parameters that restrict neighbouring meshes to form a pair, like a minimal amount of mesh overlap. The settings of these parameters influence the performance of the fine alignment of the meshes. What we want to achieve is that the iterative process of the ICP algorithm reaches convergence at an accurate alignment, regardless the running time of the algorithm. This means that our target distance error should approximate zero (0.0001 mm), which corresponds to an accurate alignment. Other stopping criteria like the number of iterations should not hinder the algorithm while trying to reach this target distance error. When the initial (coarse) alignment of some of the mesh pairs is too poor, the second set of parameters (like the minimal amount of mesh overlap) might restrict the algorithm to include these meshes in the fine alignment process.

To increase the chance of the heuristic ICP algorithm to obtain a correct fine alignment, we apply the ICP algorithm on a set of range scans twice (sequentially) using different settings. During the first run, the coarse alignment of all meshes is improved by using little restrictions on the selection of mesh pairs. Then we use the output alignment of the first run as the input alignment for the second run. The goal of the second run of the ICP algorithm is to converge the alignment to an optimal one. Therefore, we use more iterations, a larger number of point-pair samples, and more restrictions on the mesh pair selection, because barely overlapping mesh pairs can have a negative influence on the total alignment error.

It is hard to determine the optimal parameter settings for the fine alignment systems, because the heuristic ICP algorithm has a different outcome each time it is applied. The goal of this experiment was to determine settings for the first run, which enables the system to improve the coarse alignment of the range scans and settings for the second run that will

enable the system to reach a very accurate alignment. New parameter settings are selected based on the default values and their function in the ICP algorithm.

3.1 MeshAlign

MeshAlign (v.2) is a system developed by ISTI-CNR [23] to perform the coarse and fine alignment of meshes. A multiview ICP algorithm as described by Pulli [33] is used for its fine alignment. *MeshAlign* creates arcs between pairs of meshes when they are considered 'overlapping' according a number of parameters, such as a certain percentage of grid overlap. For *MeshAlign* we used a lot of default settings because of their limited influence on the alignment process. These default settings include: max point number = 10000, min point number = 30, ug expansion factor = 10, low-pass filter = 0.05, high-pass filter = 0.75, reduce factor = 0.9, maximum shear = 0.5, maximum scale = 0.5, rigid matching mode, and normal based sampling. The parameter settings of this system that were improved are shown below. When the fine alignment was applied using the default settings, often one or two coarsely aligned meshes of the *UU-memento* were excluded from the fine alignment process. So the goal was to select settings for which all meshes were aligned correctly.

As described above, we select a set of settings for each of the two runs. Settings for the first run are used to improve the coarse alignment of all the meshes. Afterwards, this improved alignment is further improved during the second run using a different set of settings. The use of a relatively large *min distance* and *max angle* reduces, for instance, the chance of excluding meshes from the fine alignment process. Since the accuracy during the first run is less important, the number of *selected samples* of a mesh and the *number of iterations* are set relatively low. During the second run it is important to obtain an accurate alignment. Therefore, the number of selected samples and the number of iterations are increased. The *min distance* and the *max angle* are reduced to use only the accurate correspondences. Increasing the *end step number* increases the chance of an accurate convergence. The default value of *min mindistabs* restricts the convergence of the ICP algorithm too much, higher accuracy was achieved using a value of zero. With the use of the new parameter settings all meshes of both the *UU-memento* and the *armadillo* were aligned correctly.

Selected settings:

Settings MeshAlign	default	1st run	2nd run
Grid overlap (%)	20	10	20
Sample number	2,000	2,000	7,000
Min distance (mm)	10	20	2
Max angle (degrees)	45	45	30
Iteration number	50	30	150
Target distance (mm)	0.05	0.0001	0.0001
End step number	5	15	15
Min mindistabs (mm)	1	0	0

3.2 RapidForm

RapidForm (2004 PP2) is a commercial system developed by INUS-Technology [22] able to perform both the coarse alignment, fine alignment and merging of a set of meshes, as well as many other 3D modelling operations such as hole filling. *RapidForm* has only a few parameters concerning the fine alignment that need to be set. They advise to use minimal ten iterations and the removal of outliers. Since time is not in issue in our experiments, we have used thirty iterations instead. During the fine alignment outliers may disturb the ICP algorithm. Therefore, the option to exclude outliers during the alignment is selected.

Selected settings:

Settings RapidForm	default	1st run	2nd run
Max number of iterations	20	30	30
Max average deviation (mm)	0.01	0.0001	0.0001
Removal of outliers	yes	yes	yes

3.3 Scanalyze

Scanalyze (1.0.3) is a software distribution developed by Stanford's Computer Graphics Laboratory [39] for the coarse and fine alignment of meshes. For the fine alignment, this system can use one of the variants of the pairwise ICP algorithms described in [36]. It automatically fine aligns all meshes by optimizing the parameters of the multiview ICP algorithm [33]

while it iteratively aligns neighbouring meshes. For this system we will only use its multiview ICP algorithm, because we want to align all meshes simultaneously.

Since Scanalyze automatically fine aligns all meshes by changing the parameters of the global ICP algorithm while it iteratively aligns neighbouring meshes, only a few initial settings are required. By selecting a high error threshold and a high number of target pairs during the first run a first robust alignment is obtained. The use of a relatively low error threshold and even a higher number of target pairs during the second run an accurate final alignment is ensured for our range scans.

Selected settings:

Settings Scanalyze	default	1st run	2nd run
Error threshold (mm)	5	10	2
Target number of pairs	200	20000	40000
Normal space sampling	no	yes	yes
Convergence tolerance (mm)	0.01	0.0001	0.0001

3.4 Results

In Figure 8 the final alignments of meshes are shown. These results show accurate alignments of meshes for all systems by means of their 'splotchiness'. Splotchiness is referred to as the visual interpenetration effect of (coloured) surfaces. In the work of Silva et al. [38] they explain the occurrence of this effect:

> 'The interpenetration effect results from the nature of real range data, which presents slightly rough surfaces with small local distortions caused by limitations in the precision of the acquiring sensor or by noise.'

Thus, when two different range scans of the same object are well aligned, the small local distortions will make it possible to see the colour of both range scans (Figure 7). In Figure 8, a small difference in splotchiness of the surfaces can be seen at the right most head of the *UU-memento* and the colour of the chest of the *armadillo*. However, it is impossible to determine visually which alignment is most accurate.

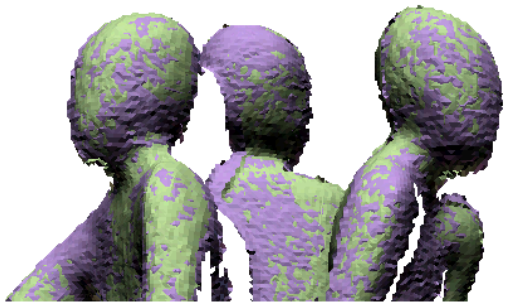


Figure 7: A good spotchiness: the interpenetration of the two surfaces makes it possible to see both colours.

Figure 8: The results of the fine alignment. From left to right the results from MeshAlign, RapidForm and Scanalyze. The chest of the armadillo and the head of the memento show a difference in splotchiness.

4 Experiment: Merging

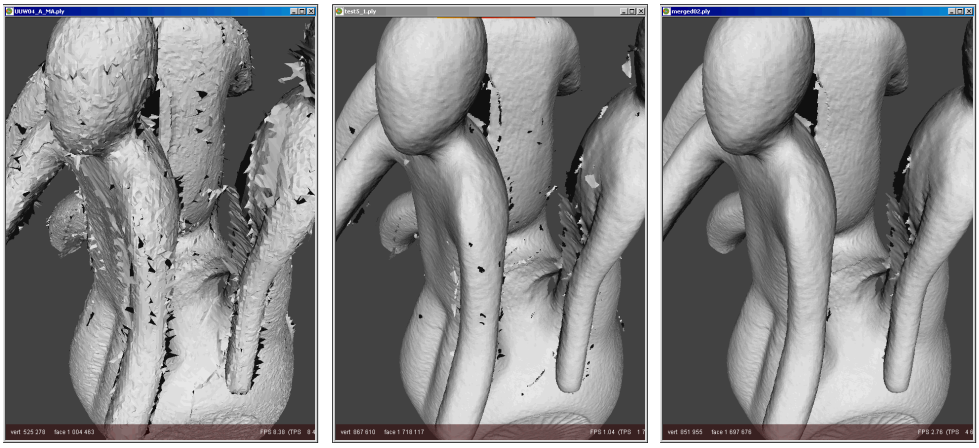
Since the alignment of meshes is performed accurately by all three alignment systems, we perform the merging experiments on the finely aligned meshes by only one, we have chosen *MeshAlign*. The merging is performed using the systems described below. Three out of four systems (*MeshMerge*, *RapidForm*, and *VripPack*) are based on a volumetric distance field grid in which geometric information of the meshes is stored. The volumetric merging of meshes always requires a predefined resolution of the voxel grid. Since the range scans were created using a scan resolution of 0.4×0.4 mm, proper voxel sizes would be in between 0.2 mm^3 and 0.4 mm^3. In this merging experiment we use the latter one and check if the obtained parameter settings can be applied during the merging process using a volumetric grid with 0.2 mm^3 sized voxels as well.

Once we determined the optimal parameter settings for a merging system, we created final merge models for the *armadillo* and *UU-memento*. After the merging of the *UU-memento*, we retained only its largest connected surface component. This was necessary to remove the noisy patches that got separated from its 'main mesh' (Figure 9).

4.1 MeshMerge

MeshMerge [23] is a tool that merges range scans based on a volumetric distance field approach. *MeshMerge* builds a carefully weighted distance field for each range map, and blends all the distance fields together in a seamless way in a single volumetric representation. The final surface is reconstructed through the standard Marching Cubes algorithm [28]. For *MeshMerge* the following parameter settings were determined:

- **-V#** The required voxel size in mm^3.
- **-w#** The distance threshold of the distance field expansion in voxels. Only overlapping parts of different range scans closer than this threshold are merged. (default 3 voxels)



(a) The alignment of (noisy) meshes (b) The result of a merging process (c) The final result after cleaning

Figure 9: The removal of disconnected (noisy) patches after the final merging process.

- **-R#** The number of refilling steps that is performed after the merging of range scans to fill small holes. (default 1 step)
- **-L#** The number of smoothing passes after each expansion of the distance field. (default 1 pass)
- **-a#** The angle threshold of the distance field expansion. (default 30 degrees)

The required size of the voxels in the voxel grid define the final mesh resolution, which we want to equal a resolution of $0.4 \times 0.4 \times 0.4$ mm.
Selected setting: Voxel size is set to 0.4 mm^3, -V0.4
The distance threshold of the distance field expansion has a default value of three voxels. For this parameter we checked the values: 3, 2, 1 and 0.8 voxels. It turned out that the use of a threshold larger than one voxel tends to create blobs out of sharp features (such as the ear of the *armadillo*), while a threshold smaller than one voxel results in many (tiny) holes (Figure 10).
Selected setting: A distance threshold of one voxel is selected, -w1
The number of refilling steps has a default value of one refill step. Checked values include: 0, 1, and 2 steps (Figure 11). Refilling the volume after the merging has completed helps to fill tiny holes correctly (tiny hole in the ear of the *armadillo*), but more often the mesh is incorrectly expanded. Especially when the holes are large and many refilling steps are applied, then the surface gets distorted and even tube-like surfaces can be created (rightmost elbow of *UU-memento*) .
Selected setting: No refilling steps, -R0
By default, one smoothing step is applied during the surface reconstruction. After the application of 0, 1, and 2 smoothing passes the default setting of one smoothing pass was selected (Figure 12). The difference between one and two smoothing passes is very small, without smoothing the final mesh tends to be less good.
Selected setting: One smoothing step after each expansion of the distance field, -L1 (default)
Another setting to influence the expansion of the distance field is the angle threshold. The default value of this setting is thirty degrees. Geometric information in the distance field grid is merged if the angle threshold is below this setting. Trying to constrain (possibly incorrect) distance field expansion we look at angle thresholds of: 30, 25, 20, and 15 degrees (Figure 13). However, the use of a threshold equal or below twenty degrees results in meshes with many holes. The use of threshold values equal to twenty-five and thirty degrees are preferred. We have selected the latter, since this is the default value.
Selected setting: Angle threshold of thirty degrees, -a30 (default)
All of the above settings are independent of or scalable with the voxel size. So the selected parameters were expected to perform well if a voxel size of 0.2 mm^3 is used instead of 0.4 mm^3. This was confirmed by the final results obtained using this voxel size of 0.2 mm^3 (Figure 14). Note that the noisy patches that got separated from the main mesh were removed from these models.

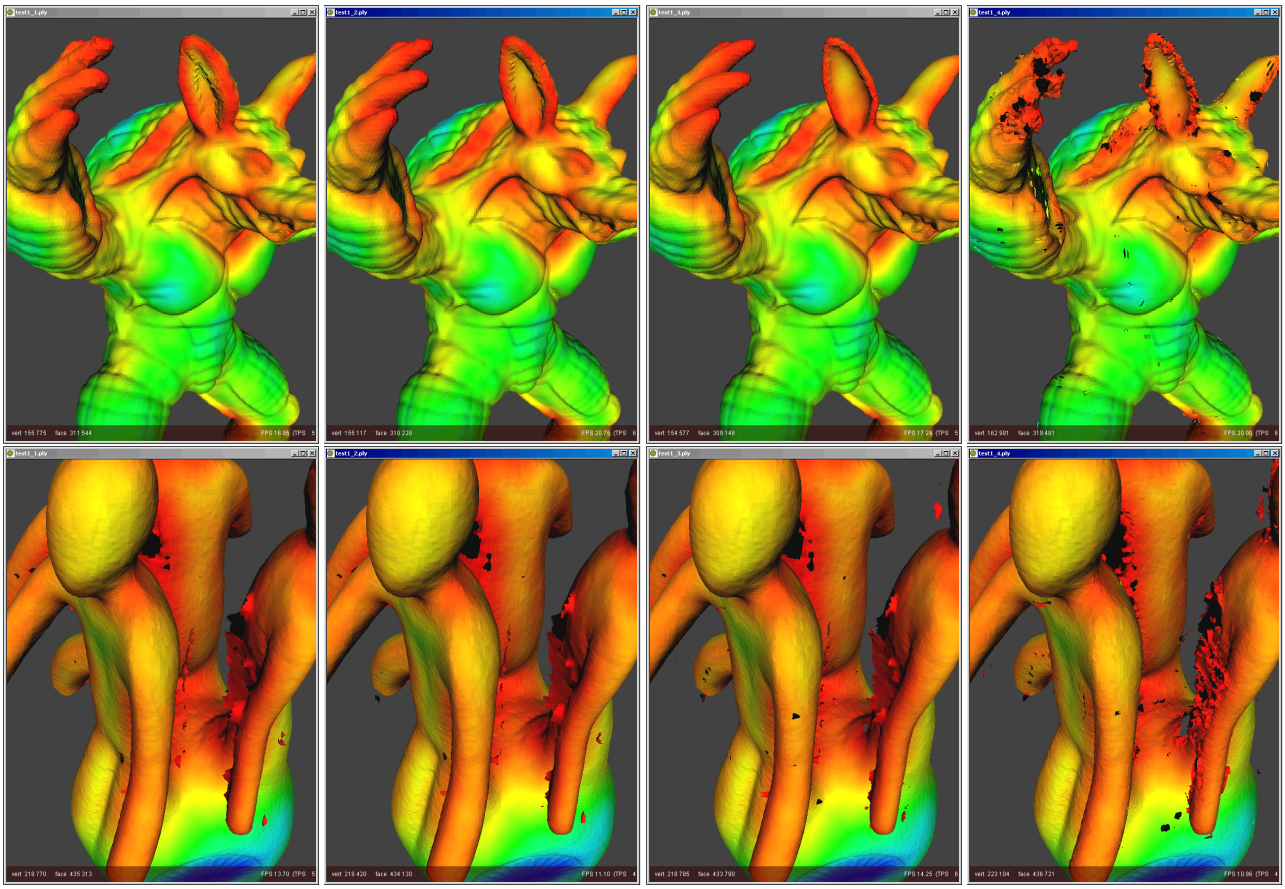


Figure 10: MeshMerge results while varying the distance threshold of the distance field expansion. Left to right: 3, 2, 1, 0.8 voxels, a value of 1 is preferred.

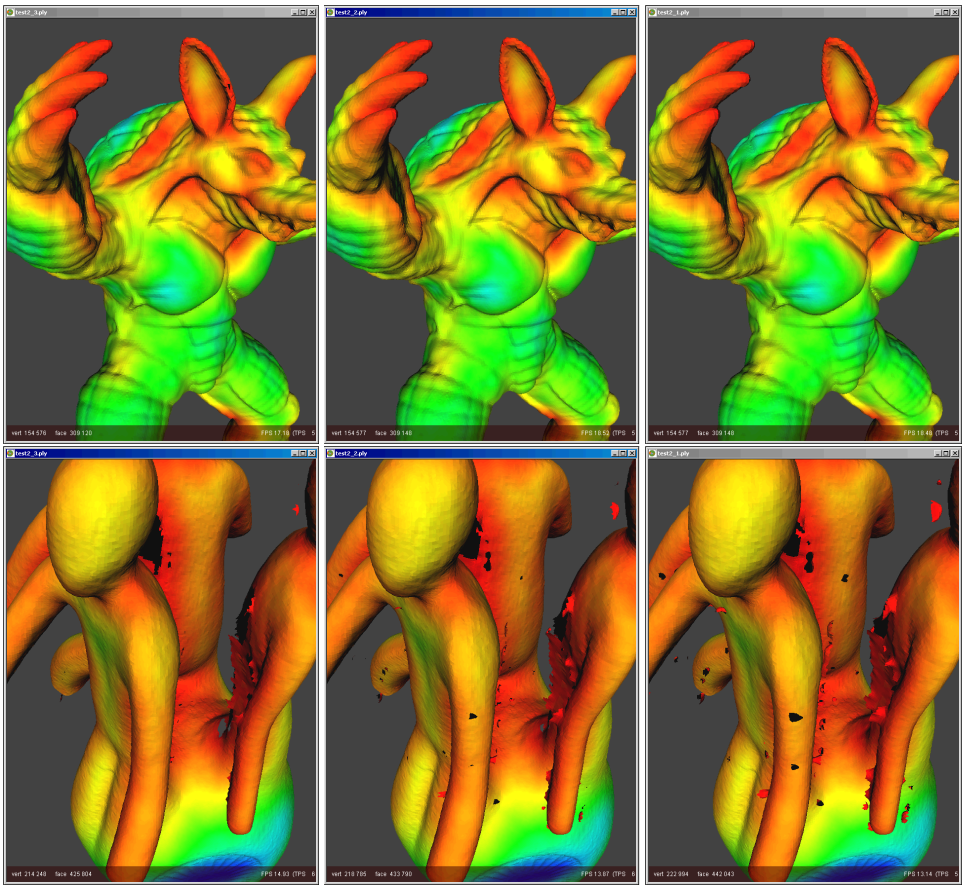


Figure 11: MeshMerge results while varying the number of refilling steps. Left to right: 0, 1, 2 refilling steps. No refilling steps are preferred, because it prevents (possibly incorrect) expansion of the surface.

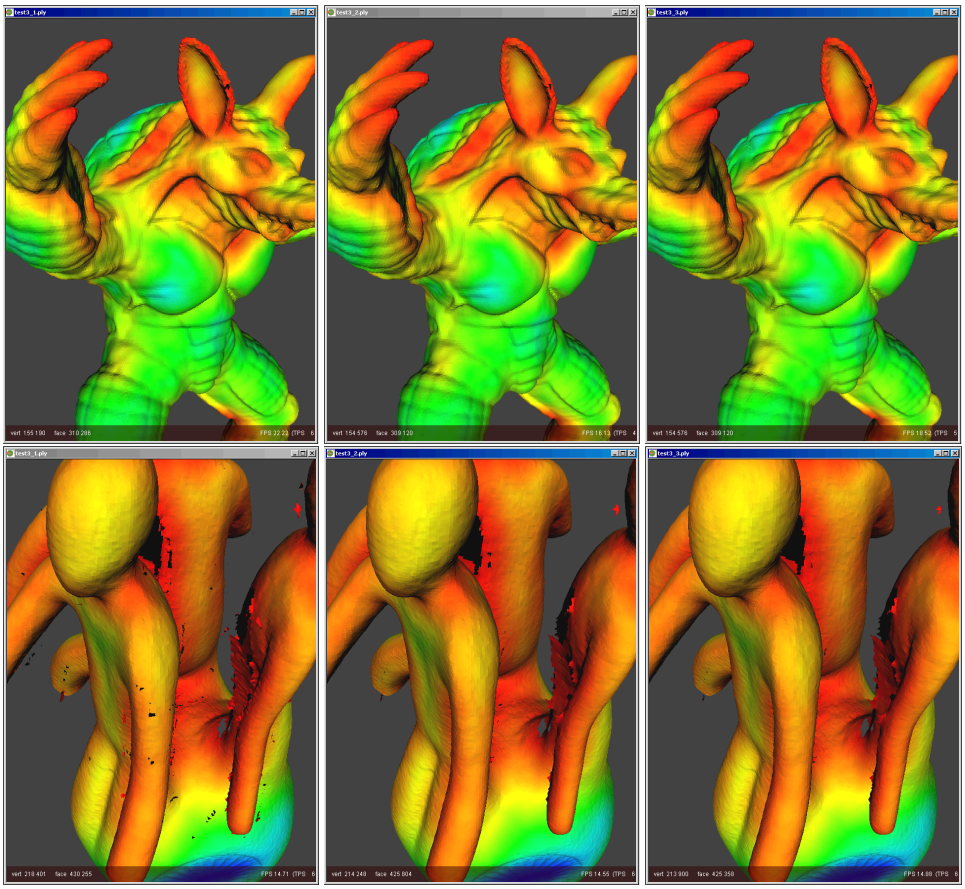


Figure 12: MeshMerge results while varying the number of smoothing passes. Left to right: 0, 1, 2 smoothing passes. The use of a single smoothing pass is preferred.

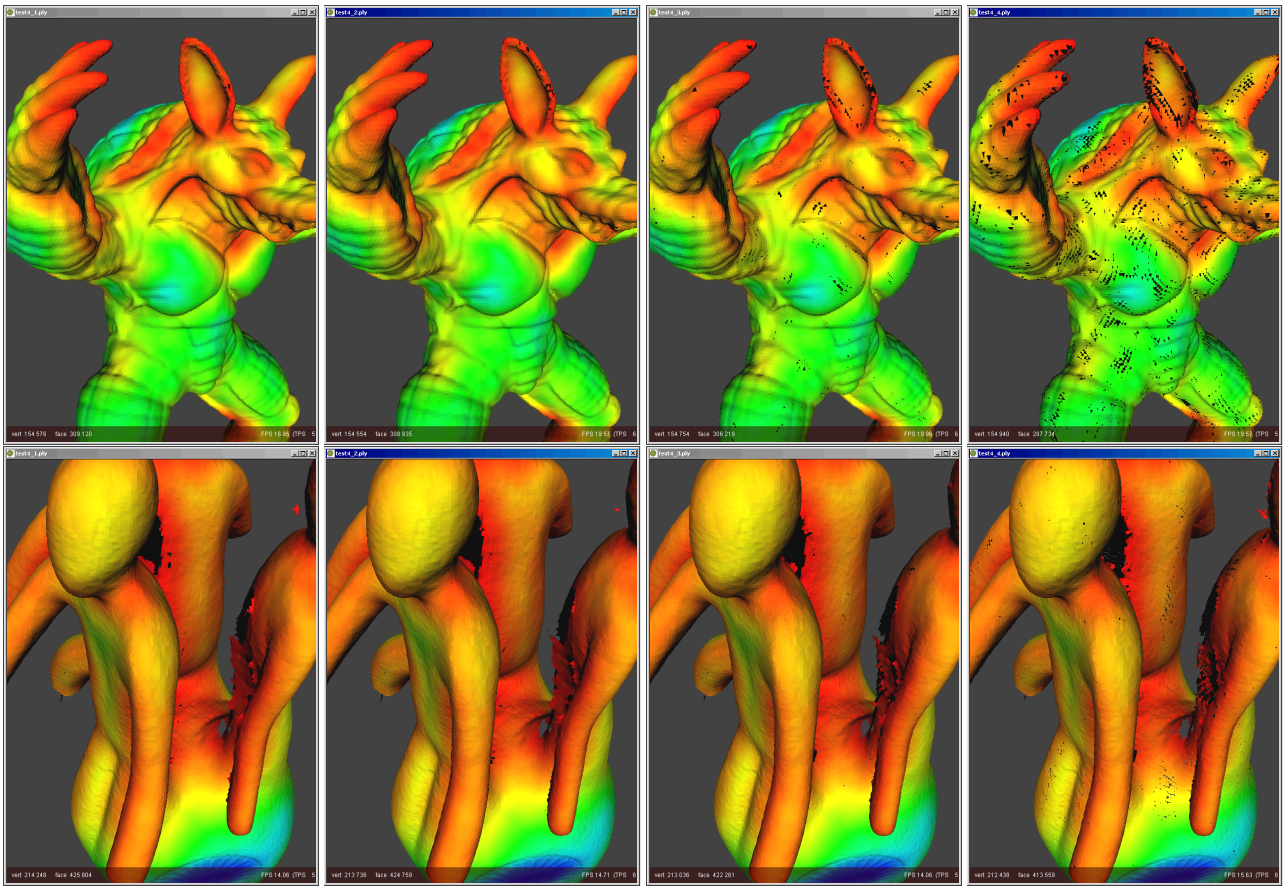


Figure 13: MeshMerge results while varying the angle threshold for the distance field expansion. Left to right: 30, 25, 20 and 15 degrees, a 30 degree threshold is preferred.

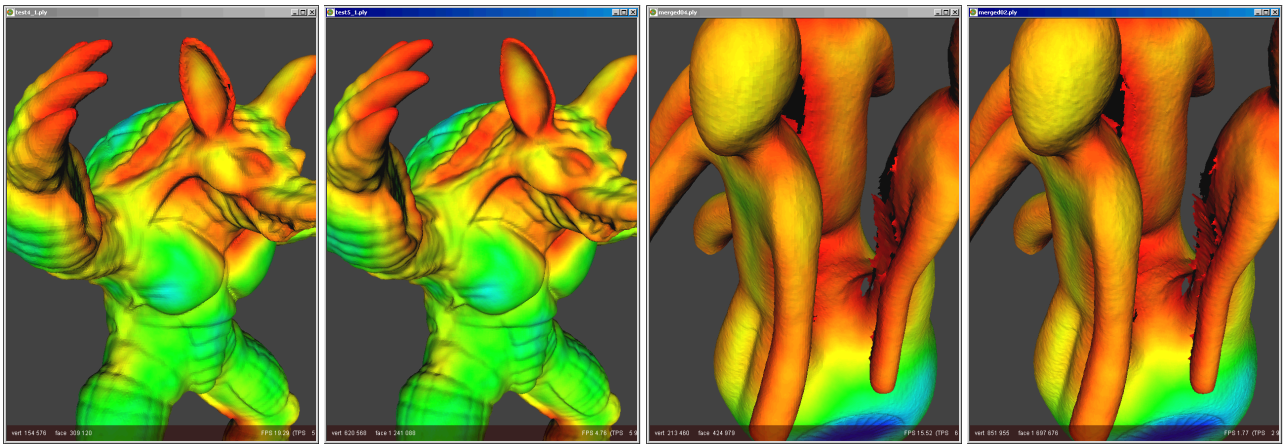


Figure 14: The final MeshMerge models. From left to right: The merged model at a resolution of 0.4 mm^3 and the merged model at a resolution of 0.2 mm^3 using the determined settings.

4.2 RapidForm

RapidForm [22] provides two kinds of merging techniques: surface zippering and volumetric merging. Surface zippering is a technique developed first by Turk and Levoy [41], this technique removes redundant surfaces, zippers adjacent meshes, and optimizes the faces in the zippered areas. The volumetric merging allocates the geometry information of the range scans to a volumetric grid and applies a Marching Cubes algorithm.

The surface zippering approach requires a distance criterion that determines whether two meshes are zippered or not. This distance criterion can be determined manually, but we selected the automatic distance criterion (which is the default). The volumetric merge approach requires the resolution of the volumetric grid. The 'resolution' of *RapidForm* is the ratio of the average vertex spacing. To acquire a final mesh resolution of 0.4×0.4 mm a 'resolution' of 0.72 needs to be selected and 0.36 to obtain a mesh resolution of 0.2×0.2 mm. The user can also decide to use quick projection, which assumes that all vertices of a mesh can be projected onto a plane without overlap (which is the case for range scans), and to optimize the final mesh. Since *MeshMerge* did not optimize the final mesh, the latter option is not used. For this system no settings have to be determined and therefore only the final results of the merging process are included below (Figure 15).

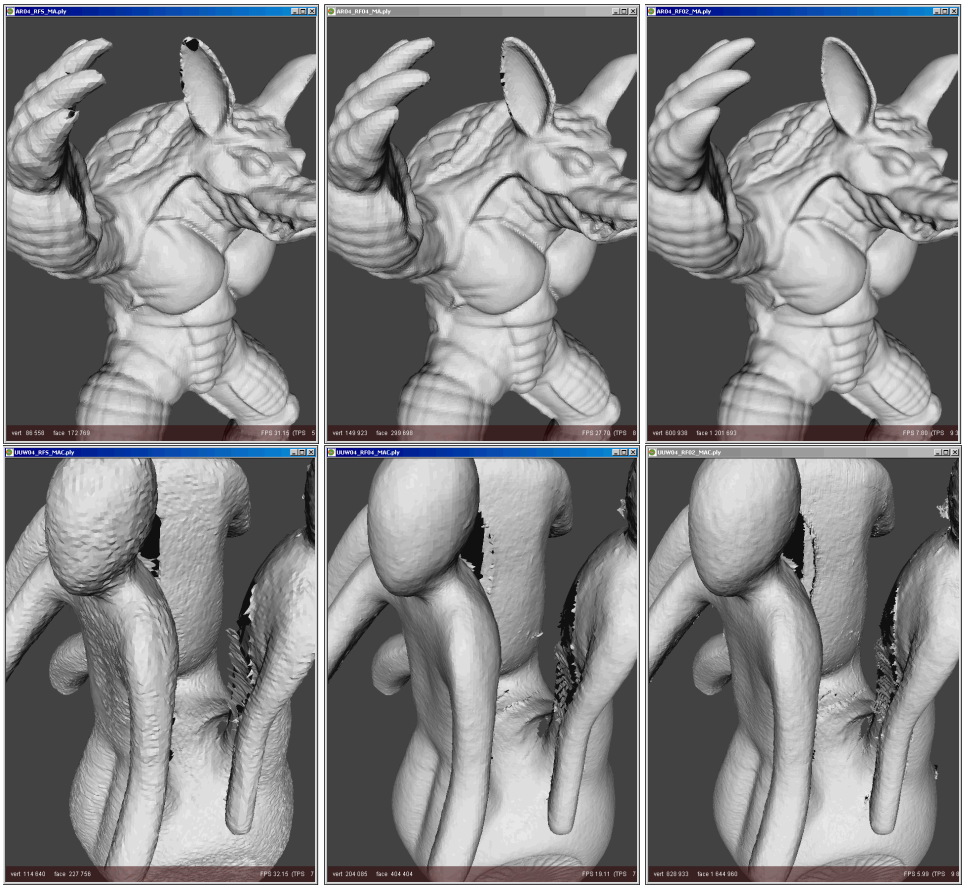


Figure 15: The final RapidForm models. From left to right: The merged model using surface merge, the merged model at a resolution of 0.4 mm^3 and the merged model at a resolution of 0.2 mm^3.

4.3 VripPack

VripPack [39] is an implementation of the volumetric approach described by Curless and Levoy [16]. The nodes of the volumetric grid store the weighted signed distances of the grid nodes to the nearest range scans along the line of sight of the sensor. The final surface is extracted from the volumetric grid using the Marching Cubes algorithm. For *VripPack* the following parameter settings were determined:

- **-ramplength #** The length of the distance ramp that determines the amount of distance field expansion.
- **-use_bigger_bbox** A slightly bigger bounding box (than the one provided) is used.
- **-carve_#** The carving technique applied on the volumetric distance field grid.

The ramplength determines the degree of distance field expansion. When the selected ramplength is too small, the volumetric distance field has too few data to extract a surface from. The result is an extracted surface with many small holes. On the other hand, a large ramplength leads to more interference of opposing surfaces during the distance field expansion (Figure 16). Ramplength settings of 3, 2, 1, 0.8 voxels are tested and a ramplength of one or two voxels turned out to perform best. During the rest of the experiments we have used a ramplength of two voxels.
Selected setting: A ramplength of two voxels is selected, -ramplength 2
The bounding box of all aligned meshes is used together with the predefined resolution, to create a three dimensional voxel grid in which the merging process is performed. The use of a bounding box which is too tight, may limit the initialization of the distance field which can result in the loss of data at the borders of the volumetric grid. (see Figure 17)
Selected setting: A bigger bounding box is preferred, -use_bigger_bbox
VripPack provides different levels of space carving. The main idea of space carving is to determine areas of the volumetric grid that should remain empty. The information that is used to determine the empty regions comes from the laser range scanning process. When a laser ray is projected onto an object, we know that the area in between the laser and the object is empty and that the area behind the object (in the direction of this ray) remains unseen by this ray. The voxel grid is initialized in the 'unseen' state. Then meshes are added with the predetermined ramplength to assign 'near the surface' areas for the distance field expansion. Then the 'empty' regions are defined by 'carving' backwards from the observed surfaces [16]. Despite the various space carving options like: -carve_conservative, -carve_aggressive, and -carve_only, no difference in the final meshes could be observed, even the number of vertices and faces were exactly the same. Therefore, we have compared only the results of conservative space carving and no space carving at all. Figure 17 shows slices with and without space carving of the volumetric distance field grid and Figure 18 shows the resulting models.
Selected setting: No significant difference, use space carving only when hole filling should be applied
Finally the selected parameters are applied using a higher resolution voxel grid. Figure 19 shows the results using a voxel grid with a resolution of both 0.4 mm^3 and 0.2 mm^3.

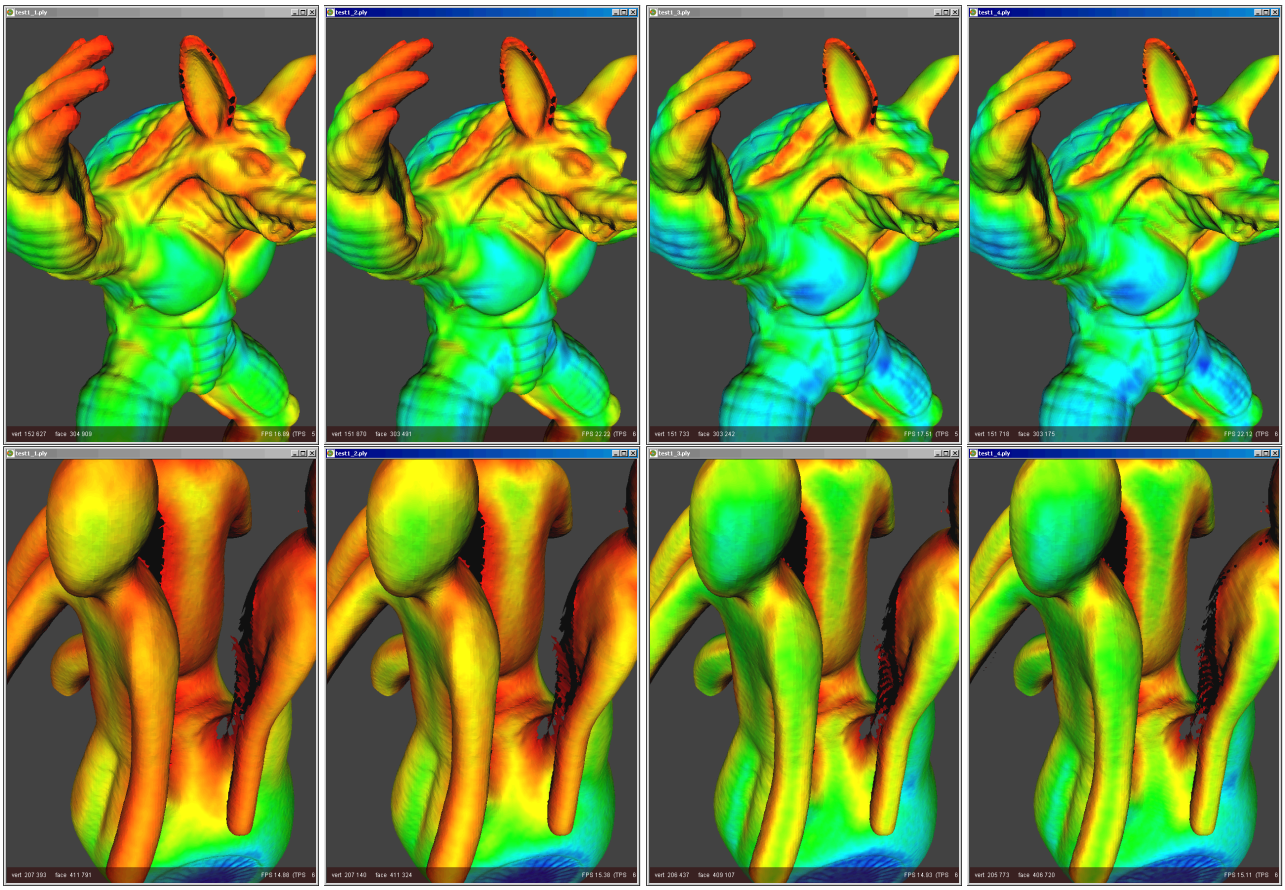


Figure 16: Results VripPack while varying the ramplength of the distance field expansion. Left to right: 3, 2, 1, 0.8 voxels, a value of 2 is preferred, because it preserves the intended surface best.

Figure 17: Slices of VripPack's volumetric distance field grid created for the armadillo (head and arms), showing the signed distance and weight functions in 2D (brown=unseen, black=empty, [black, white]=near the surface mesh). From left to right: A slice with tight bounding box, a slice with a larger bounding box, and a slice showing space carving.

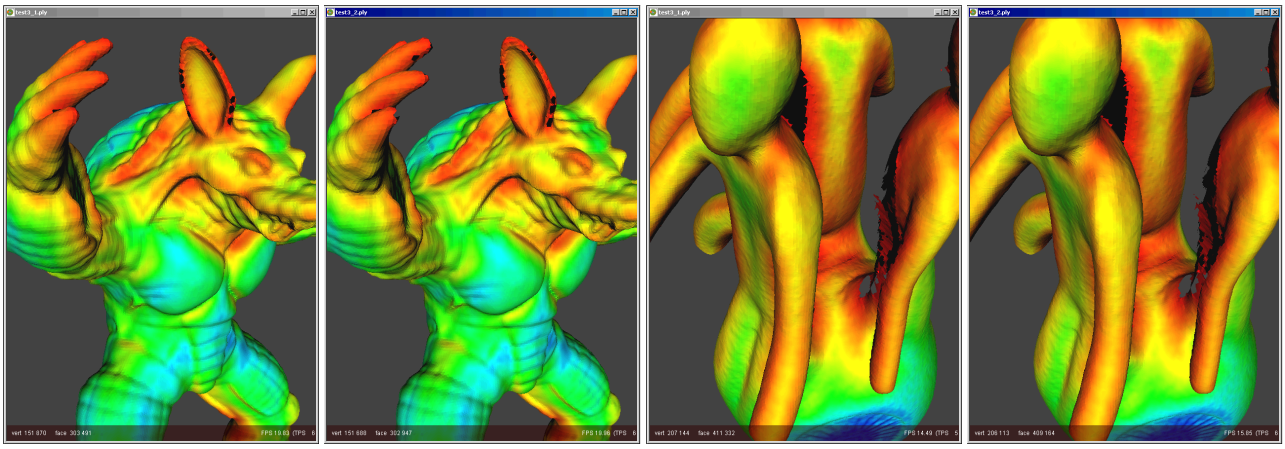


Figure 18: Results of VripPack when the space carving option is (not) used. Left: no space carving. Right: with space carving. The space carved images show slightly larger holes.

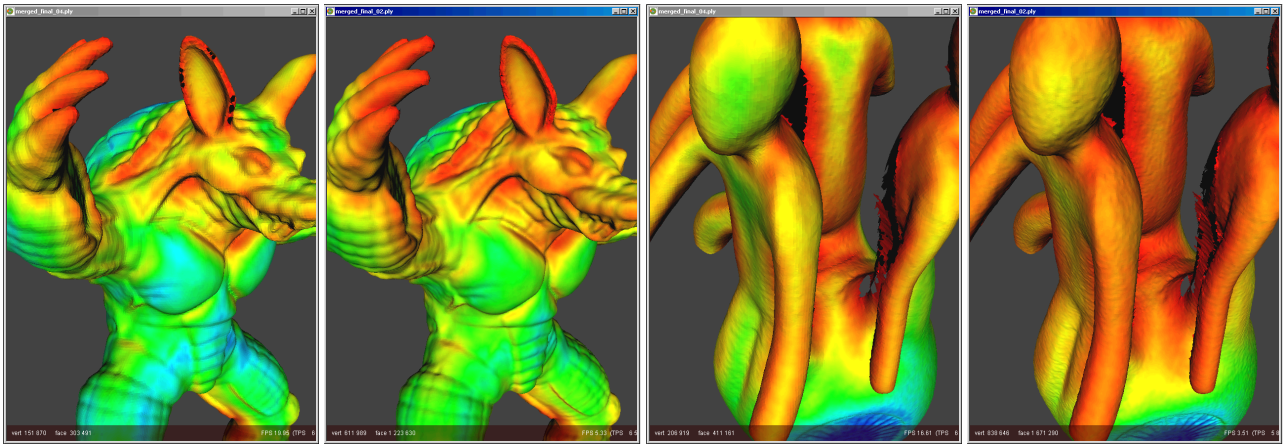


Figure 19: The final VripPack models. From left to right: The merged model at a resolution of 0.4 mm^3 and the merged model at a resolution of 0.2 mm^3.

4.4 Octree Merger

Octree Merger (OM) v1.03 [23] is a tool that creates a surface using the projection operator of point set surfaces defined in [4, 18]. The input to this tool are the point sets defined by the vertices of the aligned range scans, with normals computed independently on each range map, and the output is the reconstructed surface. For *OM* the following parameter settings were determined:

- **-h #** Hole filling, determines where to stop creating the implicitly defined surface.
- **-e #** The max number of expansion steps.
- **-da #** The octree (visiting) depth.
- **-m #** The Mahalanobis scale
- **-i #** The max number of iterations in Brent's algorithm.
- **-n #** The size of the neighbourhood.

For the merging process we want to limit the creation of surface that fills regions without vertices. Since a hole filling value of zero results in no faces at all, we use the default value of one.
Selected setting: Hole filling is set to one, -h 1 (default)
For the maximum number of expansion steps of the surface, we tried 0, 1, and 2 steps. Figure 20 shows no difference between the values 1 and 2, but many tiny holes occur for value 0.
Selected setting: The max number of expansion steps is set to one, -e 1 (default)
The octree depth and the maximal visiting depth of this octree are both eight by default. In this experiment we tried depths of 7, 8, and 9. The use of an octree depth of seven shows undersampling of the surface, while an octree depth of nine show slightly oversampling of the surface (see Figure 21).
Selected setting: The octree visiting depth is set to eight, -da 8 (default)
Several values were tried for the Mahalanobis scale (1.0, 2.0, 3.0, 4.0, 8.0) and the maximum number of iterations (20, 50, and 100), but neither varying the Mahalanobis scale nor varying the maximum number of iterations showed a difference in their results.
Selected setting: The Mahalanobis scale is set to three, -m 3.0 (default)
Selected setting: The max number of iterations in Brent's algorithm is set to fifty, -i 50 (default)
For the size of the neighbourhood, the default value is fifteen. Values that were tried to improve the merge results include 10, 15, 20, and 30. The results of these settings are shown in Figure 22. A value of thirty is preferred, which results in a cleaner surface around holes than the other values.
Selected setting: The size of the neighbourhood is set to thirty, -n 30
The final results after the removal of the disconnected patches are shown in Figure 23.

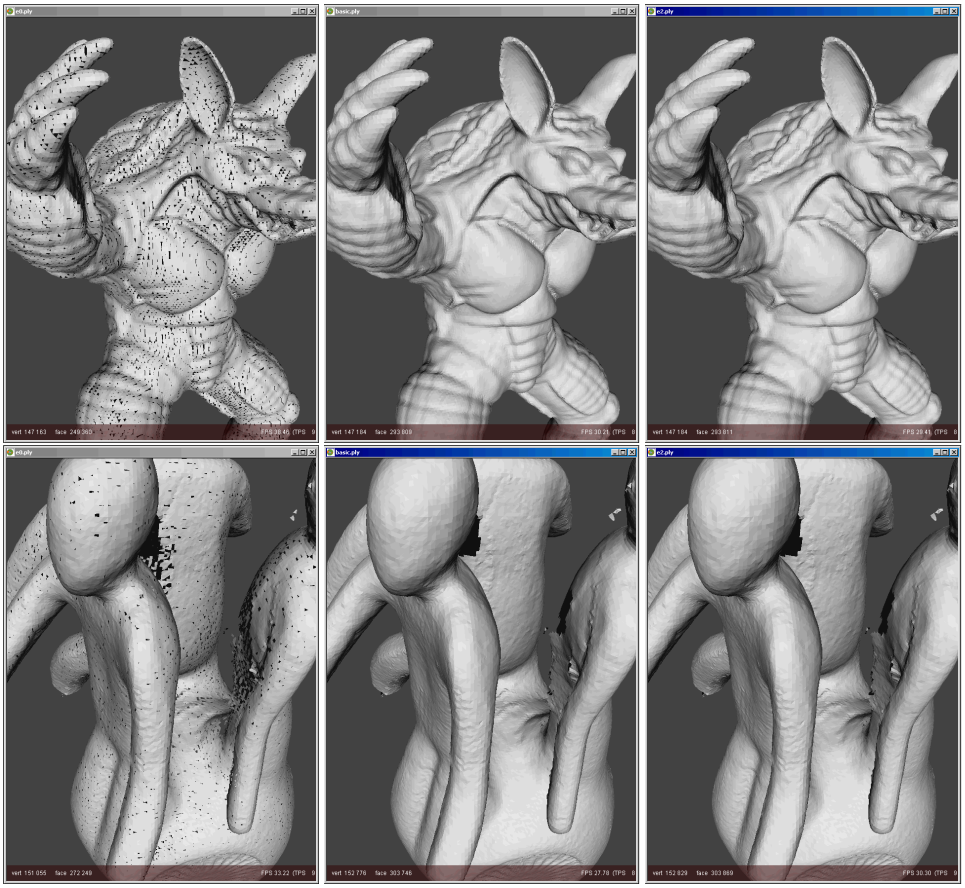


Figure 20: Results of OM while varying the maximum number of expansion steps. Left to right: 0, 1, and 2 steps. Using zero steps results in many tiny holes in the resulting surface. The use of one or two steps shows no significant difference, one step is selected.

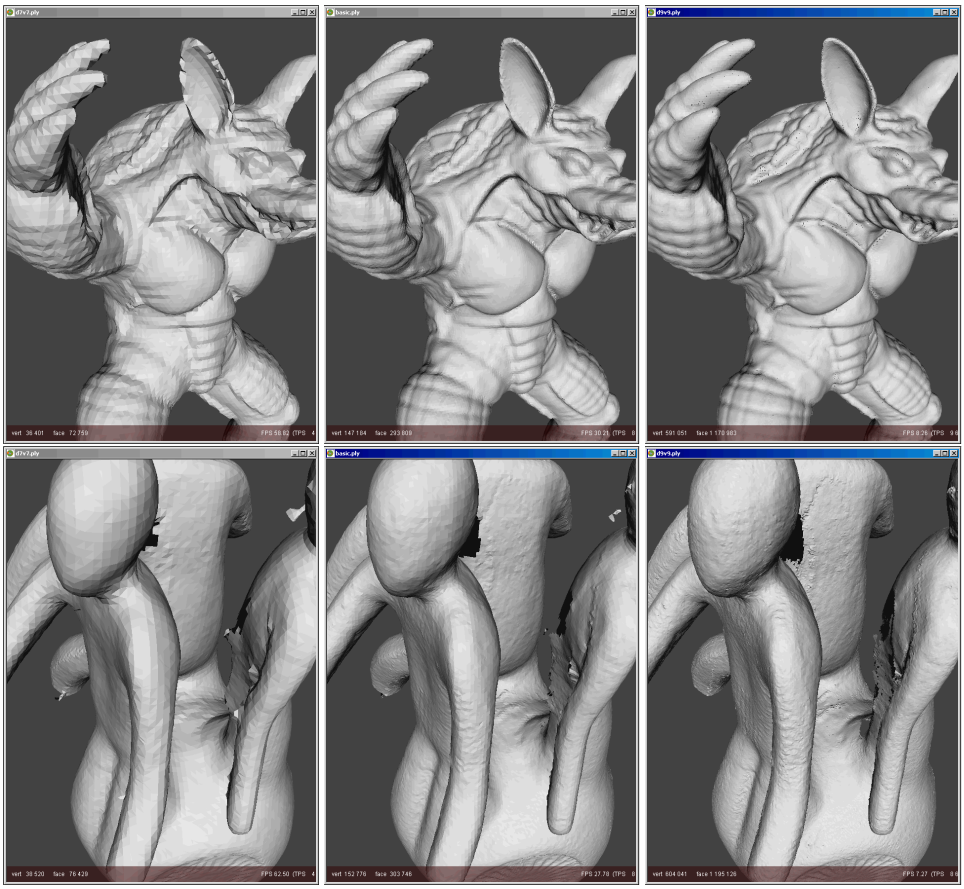


Figure 21: Results of OM while varying the octree depth. Left to right: An octree depth of 7, 8, and 9. A depth of seven shows a too coarse mesh, while a octree depth of nine shows oversampling of the mesh.

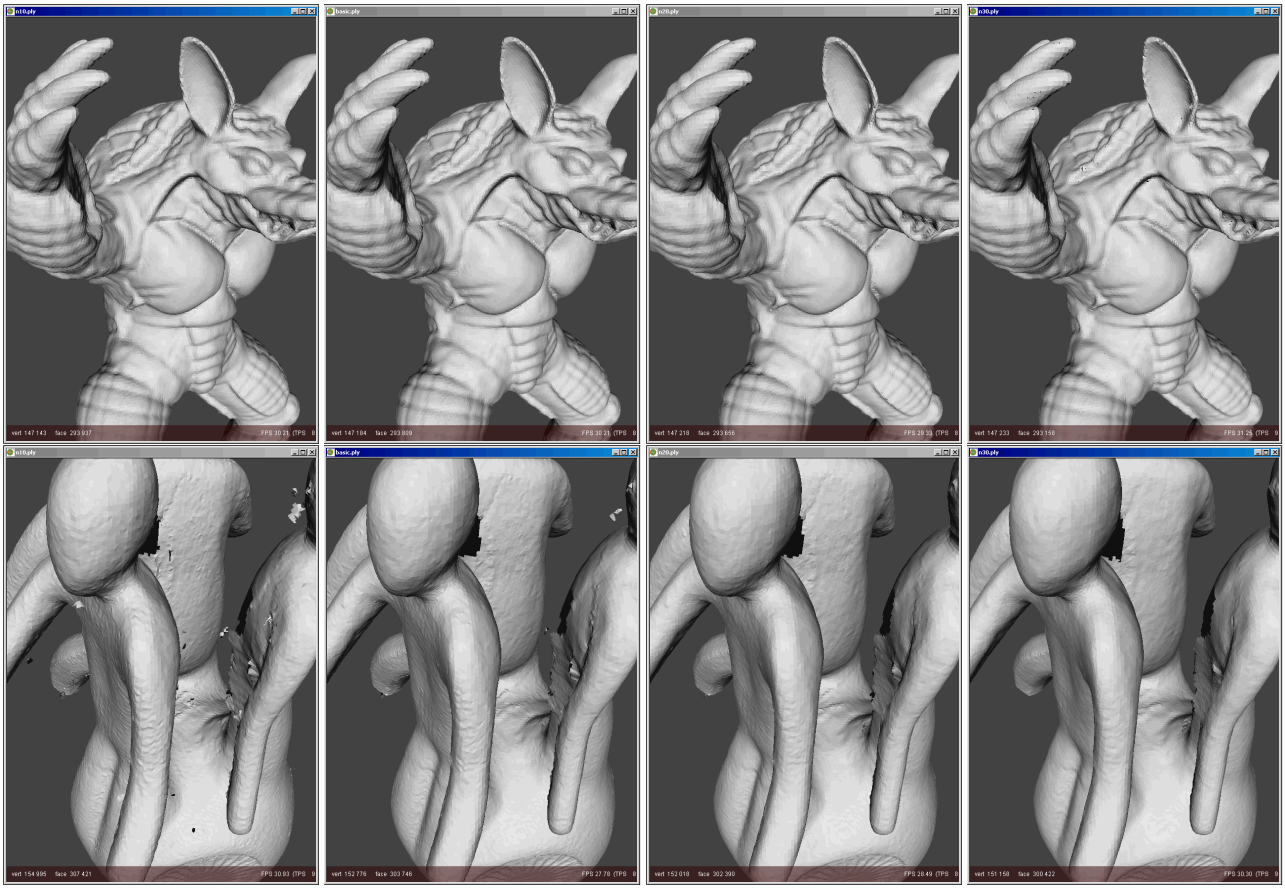


Figure 22: Results of OM while varying the neighbourhood size. Left to right: A neighbourhood size of 10, 15, 20, and 30, a value of 30 is preferred, because it resembles the intended surface best.

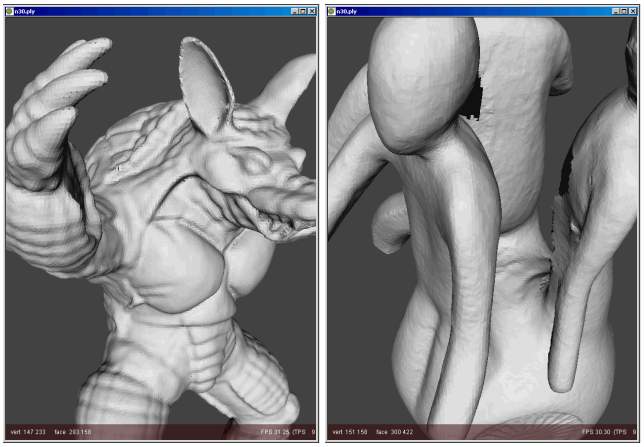


Figure 23: The final OM models after the removal of disconnected patches.

5 Experiment: Hole filling

It is hard and sometimes even impossible to observe the entire surface of an object with a laser range scanner. The result is that we miss range data in some areas. One cause of missing data might be the limited number of range scans from different viewing directions, another cause comes from the basic principle of laser range scanning itself. As described in Section 2.2, the acquisition of parts of the surface may fail due to absorption (by a dark coloured surface) and reflection (by a specular surface) of laser light, or due to occlusion (the sensor is not able to sense the laser dot). Even if it is physically possible to scan the entire surface of an object, it is most likely that holes appear in the final merged model.

In this experiment we attempt to fill the holes in the final merged models using several hole filling systems *MeshMerge*, *RapidForm*, *VripPack*, and *VolFill*. Some of these systems were designed to merge meshes (*MeshMerge* and *VripPack*), but additionally include a method to fill holes as well. The two latter systems (*VripPack* and *VolFill*) operate on the volumetric distance field grid of *VripPack* that was constructed during its merging of meshes, which is a limitation. For a fair comparison, we have to apply the systems *MeshMerge* and *RapidForm* on the merged model obtained using *VripPack*, and apply the hole filling systems *VolFill* and *VripPack* itself on the volumetric grid from which the merged model was extracted (Figure 24). In the experiment we fill the holes of the *UU-memento* model, which was reconstructed with VripPack using both the $0.4 \times 0.4 \times 0.4$ mm and $0.2 \times 0.2 \times 0.2$ mm resolution voxel grid. We used three particular views to investigate the performance of the hole filling techniques (Figure 25). Since the *armadillo* had only a few tiny holes, this model was not used in this experiment.

During the merging experiments we removed disconnected (noisy) patches from the final merged model. The removal of these patches before the hole filling process will give better results than the removal of these patches afterwards (Figure 26). The disadvantage of *VripPack* and *VolFill* is that they cannot benefit from the cleaning beforehand, since they operate on the volumetric grid instead of the extracted and cleaned surface. In the following sections the hole filling techniques are described in more detail.

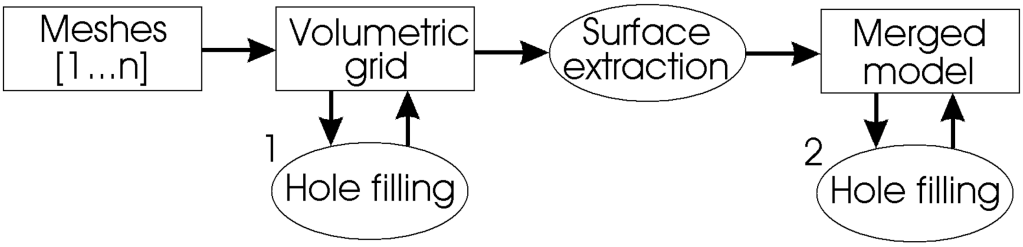


Figure 24: The hole filling process of both VripPack and VolFill (1) requires the volumetric distance field grid constructed by VripPack, while the other hole filling systems use the merged model extracted from this grid (2).

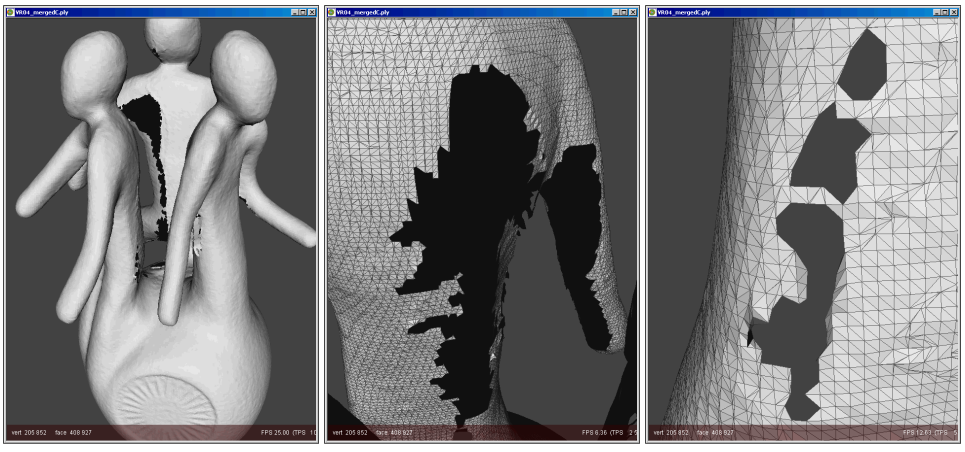
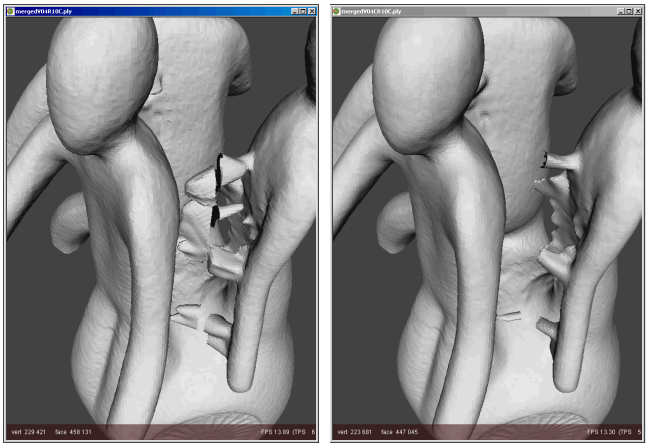


Figure 25: Three selected views to investigate the performance of the hole filling techniques: a global view (left), a huge and complicated hole (middle), and a few small and less complicated holes (right).



(a) Merging - Hole filling - Cleaning (b) Merging - Cleaning - Hole filling

Figure 26: The difference between cleaning before hole filling or after hole filling, the latter is preferred.

5.1 MeshMerge

As mentioned during the merging experiments *MeshMerge* is able to fill holes in a volumetric distance field using a number of refilling steps. When the merged model is used as input for *MeshMerge*, a volumetric distance field grid is constructed first. This distance field is expanded at the boundary areas according to a number of refilling steps. To get a good indication of this hole filling technique we applied ten refilling steps.
Selected setting: Ten refilling steps, -R10

5.2 RapidForm

RapidForm has the ability to fill holes in a surface based manner. This system automatically fills holes using either a flat based, smooth based, or curvature based technique. During the process a polygonal structure is constructed and both the hole and its surrounding region are remeshed. In case of curvature based hole filling the polygonal structure is curved to match the surrounding area.
Selected setting: Flat based filling
Selected setting: Smooth based filling
Selected setting: Curvature based filling

5.3 VripPack

The space carving operation in *VripPack* determines empty and unseen areas in its volumetric grid. During the hole filling process the distance field is expanded at the holes along the border of the unseen and empty areas until the holes are filled. Finally, the surface mesh is extracted from the volumetric grid. This method failed for the volumetric grid with resolution $0.2 \times 0.2 \times 0.2$ mm, due to the amount of memory required.
Selected setting: Space carving (-carve_conservative) followed by hole filling (-fill_holes)

5.4 VolFill

The *VolFill* (v1.0) system fills holes by expansion (volumetric diffusion) of *VripPack*'s volumetric distance field grid at the border of the unseen and empty areas. The volumetric diffusion in this system is the blurring of the volumetric distance field, which is performed a number of iterations. A hole is filled when the diffusion of the surface fills the empty area at the hole of the volumetric grid. After a number of blurring iterations the final surface mesh is extracted from the volumetric grid using the Marching Cubes algorithm. Six blurring iterations correspond approximately to ten refilling steps with *MeshMerge*.
Selected setting: Six blurring iterations, -n 6

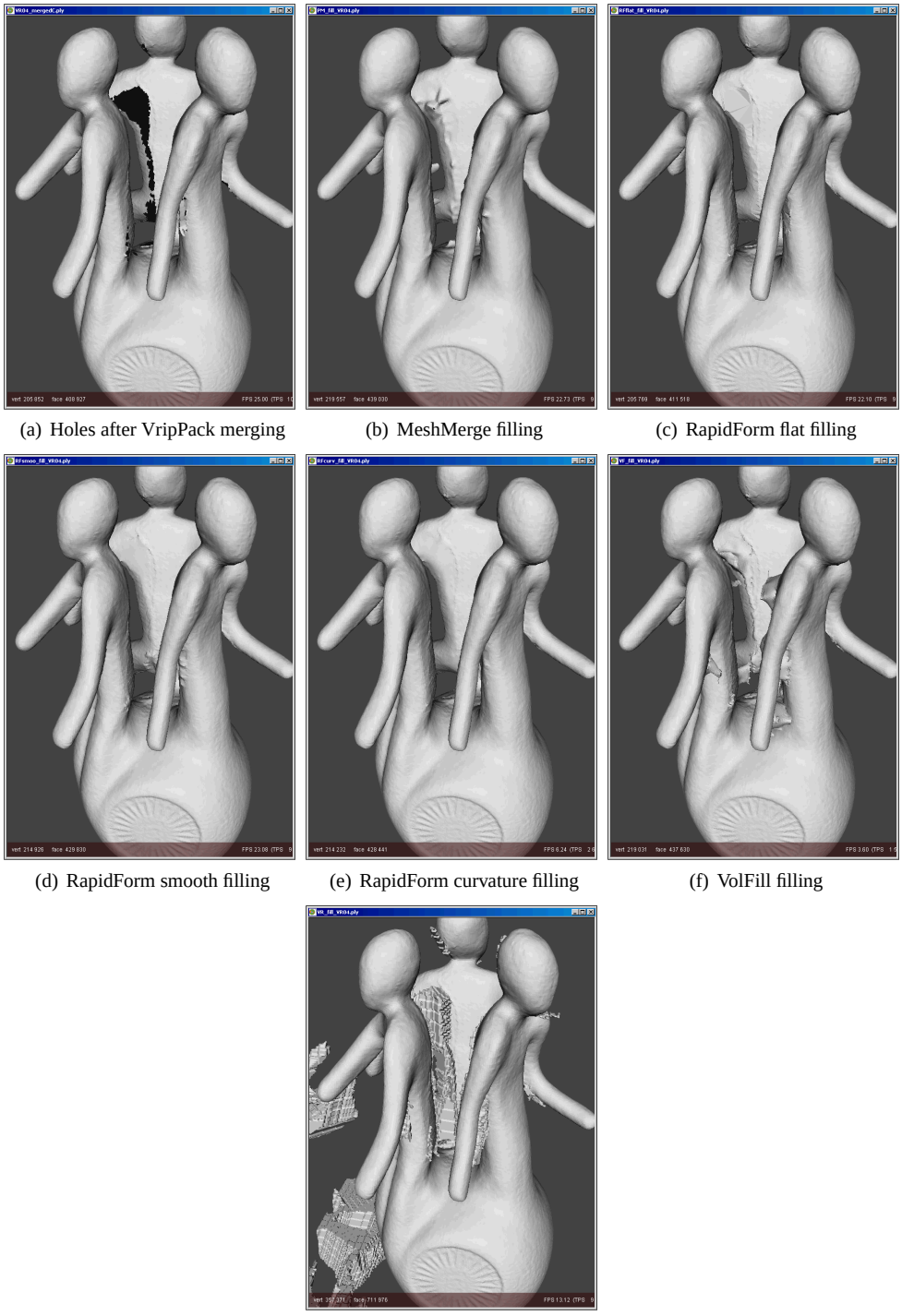
5.5 Results

In this experiment we have filled holes of the *UU-memento* models, which were reconstructed with VripPack using a high ($0.2 \times 0.2 \times 0.2$ mm) and a low resolution ($0.4 \times 0.4 \times 0.4$ mm) voxel grid. Results of the hole filling techniques were inspected for three particular views (Figure 25), and for both the high and low resolution grid. The Figures 27, 29, and 31 show the results of the hole filling systems with respect to the holes in the low resolution model. Figures 28, 30, and 32 show these results for the high resolution model. No results for the hole filling technique included in *VripPack* were obtained for the high resolution model, because it required more than than the available amount of memory (520MB RAM available).

Figure 27a shows the holes that remain after the merging process of *VripPack*. Note that these holes occur due to missing data during the acquisition stage, and that similar results were obtained using other merging systems. *MeshMerge* (27b) was able to fill the large hole, but shows a rather extruding surface. The same holds for *VolFill* (27f) with even more extrusions. The surface based hole filling techniques of *RapidForm* (27cde) were able to fill in the missing data almost perfectly, with a slightly better result for its curvature filling. *VripPack* (27g) on the other hand shows incorrect expansion of the volumetric grid into regions that were supposed to remain empty (see also [17]). For the holes in the high resolution model (29a), the results are very much the same. Only now, a smaller part of the holes was filled with the use of either *MeshMerge* or *VolFill*, because the same number of applied iterations generates a smaller amount of new (high resolution) surface.

The large and complicated hole in the low resolution model (Figure 29) shows again good results for the curvature filling (29e) and flat based filling (29c) using *RapidForm*. Reasonable results were obtained using *MeshMerge* (29b) and the smooth based variant of *RapidForm* (29d). The other systems expand the surface in an incorrect manner (29fg). For the high resolution model (Figure 30) almost none of the complicated holes were filled, only *RapidForm*'s curvature based filling was able to fill in a large hole.

The third set of inspected holes are much smaller and less complicated than the previous ones (see Figure 31 and 32). Almost all hole filling systems were able to fill in these holes well. If we look at the result of *MeshMerge* applied on the high resolution model (32b), we notice the creation of a tube-like surface. This system expands the surface according to the orientation of faces around the hole. A tube-like surface is generated because the surface around the hole has probably faces with opposing normals, causing the surface to expand perpendicular to the actual surface. This way the hole is 'lifted' instead of filled.

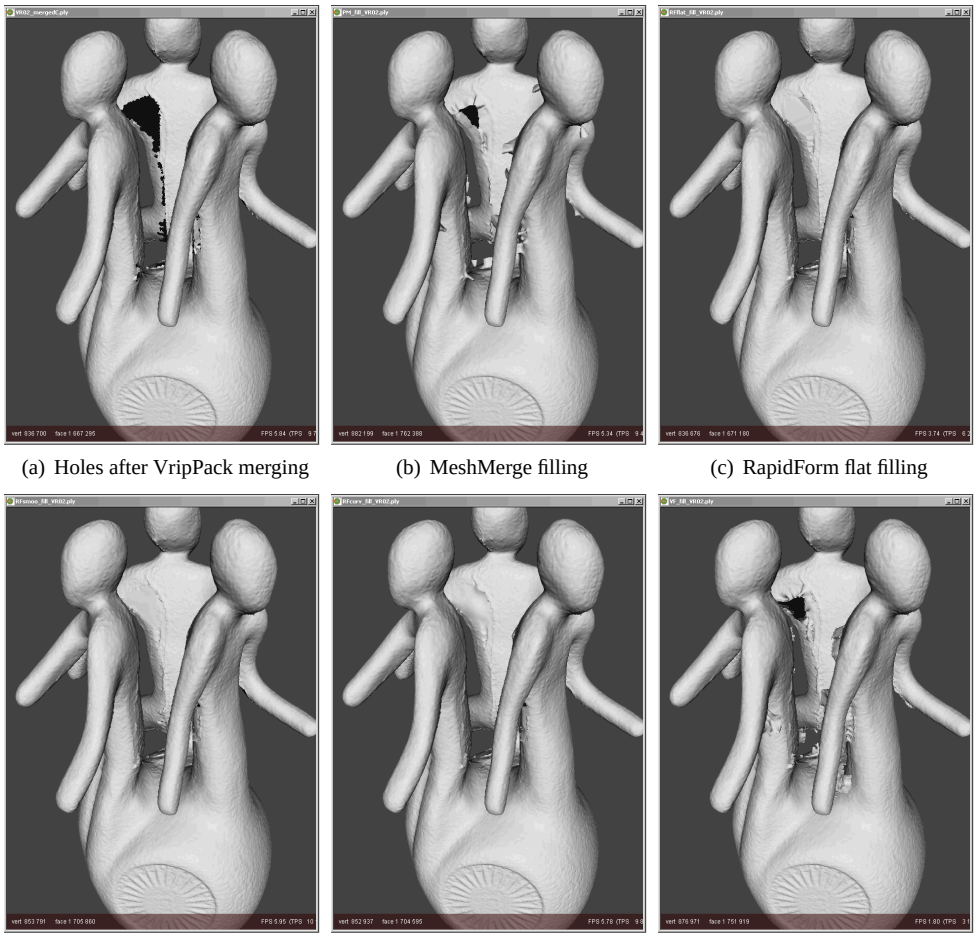


(a) Holes after VripPack merging (b) MeshMerge filling (c) RapidForm flat filling

(d) RapidForm smooth filling (e) RapidForm curvature filling (f) VolFill filling

(g) VripPack filling

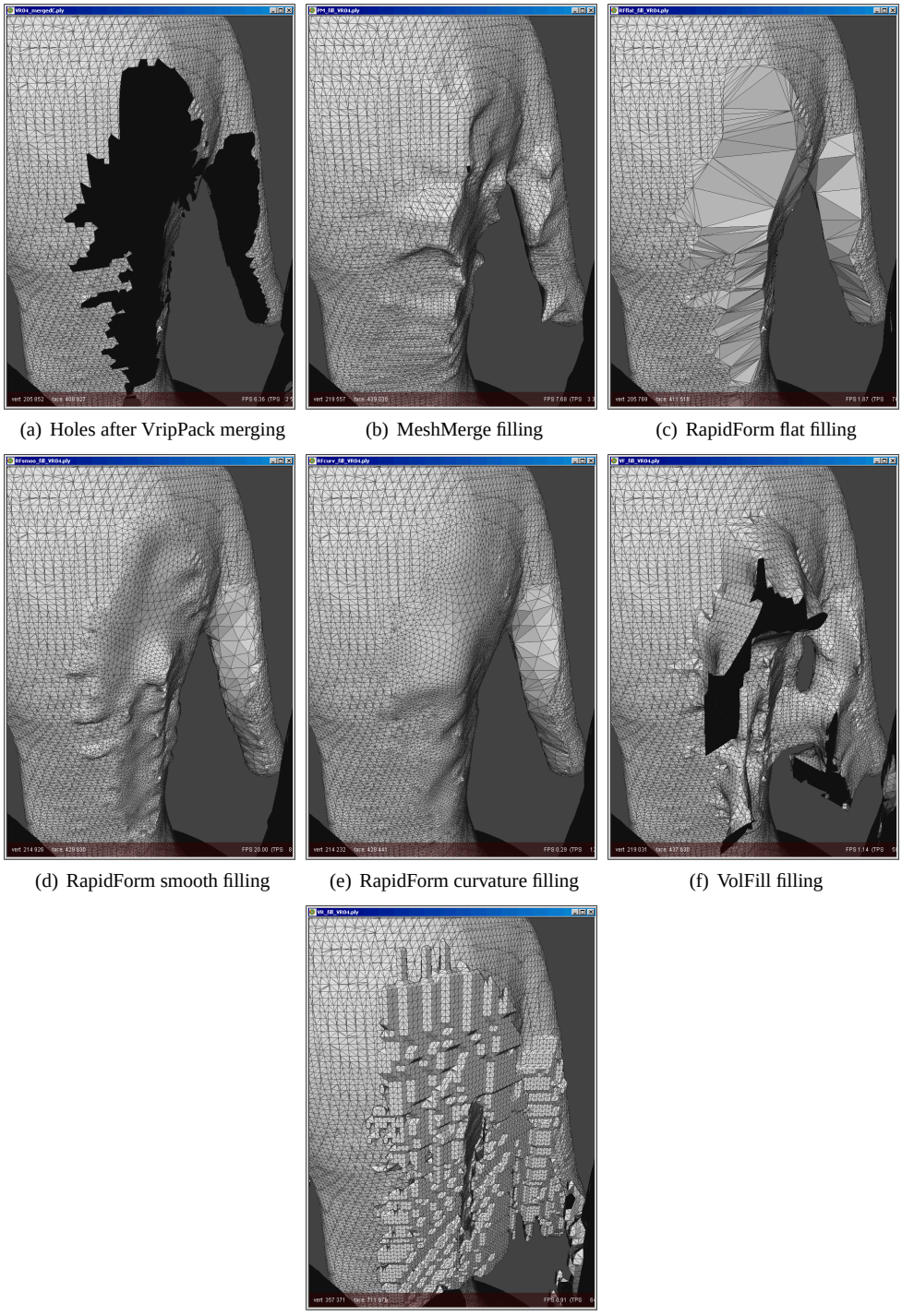
Figure 27: Results of the hole filling methods for the merged UU-memento. For the merging VripPack was used with a 0.4 mm^3 voxel grid.



(a) Holes after VripPack merging (b) MeshMerge filling (c) RapidForm flat filling

(d) RapidForm smooth filling (e) RapidForm curvature filling (f) VolFill filling

Figure 28: Results of the hole filling methods for the merged UU-memento. For the merging VripPack was used with a 0.2 mm^3 voxel grid.

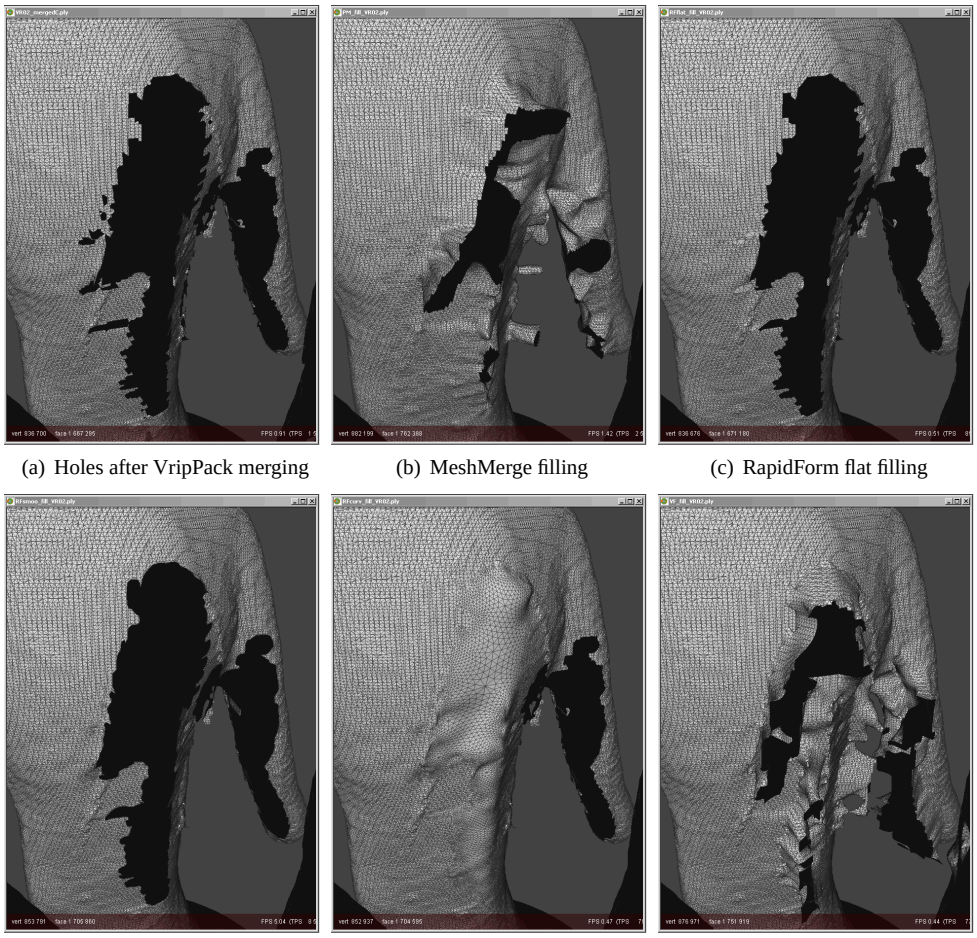


(a) Holes after VripPack merging (b) MeshMerge filling (c) RapidForm flat filling

(d) RapidForm smooth filling (e) RapidForm curvature filling (f) VolFill filling

(g) VripPack filling

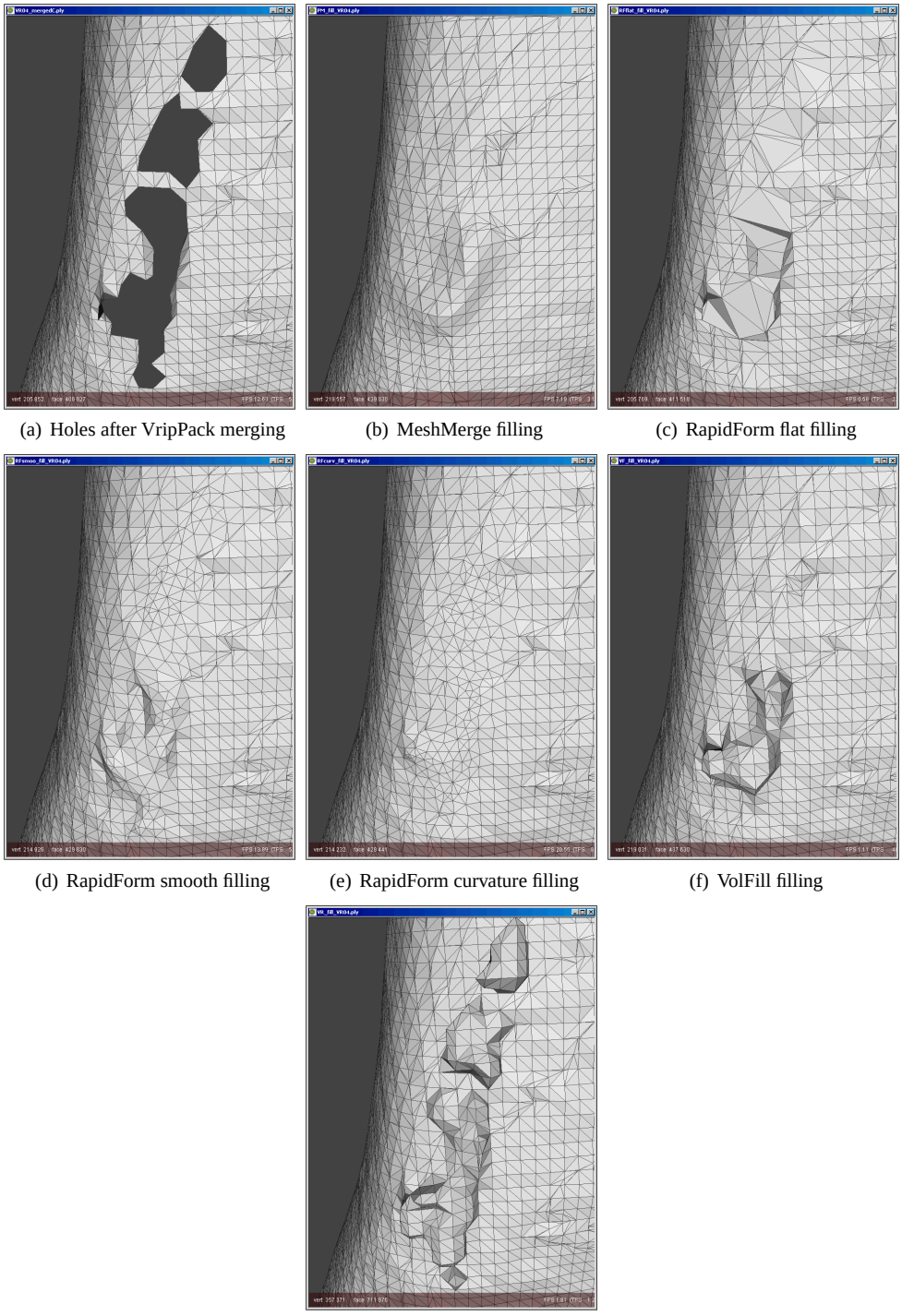
Figure 29: Results of the hole filling methods for the merged UU-memento. For the merging VripPack was used with a 0.4 mm^3 voxel grid.



(a) Holes after VripPack merging (b) MeshMerge filling (c) RapidForm flat filling

(d) RapidForm smooth filling (e) RapidForm curvature filling (f) VolFill filling

Figure 30: Results of the hole filling methods for the merged UU-memento. For the merging VripPack was used with a 0.2 mm^3 voxel grid.

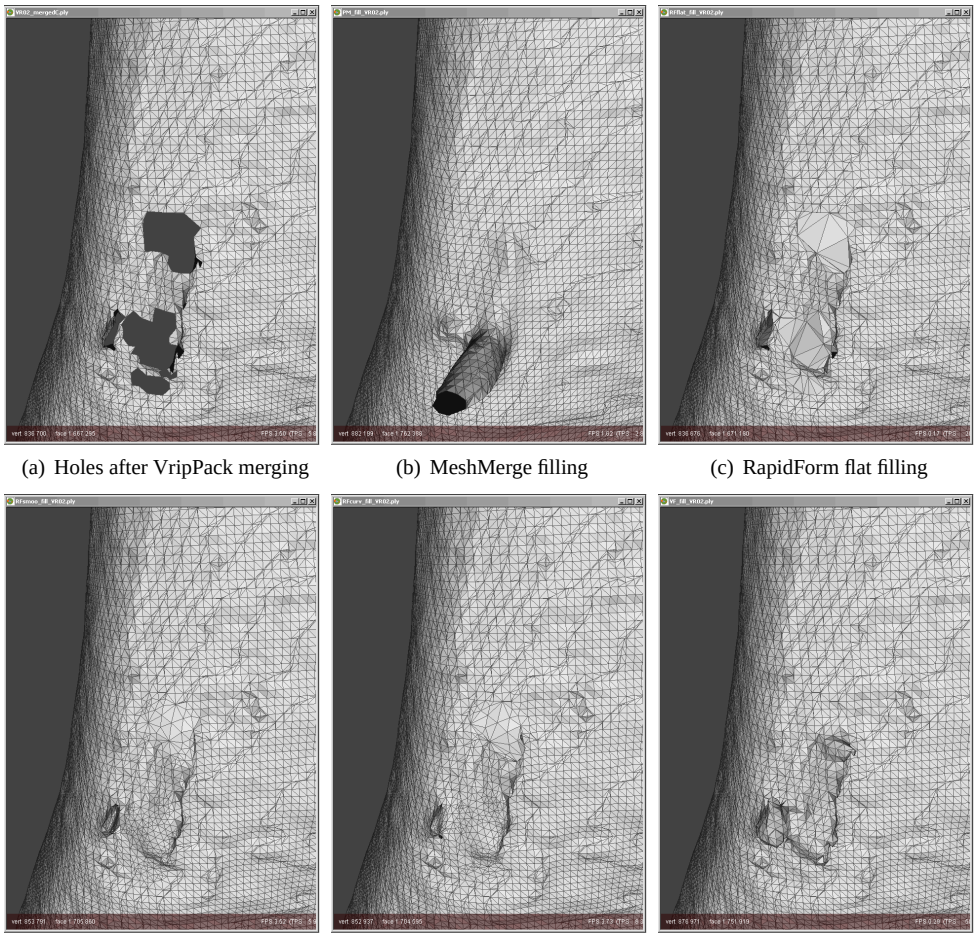


(a) Holes after VripPack merging (b) MeshMerge filling (c) RapidForm flat filling

(d) RapidForm smooth filling (e) RapidForm curvature filling (f) VolFill filling

(g) VripPack filling

Figure 31: Results of the hole filling methods for the merged UU-memento. For the merging VripPack was used with a 0.4 mm^3 voxel grid.



(a) Holes after VripPack merging (b) MeshMerge filling (c) RapidForm flat filling

(d) RapidForm smooth filling (e) RapidForm curvature filling (f) VolFill filling

Figure 32: Results of the hole filling methods for the merged UU-memento. For the merging VripPack was used with a 0.2 mm^3 voxel grid.

6 Discussion

In the experiments concerning the alignment of *armadillo* and *UU-memento* meshes, we used three systems for the fine alignment: *MeshAlign*, *RapidForm*, and *Scanalyze*. Since it was hard to finely align the manual coarse alignment at once in a robust and accurate manner, we performed the fine alignment twice using less restrictive settings during the first run. With the use of proper parameter settings during the second run each of the systems was able to perform an accurate alignment, which we verified visually by means of the splotchiness of the coloured surfaces.

For the merging of meshes we have used four different systems: *MeshMerge*, *RapidForm*, *VripPack*, and *OM*. This experiment tried to optimize the parameter settings for these systems. The final results after the removal of noisy disconnected patched, showed high resemblance to the actual objects for all four systems.

Systems that were applied on the holes of the *UU-memento* are *MeshMerge*, *RapidForm* (using three different techniques), *VolFill*, and *VripPack*. The two latter techniques had to be applied to the volumetric grid of *VripPack* instead of the final merged (and cleaned) surface. The hole filling experiment showed how difficult it is to fill different types of holes correctly. Small and not very complicated holes were easy to fill for almost all of the hole filling systems. But more complicated holes such as the backs of the *UU-memento* showed much more difference between the precision of systems. In general, the hole filling techniques of *RapidForm* using either surface curvature or simple flat triangles obtained the best results. However, these techniques failed to fill some large holes. In fact, none of the described systems filled all of the holes in the *UU-memento* models in a satisfactory manner. Since some of the inspected holes were very complicated, it is hard to say whether or not a hole filling system should be able to fill all of them properly. At least, the results showed some merits and demerits of the techniques used by hole filling systems.

In this work we have determined the parameter settings for alignment and merging systems. These settings are required to make a fair comparison between these systems, when we evaluate their performance. After such a comparison, we might be able to select an alignment system and a merging system that together obtain the most accurate 3D models out of sets of 3D range scans.

Future work will include the evaluation of the described hole filling systems, together with more recently developed hole filling systems, with the use of 3D human body models. Golden standards will be generated for some of the holes in these models, which will enable us to quantify the actual precision of each of the hole filling systems.

Acknowledgements

This research was supported by the FP6 IST Network of Excellence 506766 AIM@SHAPE. The *UU-memento* statue is copyrighted by Artihove Art Centre, Bergschenhoek (www.artihove.nl) and artist Maarten Benschop. We acknowledge Stanford for their software and the *armadillo* model, ISTI-CNR Visual Computing Laboratory, and RapidForm for the use of their software and V. Fiorin for providing *OM*.

References

[1] AIM@SHAPE Repository. FP6 IST Network of Excellence 506766. http://www.aimatshape.net/resources, April 2005.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, C. Silva, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Trans. on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[3] N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. *Computer Graphics*, 32(Annual Conference Series):415–421, 1998.

[4] N. Amenta and Y.-J. Kil. Defining point-set surfaces. In *ACM Trans. on Graphics (TOG)*, volume 23, pages 264–270, 2004.

[5] M. Attene and M. Spagnuolo. Automatic surface reconstruction from point sets in space. *Computer Graphics Forum*, 19(3):457–465, 2000.

[6] G. Barequet and M. Sharir. Filling gaps in the boundary of a polyhedron. *Comput. Aided Geom. Des.*, 12(2):207–229, 1995.

[7] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. Towards a general multiview registration technique. *IEEE Trans. PAMI*, 18(5):540–547, 1996.

[8] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[9] P. J. Besl and N. D. McKay. A method for registration of 3D shapes. *IEEE Trans. PAMI*, 14(2):239–256, 1992.

[10] P. Borodin, M. Novotni, and R. Klein. Progressive gap closing for mesh repairing. In *Advances in Modelling, Animation and Rendering*, pages 201–213, 2002.

[11] 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 *ACM SIGGRAPH 2001*, pages 67–76, 2001.

[12] C.-S. Chen, Y.-P. Hung, and J.-B. Cheng. RANSAC-based DARCES: a new approach to fast automatic registration of partially overlapping range images. *IEEE Trans. PAMI*, 21(11):1229–1234, 1999.

[13] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.
[14] J. Cheng and H. Don. A graph matching approach to 3-D point correspondences. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 5(3):399–412, 1991.
[15] C. S. Chua and R. Jarvis. Point signatures: A new representation for 3D object recognition. *Int. Journal of Computer Vision*, 25(1):63–85, 1997.
[16] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH 1996*, volume 30, pages 303–312, 1996.
[17] J. Davis, S. R. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *1st Int. Symposium on 3D Data Processing, Visualization, and Transmission*, 2002.
[18] V. Fiorin, P. Cignoni, F. Ganovelli, and R. Scopigno. A range scan merging tools based on point set surfaces. Technical Report TR-01-2006, ISTI-CNR, 2006.
[19] K. Higuchi, M. Hebert, and K. Ikeuchi. Building 3-D models from unregistered range images. *CVGIP-GMIP*, 57(4):315–333, 1995.
[20] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.
[21] D. Huber and M. Hebert. 3-D modeling using a statistical sensor model and stochastic search. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 858–865, 2003.
[22] INUS Technology. RapidForm 2004 PP2. http://www.rapidform.com, April 2005.
[23] ISTI-CNR Visual Computing Laboratory. MeshAlign v.2, MeshMerge and Octree Merger. http://vcg.isti.cnr.it/, April 2005.
[24] A. E. Johnson. *Spin Images: A Representation for 3-D Surface Matching*. PhD thesis, Carnegie Mellon University, Pittsburgh, 1997.
[25] T. Ju. Robust repair of polygonal models. *ACM Trans. on Graphics (TOG)*, 23(3):888–895, 2004.
[26] D. Levin. Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization*, pages 37–49, 2003.
[27] P. Liepa. Filling holes in meshes. In *Proc. of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP)*, volume 43, pages 200–205, 2003.
[28] W. E. Lorensen and H. Cline. Marching cubes: a high resolution 3D surface construction algorithm. In *Proc. SIGGRAPH 1987*, volume 21, pages 303–312, 1987.
[29] A. S. Mian, M. Bennamoun, and R. A. Owens. From unordered range images to 3D models: A fully automatic multiview correspondence algorithm. In *Theory and Practice of Computer Graphics (TPCG 2004)*, pages 162–166. IEEE Computer Society Press, 2004.
[30] P. J. Neugebauer. Reconstruction of real-world objects via simultaneous registration and robust combination of multiple range images. *Int. Journal of Shape Modeling*, 3(1-2):71–90, 1997.
[31] F. S. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics*, 9(2), 2003.
[32] J. Podolak and S. Rusinkiewicz. Atomic volumes for mesh completion. In *Symposium on Geometry Processing*, 2005.
[33] K. Pulli. Multiview registration for large data sets. In *Proc. 2nd Int. Conf. on 3D Digital Imaging and Modeling (3DIM)*, pages 160–168, 1999.
[34] C. Rocchini, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, and R. Scopigno. The marching intersections algorithm for merging range images. *The Visual Computer*, 20:149–164, 2004.
[35] Roland DGA Corporation. Roland LPX-250 Laser Range Scanner. http://www.rolanddga.com/products/3d/scanners/, November 2005.
[36] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. 3rd Int. Conf. on 3-D Digital Imaging and Modeling (3DIM)*, pages 145–152, 2001.
[37] A. Sharf, M. Alexa, and D. Cohen-Or. Context-based surface completion. *ACM Trans. on Graphics (TOG)*, 23(3):878–887, 2004.
[38] L. Silva, O. R. P. Bellon, and K. L. Boyer. Enhanced, robust genetic algorithms for multiview range image registration. In *Proc. 4th Int. Conf. on 3-D Digital Imaging and Modeling (3DIM)*, pages 268–276, 2003.
[39] Stanford Computer Graphics Laboratory. 3D Scanning Repository, Scanalyze and VripPack. http://graphics.stanford.edu/software/, April 2005.
[40] A. J. Stoddart and A. Hilton. Registration of multiple point sets. In *Proc. 13th Int. Conf. on Pattern Recognition*, volume A, pages 40–44, 1996.
[41] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proc. SIGGRAPH 1994*, pages 311–318, 1994.
[42] J. V. Wyngaerd, L. V. Gool, and M. Proesmans. Invariant-based registration of surface patches. In *IEEE Int. Conf. on Computer Vision*, pages 301–306, 1999.