Partial Surface integration based on Variational Implicit Functions and Surfaces for 3D model building

P. Claes*
ESAT/PSI/Medical Image Computing
Katholieke Universiteit Leuven

D. Vandermeulen
ESAT/PSI/Medical Image Computing
Katholieke Universiteit Leuven

L. Van Gool
ESAT/PSI/VISICS
Katholieke Universiteit Leuven

P. Suetens
ESAT/PSI/Medical Image Computing
Katholieke Universiteit Leuven

Abstract

Most three-dimensional acquisition systems generate several partial reconstructions that have to be registered and integrated for building a complete 3D model. In this paper, we propose a volumetric shape integration method, consisting of weighted signed distance functions represented as variational implicit functions (VIF) or surfaces (VIS). Texture integration is solved similarly by using three weighted color functions also based on VIFs. Using these continuous (not grid-based) representations solves current limitations of volumetric methods: no memory inefficient and resolution limiting grid representation is required. The built-in smoothing properties of the VIS representations also improve the robustness of the final integration against noise in the input data. Experimental results are performed on real-live, noiseless and noisy synthetic data of human faces in order to show the robustness and accuracy of the integration algorithm.

1. Introduction

Due to the limited field of view of 3D acquisition systems, three-dimensional models are often assembled from several partial reconstructions from different viewpoints. We will refer to the reconstruction from a single viewpoint as a patch. The number of views necessary is determined by the complexity of the object, the required detail, and the intrinsic resolution of the cameras involved. Combining several patches into a single model traditionally involves two main phases. First the patches need to be aligned into a common coordinate frame, which is called the registration phase. Secondly, the registered patches need to be integrated into a single entity, which is done in the integration phase. In this paper we concentrate on the integration phase. For the registration we refer to [1] for an overview of different registration tasks, problems and methods.

Surface integration methods differ by the type of input data used, unorganized or connected point sets, and the type of surface representation, parametric or implicit. Examples of methods integrating unorganized point sets are [3, 4], using parametric surfaces, and [5, 6], using implicit surface representations. Since these methods do not require a specially organized input set, they can be applied in more general situations. At the same time, however, these methods are less robust against noisy data, outliers and cannot reliably integrate high curvature regions. The integration can be improved using structured input data and parametric surfaces such as in [7, 8], but according to [2], they can still fail in areas of high curvature. The more successful approaches use structured data, while hiding the topological problems in the previous parametric surface based methods by using implicit volumetric representations. Johnson et al. [9, 10] create surface occupancy grids, which were the earliest and simplest volumetric representations. However, final surface extraction, based on ridge-detection in the surface likelihood, is not very robust [2]. In [2, 11, 12], volumetric integration algorithms are presented that construct a grid-based weighted signed distance function to the final surface. Triangular surface representations are extracted by Marching Cubes (MC) [13]. The methods differ in the way the implicit surface is constructed and the volumetric data organized. Signed distance functions are superior to occupancy grids since they can unify integration and registration into one step [1, 14, 15], while traditionally they were performed separately.

The amount of literature about texture integration or blending is much smaller than shape integration. In [9, 10] texture blending is done by weighted averaging of overlapping textures from the original contributing patches. Texture weights are a function of the angle between the consensus surface normal and the viewing direction or relative
pose of the patch. Pulli et al. [16] also perform a weighted texture blending, where the weights depend on the angle between the viewing direction of the contributing patch and the viewing direction of a virtual viewer, such that texture integration is view-dependent.

In this paper we propose a volumetric shape integration method consisting of weighted signed distance functions represented as variational implicit functions (VIF) or surfaces (VIS). Texture integration is solved similarly by using three weighted color functions also based on VIFs. Using these continuous (not grid-based) representations solves current limitations of volumetric methods: no memory inefficient and resolution limiting grid representation is required. The built-in smoothing properties of the VIS representations also improve the robustness of the final integration against noise in the input data.

The remainder of this paper is organized as follows. In section 2, we explain the principle of volumetric integration, based on [2], and give some drawbacks of current volumetric methods. In section 3, we introduce Volumetric Implicit Function (VIF) and Volumetric Implicit Surface (VIS) representations together with their most important properties, which are used to overcome the drawbacks of current volumetric integration methods. Section 4 explains the use of the VIF’s and VIS’s to solve the shape and texture integration problems. In section 5 we show experiments on real-time data, as well as noiseless and noisy simulation data. Finally a conclusion and discussion are given in section 6.

2. Volumetric Integration

Volumetric integration of different patches \( \{P_1, P_2, \ldots, P_j, \ldots, P_n\} \) into a model, as proposed by [2], uses a discrete, grid-based, implicit function \( D(x) \), representing a weighted average of signed distances to multiple patches. The resulting integrated surface is defined as the zero-valued iso-surface \( D(x) = 0 \).

\[
D(x) = \frac{\sum_{j=1}^{n} W_j(x) \times d_j(x)}{\sum_{j=1}^{n} W_j(x)}
\]

(1)

d_j(x) is the signed distance function of patch \( P_j \), where \( d_j(x) = 0 \) represents the patch surface. \( W_j(x) \) is a weight function, which locally gives a measure of confidence for the function \( d_j(x) \), where the confidence measures are derived from [17] for optical triangulation scanners. The functions \( d_j(x) \) and \( W_j(x) \) are constructed on a regular three-dimensional voxel-grid, by shooting rays through every voxel and the 3D sensor location of the patch acquisition.

Interesting properties of this volumetric integration, such as outlier robustness, incremental update, incorporation of range uncertainty, \( \ldots \), are listed in [2]. A disadvantage of the ray shooting algorithm, however, is that the signed distance functions are dependent on the sensor location. Other authors [11, 12, 15] resolve this problem by constructing the distance functions on the voxel-grid differently. It is also preferable to incorporate patch normals during build up of the distance functions in order to cope with sharp structures in the model. A major drawback of these volumetric integration methods is the use of a 3D voxel-grid, such that the final model resolution is limited by memory requirements and computational complexity. In [2] this problem is alleviated by run length encoding of each 2D slice in the grid, which depends on a complicated space carving procedure. In [12] grid octree structures are used, which is a hierarchical representation capable of increasing voxel resolution locally. In [16] the octree idea with hierarchical space carving is combined. Whether an octree or space carving is used, both methods introduce additional overhead, like voxel classification and octree building, in order to cope with the memory problem. Furthermore, features smaller than the voxel size will be missed.

It still remains unanswered to what extent signed distance build-up is sensitive to the presence of noisy data. Is the final signed distance value in each voxel reliable, when the input patch is very noisy (larger than the voxel size, \( \ldots \)) and does this affect the final integration? Especially patch normal information is severely influenced by noise. Building up the signed distance function with incorporation of these corrupted normals will lead to a wrong result.

Every surface integration method also has to cope with the presence of holes in some of the patches. A possibility is to perform hole filling after the integration and final mesh extraction, which is difficult when the holes are highly non-planar. In [2] hole filling is performed on the volume representation, but special voxel classification into near-surface, empty and unseen is necessary. In [18] holes in the distance field are filled by fitting local quadrics. No volumetric integration method uses a hole filling inherent representation.

Finally, disadvantages of current texture integration methods such as [9, 10] and [16] are the need for surface extraction before texture weight determination and the dependency of the weights on the relative poses of the patches, which makes the texture weights and blending sensitive to the accuracy of the patch registrations. Furthermore in [9, 10] additional vector information needs to be stored in every voxel of the grid, thereby increasing the memory requirements.

In the next section we will introduce a new volumetric representation for integration, that retains the desirable properties listed in [2], but at the same time resolves current limitations of volumetric integration.
3. Variational implicit functions and surfaces

A variational implicit surface (VIS) is an iso-surface of a 3D variational implicit function (VIF) created from constraint points using a variational scattered data interpolation approach as was first introduced in [19]. Alternatively, a VIS representation is the implicit modelling of a surface with a variational implicit function. These functions are obtained as special cases of a scattered data interpolation problem using variational techniques. We therefore first briefly introduce this particular technique. Given a collection of constraint points \( \{x_i| 1 \leq i \leq N\} \) with function constraints \( \{h_i| 1 \leq i \leq N\} \) construct a "smooth" scalar-valued function \( f(x) \) such that \( f(x_i) = h_i, 1 \leq i \leq N \). Smoothness is determined as a functional constraint points using a variational scattered data interpolation. These functions are approximated as a sum of weighted radially symmetric basis functions (RBF’s) centered at the constraint locations.

\[
f(x) = v(x) + \sum_{i=1}^{N} \lambda_i \phi(\|x - x_i\|)
\]

(2)

where:

- \( v(x) \) is a low degree polynomial, typically linear or quadratic.
- \( \phi \) is a real valued function (RBF) centered around a constraint point.
- the \( \lambda_i \)'s are the RBF coefficients or weights.
- the \( x_i \)'s are the RBF centers or constraint points.

Let \( V = \{v_1, \ldots, v_l\} \) be a monomial basis for polynomials of the degree of \( v(x) \), and \( c = \{c_1, \ldots, c_l\} \) be the coefficients that give \( v(x) \) in terms of this basis. The constraints (3) can then be written in matrix form as a linear system (4) for determining \( \Lambda \) and \( c \), and hence \( f(x) \).

\[
\begin{pmatrix}
A \\
V^t
\end{pmatrix}
\begin{pmatrix}
V \\
0
\end{pmatrix}
= \begin{pmatrix}
\Lambda \\
c
\end{pmatrix}
= \begin{pmatrix}
h \\
0
\end{pmatrix}
\]

(4)

where:

\[
A_{i,j} = \phi(\|x_j - x_i\|) \quad i, j = 1, \ldots, N
\]

\[
V_{i,j} = v_j(x_i) \quad i = 1, \ldots, N \quad j = 1, \ldots, l
\]

(5)

Solving (4), which is called RBF fitting, directly for more than a few thousand constraint points is computationally expensive and impractical. This is the main reason why the widespread adoption of RBF’s for data interpolation has been delayed. Recently, R.K. Beatson developed algorithmic advances involving hierarchical domain decomposition [20] and fast multi-pole methods [21] for the function fitting and evaluation to reduce the original computational complexity and the storage requirements in 2D and 3D. This makes RBF’s practical for 1 million and even more constraint points. The approximation algorithms of Beatson are implemented in the FastRBF library, which is commercially available at [22]. As a consequence of the approximation methods, the equalities of the constraints (3) are only approximated. Therefore two user-defined accuracies are needed. The first is the fitting accuracy and is defined as:

\[
\max_{i=1,\ldots,N} |f(x_i) - h_i|
\]

(6)

The second is the evaluation accuracy defined as:

\[
\max_{i=1,\ldots,N} |f(x_i) - a_i|
\]

(7)

Where \( a_i \) are the approximate values of the RBF’s at the points \( x_i \) for \( i = 1, \ldots, N \).

Given the machinery of variational techniques for scattered data interpolation, how can we use this to build VIS representations? The constraints (3), that we use to build a VIS representation based on a function \( f_j(x) \) for a patch \( P_j \), consist of boundary-constraints and normal-constraints. The boundary constraints are the 3D patch point locations through which the zero iso-surface of \( f_j(x) \) should pass \( (h_1 = 0) \) and the normal constraints are positive/negative valued constraints \( (h_1 = \pm 1) \) that are placed near the boundary constraints, and are positioned outside/inside the patch at a unit distance along the normal direction. Choosing these constraints leads to an implicit function \( f_j(x) \), which is a very good approximation (at least close to the surface, and gracefully degrading away from it) of a Euclidian signed distance-to-surface function. Representation of patch texture is done with three variational implicit functions. Every point \( p_{ji} \) of a patch \( P_j \) has an RGB-value \( (r_{ji}, g_{ji}, b_{ji}) \), from which three color variational implicit functions \( r_j(x), g_j(x) \) and \( b_j(x) \), can be constructed based on the constraint points \( \{p_{ji}| 1 \leq i \leq K_j\} \), where \( K_j \) are the number of points on \( P_j \), together with the following three function constraints:

\[
\begin{align*}
\{r_j(p_{ji}) &= r_{ji} | 1 \leq i \leq K_j\} \\
\{g_j(p_{ji}) &= g_{ji} | 1 \leq i \leq K_j\} \\
\{b_j(p_{ji}) &= b_{ji} | 1 \leq i \leq K_j\}
\end{align*}
\]

(8)

For the RBF’s \( \phi \) we use 3D biharmonic splines which are known as “smoothest” interpolators, in the sense that
they minimize certain energy functionals and interpolate the data. As a consequence interpolation or hole filling and extrapolation are inherent to the VIS representation. Small holes in the patches are interpolated and the function \( f_j(x) \) can be evaluated anywhere in 3D, on and off the surface, thanks to the extrapolation. Evaluating the function in a point off the surface will give an approximation to the signed distance to the surface in that point when using our constraints. Figure 1 shows the interpolation and extrapolation properties on an example patch. The three functions \( r_j(x) \), \( g_j(x) \) and \( b_j(x) \) can also be evaluated everywhere in 3D, also shown in figure 1 for \( r_j(x) \). The further away from the original patch, the smoother the color function is. The fitted 3D variational implicit function, from which the VIS can be extracted is given by Eq. 2. It is a continuous function that can be evaluated anywhere, not just at discrete grid points. Furthermore, gradients are continuous and smooth and can be determined analytically. The representation is topology independent, but in contrast to most other implicit function representations no memory inefficient voxel-grid is needed. The number of parameters to be stored is only dependent on the number of constraint or center points. The FastRBF library also offers a refined greedy algorithm for fitting RBF’s, which reduces the number of center points used to interpolate the data within a user-defined fitting accuracy [23]. Typically more constraint points are retained in regions with high curvature and less in flat regions. Data from range scanners are often noisy and therefore data smoothing is necessary. This can be easily done with variational implicit functions and surfaces either during evaluation or during fitting of the function (2)[24]. The first way is achieved by smoothing the function fitted to noisy data during evaluation, by means of low pass filtering. The controlling parameter is a cut-off frequency corresponding to the frequency of noise present in the data. In this way smoothing can be varied a posteriori. The second way is to smooth during the fitting process of the function to noisy data with a smoothest restricted range fitter, which assumes that there exists a range of values at each data point between which the true function value lies. During fitting the smoothest function which passes within the range of all the data points is found. The key parameter is an estimate of the noise magnitude at each data or constraint point. An alternative to the smoothest restricted range fitting is spline smoothing during fitting, which is a traditional RBF smoothing technique. Though it produces very good results, choosing the smoothing parameter can sometimes be difficult since it does not have a straightforward physical analogy as the range parameter in the smoothest restricted range fitting or the cut-off frequency in the low-pass filtering approach have.

The next section shows the use of VIF’s and VIS’s for volumetric integration.

4. Shape and texture integration

Shape integration is performed by replacing \( d_j(x) \) in equation (1) by the VIF \( f_j(x) \) for every patch \( P_j \).

\[
D(x) = \frac{\sum_{j=1}^{n} W_j(x) * f_j(x)}{\sum_{j=1}^{n} W_j(x)}
\]

The weight functions \( W_j(x) \) are derived from variational implicit weight functions based on range uncertainty or point confidences similar to [2]. First we determine the border \( \partial P_j \) of every patch \( P_j \) and the distance \( d(p_{ji}, \partial P_j) \) of every other point \( p_{ji} \) in \( P_j \) to that border for \( i = 1, \ldots, K_j \). For every 3D point \( p_{ji} \) of a patch \( P_j \) a corresponding weight \( w_{ji} \) (point confidence) is calculated according to:

\[
c_{1ji} = \log(1 + \frac{\max\{d(p_{ji}, \partial P_j)\}}{\log\{100\}})
\]

\[
c_{2ji} = \frac{\arcsin(n_{ji}, \hat{n}_j)}{\pi/2}
\]

\[
w_{ji} = c_{1ji} * c_{2ji}
\]

\( c_{1ji} \) gives lower weights to points near a patch border, because borders of patches are often less reliable. \( c_{2ji} \) takes into account the angle between the normal \( n_{ji} \) in a point of a patch \( P_j \) and the sensor direction \( \hat{n}_j \) from which the patch was captured. Points at grazing angles get lower weights. \( w_{ji} \) combines \( c_{1ji} \) and \( c_{2ji} \). Figure 2 shows \( c_{1ji} \) and the combined \( w_{ji} \) on a patch. Based on the points of the patch, being the constraint points, and there weights \( w_{ji} \), being the function constraints, a new variational implicit function \( w_{ji}(x) \) can be constructed and together with

![Figure 1](image-url)
the function \( f_j(x) \) a new weight function \( W_j(x) \) per patch can be defined.

\[
W_j(x) = \begin{cases} 
0 & \text{if } f_j(x) > t_d \text{ or } w_j(x) < t_w \\
|w_j(x)| & \text{if } f_j(x) \leq t_d \text{ and } w_j(x) \geq t_w
\end{cases}
\]  

(11)

The threshold \( t_d \) is to eliminate the influence of a patch to points in front or behind the patch which are further away than \( t_d \), so that thin objects can be reconstructed. Also, two patches at opposite sides of an object must not influence each other. The threshold \( t_w \) limits the influence of a patch outside his border. Hole filling consists in a good determination of this threshold or by not taking into account borders or small holes in the patches for the determination of distances to the closed border for every patch point.

Texture integration is performed in the same way as shape integration, based on three weighted sums of color VIF’s \( r_j(x), g_j(x) \) and \( b_j(x) \) of every patch.

\[
R(x) = \sum_{j=1}^{n} \frac{W_j(x) \cdot r_j(x)}{\sum_{j=1}^{n} W_j(x)} \\
G(x) = \sum_{j=1}^{n} \frac{W_j(x) \cdot g_j(x)}{\sum_{j=1}^{n} W_j(x)} \\
B(x) = \sum_{j=1}^{n} \frac{W_j(x) \cdot b_j(x)}{\sum_{j=1}^{n} W_j(x)}
\]

(12)

The weight functions \( W_j \) are the same as in equation (9), such that the texture weights for blending depend on the point confidences in the individual patches.

Final implicit surface polygonisation is done by Marching Triangles (MT) [25], which is a new surface based approach to implicit surface extraction. The most interesting property of MT compared to Marching Cubes (MC) is that no 3D voxel-grid is needed. There are only three requirements needed to use MT. Firstly, a seed triangle is needed to initialize the growing. One or more triangles of the original patches can be used and projected on the zero iso-surface of \( D(x) \). Secondly, evaluation of the implicit function must be possible for every point in 3D. This is done with Eq. (9). Finally, it must be possible to determine the derivative of the implicit function in a 3D point. To calculate the derivative of \( D(x) \) in an arbitrary 3D point \( x \) the patches that have an influence on the point are to be determined. Then, the derivative of \( D(x) \) from those patches is calculated according to:

\[
S = \{ j \mid d_j(x) \leq t_d \text{ and } w_j(x) \geq t_w, \text{ for } j = 1, \ldots, n \}
\]

\[
\frac{\partial D(x)}{\partial x} = \sum_{j \in S} \frac{\partial w_j(x)}{\partial x} f_j(x) + \sum_{j \in S} \frac{\partial w_j(x)}{\partial x} b_j(x) \left( \sum_{j \in S} w_j(x) \right)^2 \ldots
\]

(13)

Texturing of the final integrated model mesh, obtained by MT, is done by evaluating every mesh vertex into \( R(x), G(x) \) and \( B(x) \) of equation (12), resulting in an RGB triplet for every vertex.

5. Experiments

This section shows some experimental results on real-time data, noiseless - and noisy simulated data to analyze the performance obtained by our integration algorithm. The data was acquired using the ShapeCam hardware and ShapeWare software technology [26], where a grid is projected on the object to reconstruct. Calibration of the system is obtained using a calibration box. The objects we reconstruct are human faces, built as part of a project on craniofacial reconstruction. Human faces are fairly complicated objects to reconstruct. The nose typically causes a lot of problems, because of the amount of curvature and it always being partially occluded in different patches. Hair and eyebrows introduce severe noise and local surface misinterpretation. A last difficulty is the non-rigidity, like swallowing, of the face during acquisition of different patches, which introduces imperfect surface matches between patches.

Table 1 shows a comparison of memory occupation in Matlab R13 (double accuracy) between a uniform grid representation with no memory optimization and a VIS representation of a face. In practise typical grid optimizations described in section 2 are need to be taken into account.

The real-live data set is used to show the robustness of the algorithm against bad data and consists of 11 patches. Notice the very bad nose reconstruction in the closeup images, together with holes typically located at the nose and the eyebrows in figure 3 (a-c). Normally the individual patches

\[\begin{array}{|c|c|c|c|}
\hline
\text{input triangles} & \text{X dim.} & \text{Y dim.} & \text{Z dim.} \\
\hline
54850 & 211 \text{ mm} & 298 \text{ mm} & 260 \text{ mm} \\
\hline
\end{array}\]

Table 1. Comparison of memory occupation between a grid and a VIS representation

\[\begin{array}{|c|c|c|c|}
\hline
\text{GRID representation} & \text{VIS representation} \\
\hline
\text{Voxel size} & 1 \text{ mm} & \text{accuracy} & 1e-04 \text{ mm} \\
\hline
\text{Dimensions} & 251x259x182 & \text{constraints} & 55790 \\
\hline
\text{memory} & 78.25 \text{ Mb} & \text{memory} & 1.85 \text{ Mb} \\
\hline
\end{array}\]
are first cleaned up manually with the ShapeWare software, which involves smoothing, hole filling and deletion of bad parts. We choose not to perform the manual cleanup. Before integration, the patches were registered with a registration algorithm that we developed in [1]. The results of the registration and integration are depicted in figure 3 (d-f).

When observing the final integrated model in figure 3, we see that the nose is not bad, but it is not yet perfect. This can be resolved by deleting some really bad nose parts, like in the closeup images (b,c). However this involves cumbersome manual intervention. Another option is to increase the number of patches so that more consistent measurements of the same data is incorporated. A third option is to define a weight function similar to $c_{1ij}$ and $c_{2ij}$ in (10), that detects bad parts like the nose close-up in figure 3(b,c) and gives them a lower weight during integration.

The second data set is a noiseless simulated or synthetic data set in order to show the accuracy of the integration algorithm in combination with a registration algorithm [1]. Starting from a complete 3D model of a face obtained by using the ShapeWare software, after some manual editing, 10 partial overlapping patches were cut out of the 3D model. Figure 4 shows the original 3D face model (a,b) and an example patch (c). Then the patches except one were randomly translated and rotated in 3D space. The registration in [1] realigned the patches into the coordinate system of the unchanged patch. By doing so we can compare the registration and the integration with the ground truth, being the starting model. The resulting accuracies can then be calculated as the mean and the standard deviation in $mm$ of the absolute function evaluations of patch and integrated mesh points into the variational implicit function of the complete 3D model.

During integration the influence limitation thresholds $t_d$ and $t_w$ were set to 15 $mm$ and 0 respectively. Figure 5 shows the visual results of the integration, while table 2 reports the errors or accuracies in $mm$ and additional information, like memory needed for the VIS representation and the weight and texture VIF’s of all patches in Matlab$^TM$ R13.

The last data set is a noisy synthetic data set constructed very similarly to the previous data set, except that we added noise to the patch point coordinates as shown in figure 4 (d). The purpose of this data set is to show the robust and accurate performance of the integration algorithm in combination with a registration in the presence of noise, by making use of the smoothing properties described in section 3. The noise followed a normal distribution with zero mean and 0.5$mm$ standard deviation, so that the maximum difference between a noiseless and a noisy patch point is 0.866$mm$.

The simulated noise we added does not necessarily occur in reality, but when knowing the noise distribution of the acquisition system or the noise per data point, one of the
smoothing techniques in section 3 can be used accordingly, which is well shown in [24]. As a consequence, the main strategy and robustness of the algorithm against noise, does not change depending on the type of noise.

To create smooth functions \( f_j(x) \) from the noisy data we choose to use the spline smoothing technique during fitting, with a smoothing parameter \( \rho = 32 \). The accuracy results together with the mean and standard deviation of the absolute noise are reported in Table 2, while figure 6 shows the results visually.

6. Conclusion

In this paper we proposed a volumetric integration algorithm based on structured input data. By making use of variational implicit surface and function representations of the patch surfaces, textures and weights we overcome some limitations of current volumetric integration methods. No memory inefficient 3D voxel-grid is used, such that the final resolution or accuracy is not limited by memory requirements. Hole filling does not require an extra step because of the interpolation and extrapolation properties of the chosen representations. Furthermore, the representation has useful smoothing properties when dealing with noisy input data. Texture integration was performed similarly to shape integration, making use of the same weight functions in order to reduce the texture influence of low confidence patch points. Experiments on real-live data, noiseless and noisy synthetic data show that the integration is accurate and robust against noisy data.

The drawback of our integration algorithm is computational complexity. This is because during point evaluation of a variational implicit function every RBF has to be evaluated in that point. Increasing the number of constraint points increases the number of RBF’s, resulting in a slower point evaluation of the function. The FastRBF library already improves the speed by reduction of constraint points during fitting, domain decomposition and multi-pole methods, but still the speed performance is lower than point evaluation in a 3D voxel-grid. Improving the speed of variational implicit function evaluations is an active research topic. When in the future faster methods are available, these can be easily plugged-in into our algorithm. Furthermore, other and better confidence definitions for patch points to cope with

### Table 2. Data information and accuracy results in mm for noiseless and noisy data

<table>
<thead>
<tr>
<th></th>
<th>noiseless</th>
<th>noisy</th>
</tr>
</thead>
<tbody>
<tr>
<td>patches</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>input triangles</td>
<td>582344</td>
<td>582344</td>
</tr>
<tr>
<td>output triangles</td>
<td>93652</td>
<td>94740</td>
</tr>
<tr>
<td>total memory</td>
<td>54 Mb</td>
<td>47 Mb</td>
</tr>
<tr>
<td>X dim. mm</td>
<td>160</td>
<td>160</td>
</tr>
<tr>
<td>Y dim. mm</td>
<td>193</td>
<td>193</td>
</tr>
<tr>
<td>Z dim. mm</td>
<td>142</td>
<td>142</td>
</tr>
<tr>
<td>noise ( \mu )</td>
<td>0</td>
<td>0.4105</td>
</tr>
<tr>
<td>noise ( \sigma )</td>
<td>0</td>
<td>0.3107</td>
</tr>
<tr>
<td>registration ( \mu )</td>
<td>0.0137</td>
<td>0.4159</td>
</tr>
<tr>
<td>registration ( \sigma )</td>
<td>0.0163</td>
<td>0.3148</td>
</tr>
<tr>
<td>integration ( \mu )</td>
<td>0.0223</td>
<td>0.0864</td>
</tr>
<tr>
<td>integration ( \sigma )</td>
<td>0.0361</td>
<td>0.0961</td>
</tr>
</tbody>
</table>
bad data are to be explored.

Acknowledgements

This work is supported by the Flemish Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT, project IWT/GBOU/020195), by the K.U.Leuven (project IOF/GOA/2004/05) and by the Fund for Scientific Research - Flanders (FWO-Vlaanderen, project FWO/G.0258.02). We also acknowledge T.Brouwers and W.Coenen, who implemented parts of this work as part of their MSc in Electrical Engineering thesis.

References


[18] Takashi Masuda. "Filling the Signed Distance Field by Fitting Local Quadrics.” In proceedings of the 2nd international symposium on 3D Data Processing, Visualization, and Transmission (3DPVT’04): 2004


[25] www.eyetronics.com