Optoinspect3D Inline UI

2.1

User interface for testing the Optoinspect3D Inline software library

Contact information: **Fraunhofer Institute for Factory Operation and Automation IFF**
Business unit of Measurement and Testing Technology (MPT)
Dr. Christian Teutsch

Phone: +49 391 4090-239
Email: christian.teutsch@iff.fraunhofer.de

Fraunhofer IFF

Content

1 Introduction

Non-contact optical methods of 3D digitization have replaced conventional methods in many applications. Regardless of whether the triangulation sensors operate with points, lines or planes - many millions of measured 3D data are generated in a few seconds. These data deliver a description of a digitized component's surface topography. After digitization, verification of dimensional accuracy for quality assurance tasks requires methods and algorithms that evaluate and analyze the 3D data in order, for instance, to ascertain deviations of dimensions, shape and position from a standard model.

Established programs for point cloud processing that determine the desired geometric parameters manually and interactively are available for offline analysis. Systems integrated in production require rapid, automatic and integrated data evaluation, however. The **Optoinspect3D Inline** package for measured 3D data processing was developed for this purpose. The **Optoinspect3D Inline** package contains:

- **Optoinspect3D Inline function library:**
  - Rapid algorithms optimized by multicore support
  - Efficient data structures
  - C-Interface, which can be easily wrapped to a C++-interface
- **Optoinspect3D Inline UI** graphical demo application:
  - Interactive graphical tool for development, testing and parameterization of functions based on proprietary measured data
  - Custom selection modes to apply individual algorithms to a predefined subarea of the total volume of data

The library provides a generic C-Interface for an easy integration into a wide variety of applications and systems. C++ wrappers can be easily written since the style of C-interface is based on the standard STL-algorithms with STL vectors. A simple Ready-To-Run test application is provided with this package.

System Requirements for the library

- Microsoft Windows 7 SP1 and higher (64-Bit)
- Installation of the supplied vc_redist.exe (64-Bit)

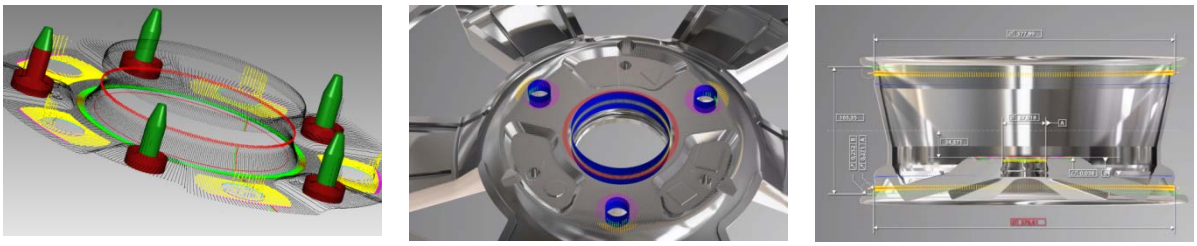


Figure 1: metrological analysis of 3D point clouds derived from optical 3d scanning devices.

2 Installation

Before using this software please **install the Microsoft Visual Studio Runtimes** which are supplied with „vc_redist.exe“. Administrator rights may be required.

In order to start the program the file „ Optoinspect3D Inline_x64.exe“ can be executed directly without having any special user rights.

2.1 Requirements

- 64-Bit Windows 7 or higher
- Graphics card that supports OpenGL 3.3.

3 Graphical User Interface

The graphical user interface provides two main areas for the interaction. A large OpenGL based area for the visualization and an area at the right-hand side for the object tree and for the parameterization of the algorithms.

Algorithms can be selected under the main menu entry „Algorithms“.

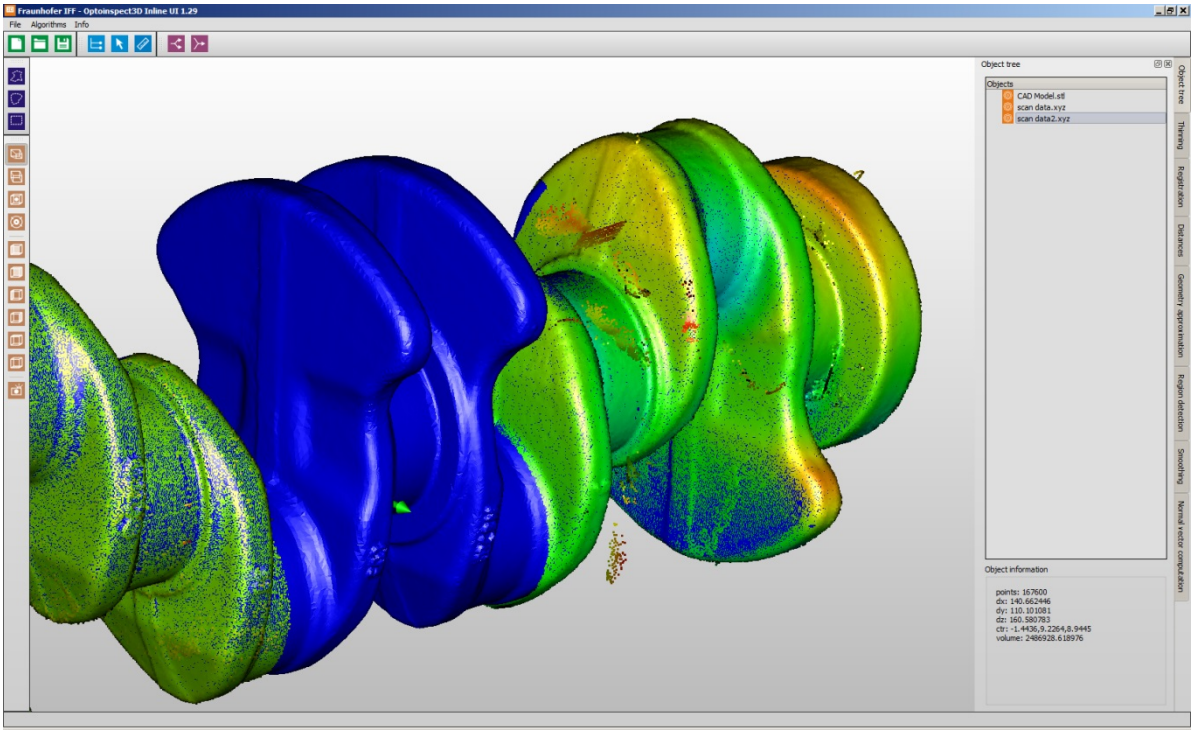


Figure 2: graphical user interface overview.

3.1 Loading and Saving point clouds

Clears the scene and removes all objects.

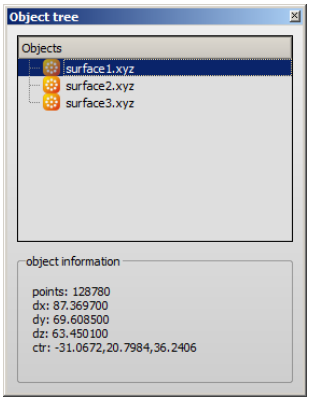
Opens a file selection dialog. Supported file formats xyz (ASCII), xyzc (ASCII) and scn (Fraunhofer IFF). Drag&Drop for files is also supported.

Opens a file selection dialog.
All **selected** point clouds are saved to one single file (XYZ/XYZC).

3.2 Interaction

Rotate	Press **left mouse button** to rotate the scene camera The rotation direction (horizontal/vertical) is fixed when ALT-Key is pressed simultaneously.
Zoom	Use **mouse wheel** or Shift+left mouse button to zoom.
Move	Press **CTRL + left mouse button** to move the scene camera.

3.3 Data management



Object tree

- Shows the current objects (point clouds and geometry elements)
- Additional object specific options are available by pressing the right mouse button. This includes display/hide object, remove object and select object color.

Object specific information

- Shows detailed object specific information of the currently selected object.

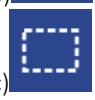
Object selection with the mouse

Alternatively, objects may be picked and selected with the selection tool. The arrow button enables the object selection mode. By clicking on an object with the mouse the object gets selected.

3.4 Selection


(a)


(b)


(c)

Select points

A polygon can be drawn to select a region of points **(a)**. Use the **left** mouse button to add points/edges to the polygon. Selection ends by clicking near the first point or by pressing the **right** mouse button. Additionally a free-hand selection **(b)** and a rectangular selection **(c)** are provided. All selected points were colored in red.

De-select points

Selected points can be de-selected by pressing **CTRL** during selection.

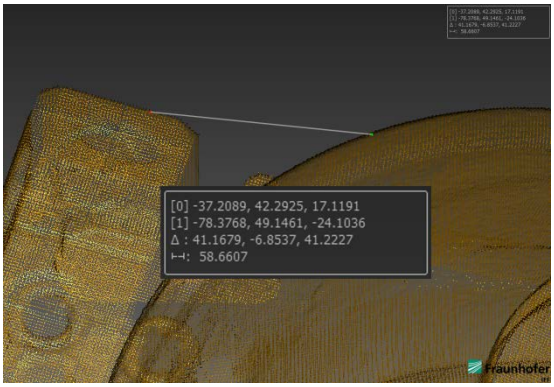
delete selected points

Selected points can be deleted by pressing **DEL**-Key.

NOTE: point cloud objects must be selected on order to select single points inside.

3.5 Distance Measurement

The ruler tool enables the 3D distance measurement between two clicked positions on the screen. **Press the ALT-Key when clicking** on objects on the screen and the underlying 3D coordinate is computed.



3.6 Viewing

(a) 

Changing the camera views to the scene.

A perspective (a) and an orthogonal projection (b) can be selected.

(b) 

A snapshot of current visualization is made by clicking on the camera icon (c). Images are saved in BMP or PNG format.

(c) 

3.7 Splitting and Merging point clouds

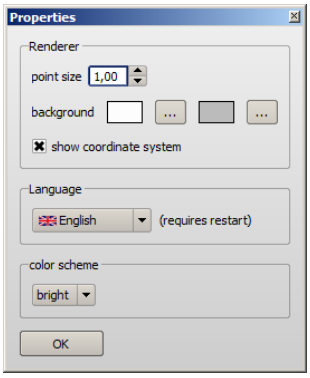
(a) 

The tool (a) splits the selected points from the active point cloud object and creates a new point cloud object in the object tree.

The tool (b) merges the selected point cloud objects into one single point cloud object.

(b) 

3.8 Program configuration



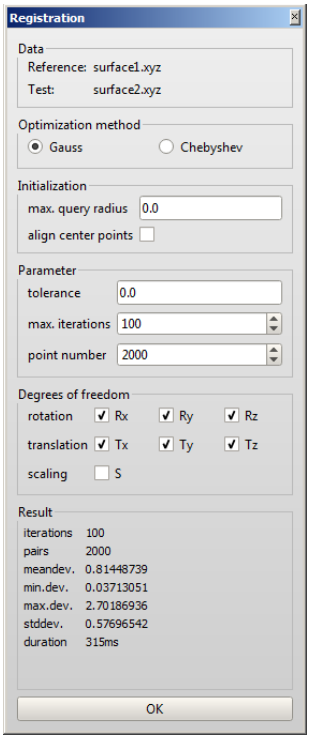
Properties

- Adjust point size
- Adjust the scene background
- Show coordinate frame
- Adjust program language
- Select the program's color scheme (dark/bright)

4 Algorithms for 3d point clouds

4.1 Registration / Alignment

The registration function aligns **two point clouds** or **one point cloud to an STL-mesh**. Both objects must be selected in the object tree. The object that has been selected **first** is used as the **static reference** data set and remains unchanged. The **second** object selected is used as the **dynamic data set** which is transformed iteratively until it is aligned to the first, static data set.



Data (shows the selected data sets)

- static (reference)
- dynamic (test data to be transformed)

Optimization method

- Gauss (minimize sum of squared distances)
- Chebyshev (minimize maximum absolute distance = minimum zone). Note that it is advisable to apply the Gauss method first. Chebyshev is only available for point to mesh registration

Initialization

- maximum search radius for the nearest neighbor(s) (0=auto)
- Define if the centers of gravities should be aligned first. This is useful if the initial positions of both data sets differ a lot.

Parameter

- Tolerance value that defines the remaining mean distance at which the algorithms stops the iterations
- Maximum number of iterations before the algorithm stops
- Number of data points to use for the alignment (0=all).
  - For the Gauss method it is often sufficient to only use a subset of the entire point data in order to reduce computation time (typically 2000 points is enough)
  - For the Chebyshev method the points with the largest distances are required. For the exact result all points should be used.
  - A minimum of 3 points is required anyway.

Degrees of freedom

- Enable rotation, translation and scaling and single axes.

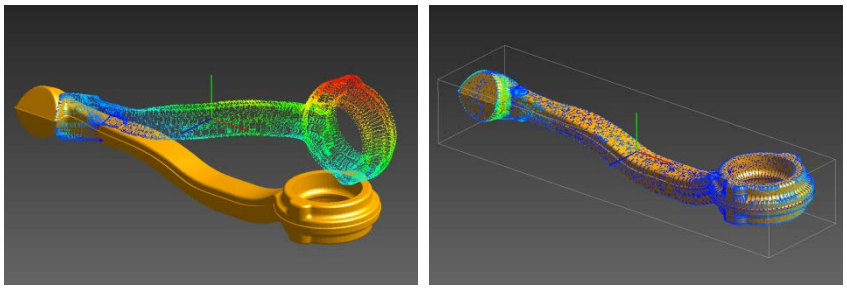


Figure 3:
alignment of a point cloud to a STL model

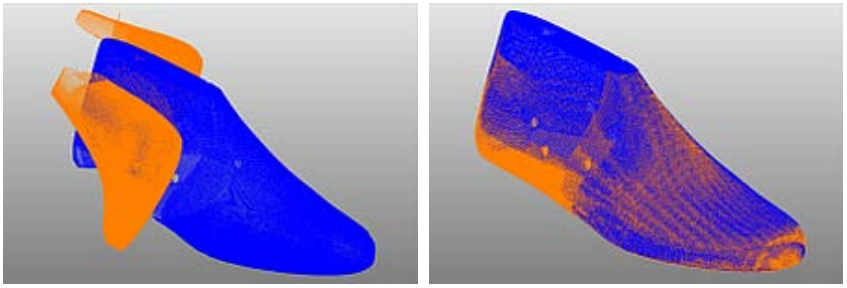
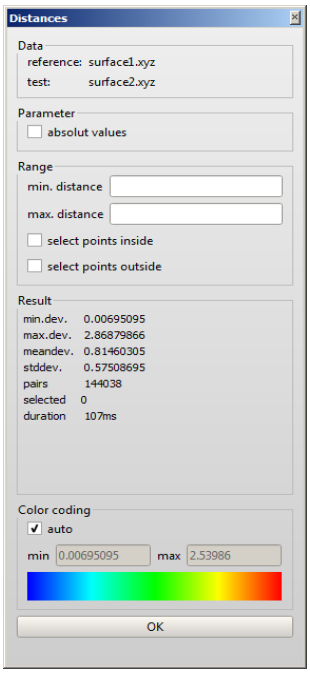


Figure 4:
alignment between two point clouds

4.2 Distance computation



Shows the selected data sets

- **Source**: first selected point set. To this data set the distance are computed.
- **Destination**: second selected point set. For all points of this data set the distances to 'src' are computed. Depending on the distance each point is shown in an appropriate color.

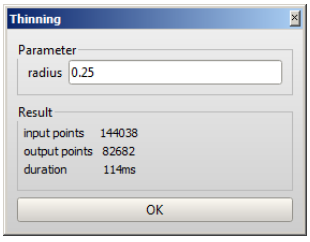
Parameter

- Minimum/maximum distance can be provided to include/exclude a range of points from the analysis and visualization
- Absolute distance can be shown instead of signed distances
- Points outside the minimum/maximum range can be selected in order to process/delete them later

Color Coding

- Manually change the color gradient

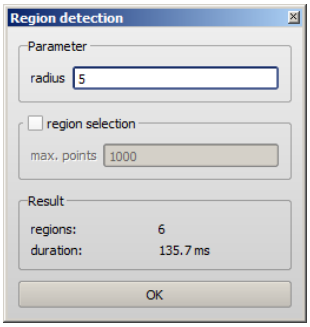
4.3 Thinning / Homogenization



Parameters

- Radius, minimal distance that one point should have to any other (in its spherical neighborhood).

4.4 Region Detection

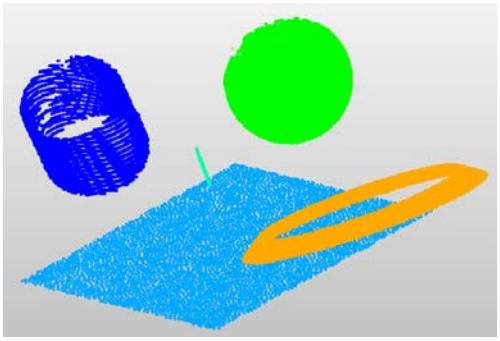


Parameter

- Radius that describes the maximum distance of a single point to the nearest region in order to belong to the same region.

The algorithm detects connected sets of points and gives them a different color. The functionality of the underlying library may be used to segment and separate objects within a point set.

"region selection" can be activated. In this case all regions with a maximum number of points (user-defined) are selected for removal. This is especially useful to remove single points, artifacts and smaller outlying point groups.



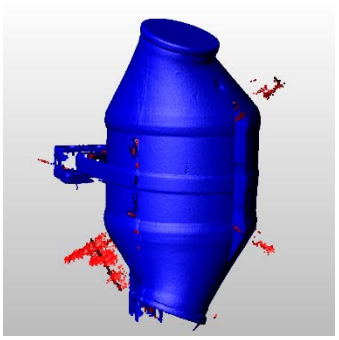
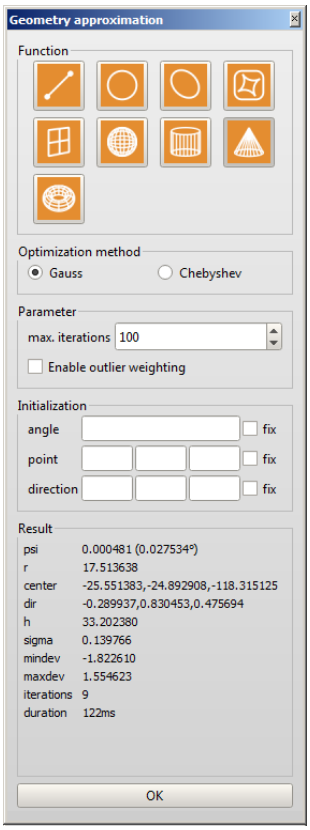


Figure 5: automatically detected regions of spatially connected points and outliers.

4.5 Geometry Approximation



Approximation functions for the following geometric primitives

- line, circle, ellipse, superellipse, plane, , sphere, cylinder, cone, torus

The function is applied to the object or the points currently selected.

Optimization method

Two numerical optimization methods are available: Gaussian or Chebyshev.

- Gaussian: minimizes the sum of squared distances (Levenberg-Marquardt)
- Chebyshev: minimizes the maximum deviation (minimum zone). Selectable for plane, sphere, cylinder, cone and torus.

Parameter

The option „outlier weighting“ enables a more robust approximation to extreme outlier data and may requires more iterations.

The parameters of each function can be initialized manually to increase robustness. Normally this is not required. The parameters may also be fixed so they remain unchanged and the free parameters are computed only (e.g. sphere approximation with a fixed/known radius in order to reconstruct the best-fit center point only).

2d geometries within 3d data sets

2-dimensional geometries can be approximated to 3d data by selecting a planar region.

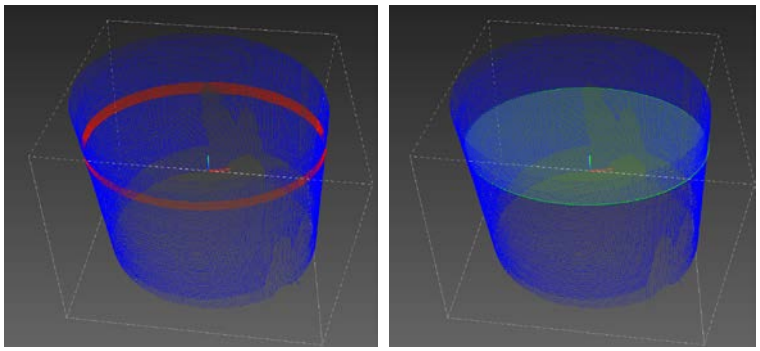
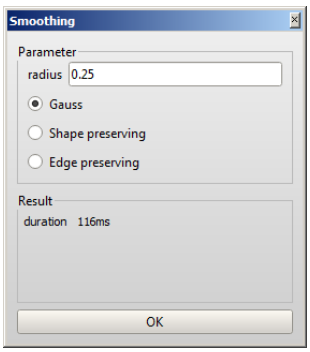


Figure 6: Selection of a planar region within a set of 3d points (left), and application of 2d geometry approximation to the selected point data (right).

4.6 Smoothing



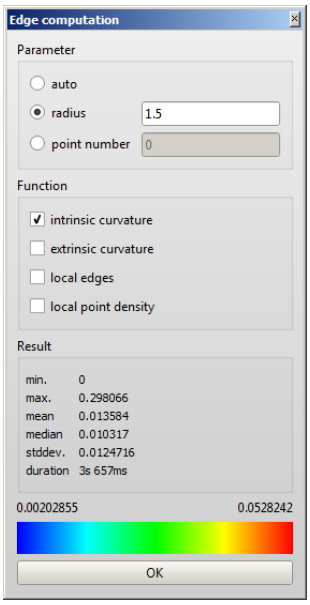
Parameters

- Radius that defines the spherical region that is used for the computation of the smoothing for each single point.

The smoothing function can be

- Gaussian: weighted average that not preserves shape or edges.
- Shape preserving: projection to local neighborhood
- Edge preserving: curvature depending smoothing

4.7 Surface features



The algorithms analyze the local neighborhood of a point and compute the surface features by using local filter and approximation operations.

Parameters

- **Radius/Point number:** Neighborhood/Kernel size that defines the local region that is used for the computation for each single point.
- **Intrinsic curvature:** Computes the local mean curvature using the principal curvature directions taking all points from the neighborhood into account. Sensitive to local surface changes.
- **Extrinsic curvature:** Curvature is approximated from a quadratic function. More sensitive than intrinsic curvature but longer computation time.
- **Local edges:** This algorithm approximates the magnitude of edges mainly depending on the local density. It is useful to detect holes.
- **Local density:** computes the number of points per area unit in the vicinity of a single point.

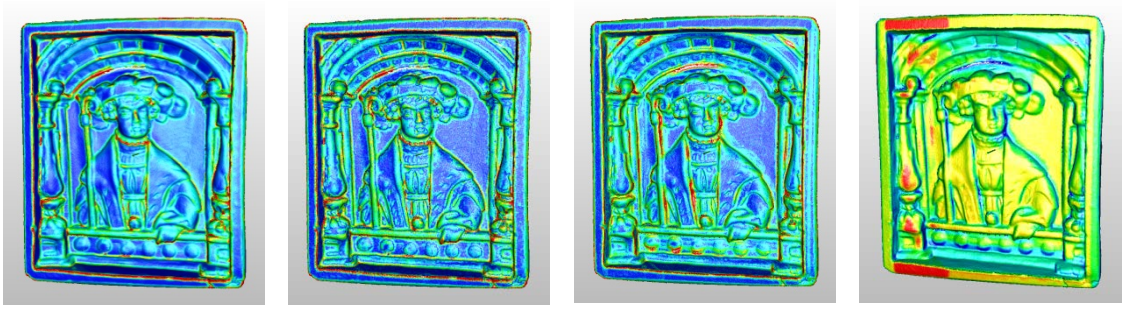
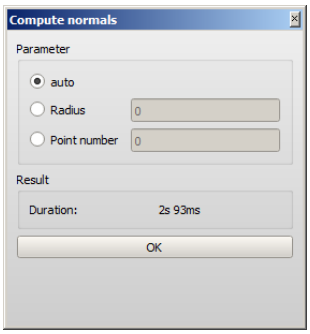


Figure 7: intrinsic curvature, extrinsic curvature, local edges, local density.

4.8 Normal vector computation



The algorithm approximates local surface normal vectors within a neighborhood specified by the parameters und shows the result as shaded point cloud using the resulting normal vectors.

Parameters

- **Auto:** automatically chooses the optimal parameters to compute the size of the local neighborhood
- **Radius:** radius of the local neighborhood
- **Point number:** neighborhood is defined by the k-nearest neighbors

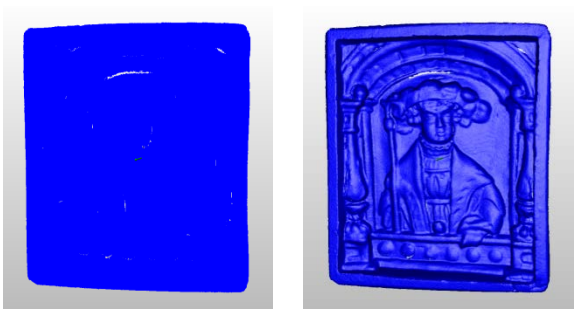


Figure 8: Raw point cloud (left) and shaded point cloud using the normal vectors (right).