Photogrammetry & Machine Vision

1. Image sensors	
(a) Fundamentals of image sensors	*I1 24.2*
(b) CCD image sensing	*I2 3.3*
(c) CMOS photodiodes	*I3 10.3*
2. Photogrammetric measurement pipeline	
(a) Measurement in images, correlations, etc.	*C1 17.3*
(b) Camera calibration and orientation	*C2 24.3*
(c) Point cloud processing, surface generation, texturing	*C3 31.3*
(d) Image modeling pipeline in practice	*C4 14.4*
3. Range based measurement pipeline	
(a) Active 3D sensors (laser, white light, ToF)	*C5 21.4*
(b) Modeling pipeline (alignment, processing, editing)	*C6 28.4*
4. Sensor integration	
(a) hand-held scanning, mobile mapping	*C7 5.5*
5. Demonstrations, labs and exercises	*12.5, 19.5, 26.5, 2.6*

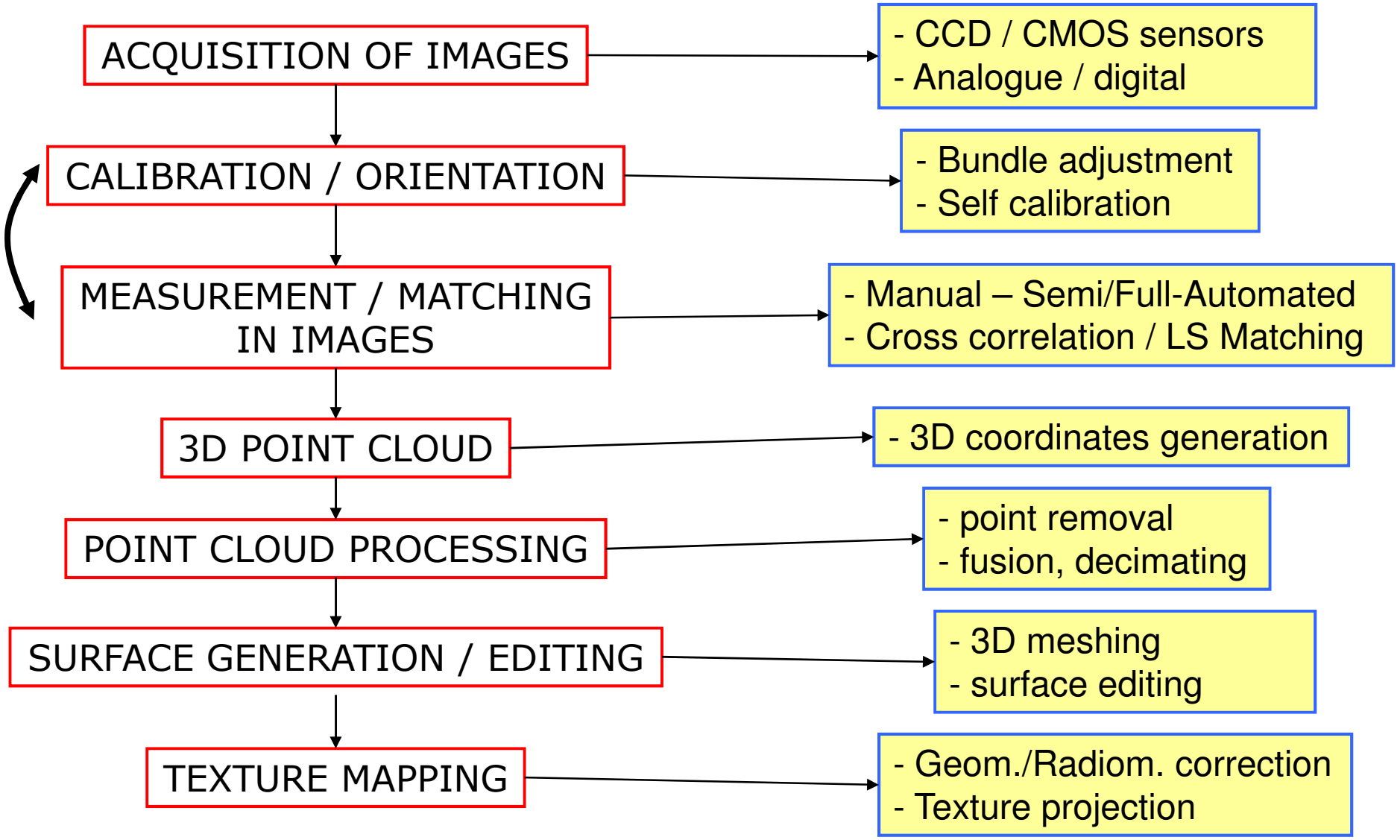
Point cloud processing, surface generation, texturing

1. 3D point cloud
   (a) General introduction
   (b) Photogrammetric pipeline - forward ray intersection
   (c) Types of 3D point clouds (volume, surface, edges, static, dynamic)

2. Point cloud processing
   (a) Raw point cloud (background, outliers, noise)
   (b) Point could editing (manual, automatic, removal points)
   (c) Point cloud processing (fusion, decimating, etc.)

3. Surface generation and texturing
   (a) 3D meshing
   (b) Surface editing (hole filling, smoothing)
   (c) Texturing methods (projection, mesh texture, etc.)

Photogrammetric measurement pipeline



Photogrammetric measurement pipeline

ACQUISITION OF IMAGES

CALIBRATION / ORIENTATION

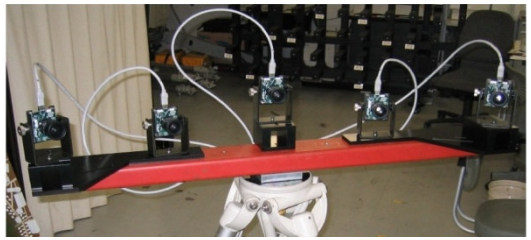
MEASUREMENT / MATCHING IN IMAGES

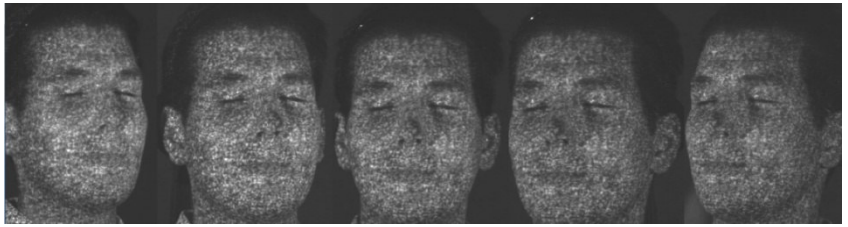
3D POINT CLOUD

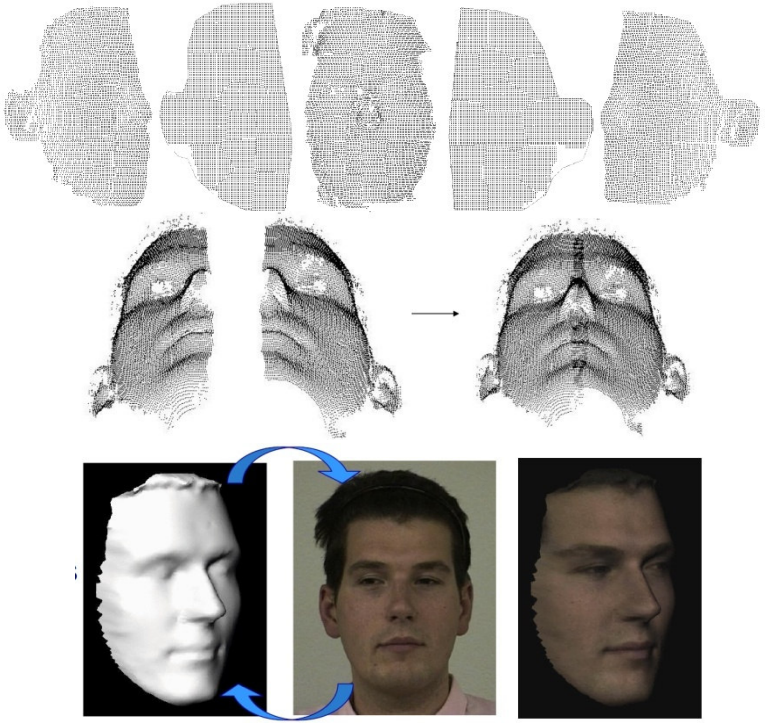
POINT CLOUD PROCESSING

SURFACE GENERATION / EDITING

TEXTURE MAPPING

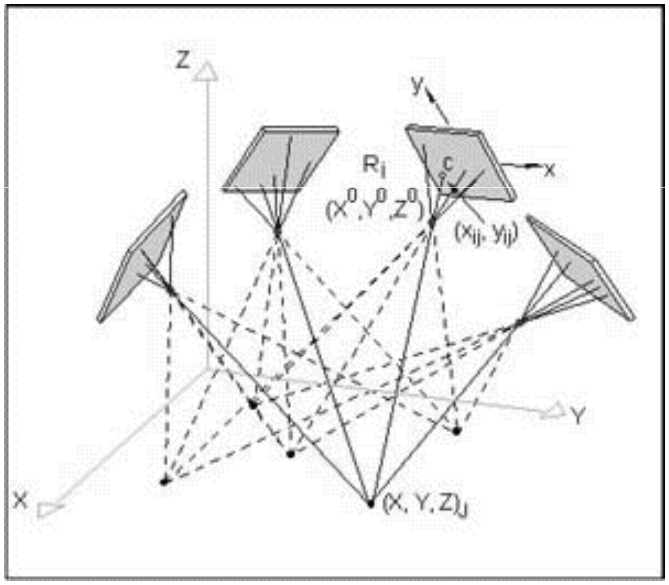




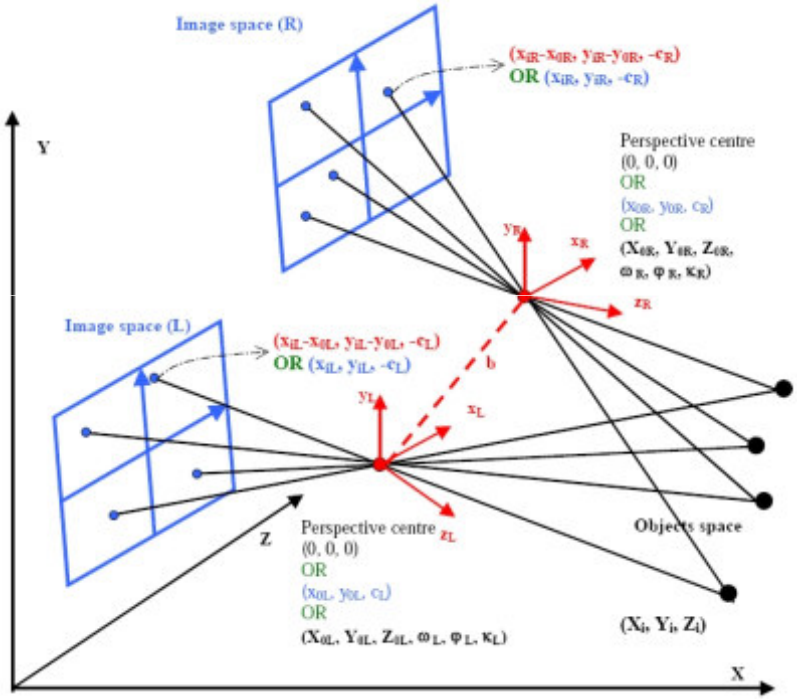


Generation of 3D point cloud

- Bundle adjustment (e.g. self calibration)
- Forward ray intersection



Self calibration



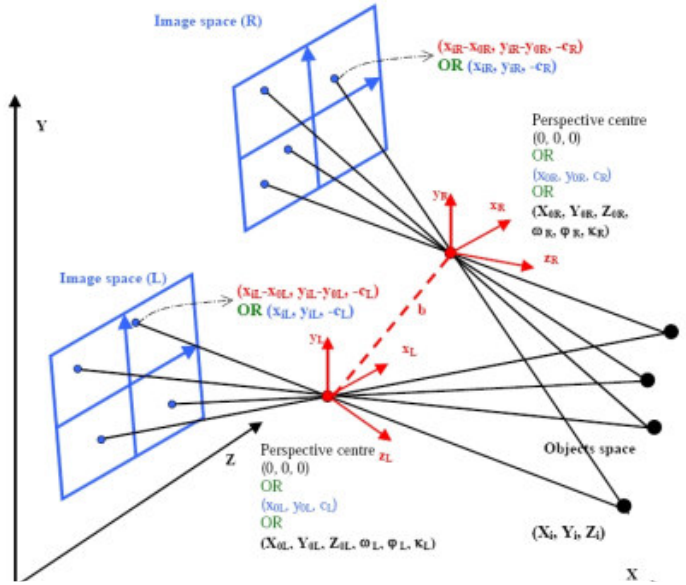
Forward ray intersection

Forward ray intersection 1

Ideal case:

- Perfect camera calibration and orientation
- Perfect image coordinate measurement

-> Rays “image point – projection center” intersect in the 3D object point



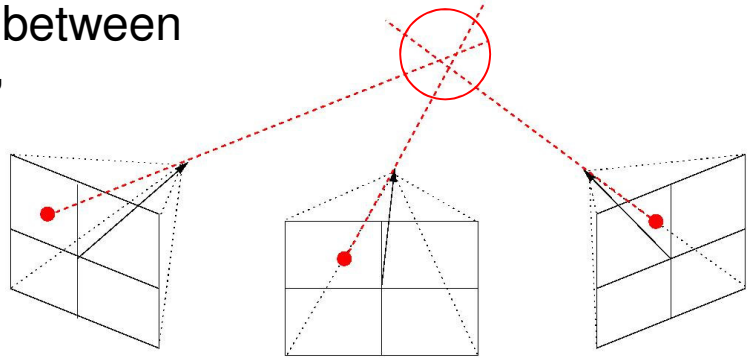
Ideal case (stereo images)

Normal case:

- Rays do not intersect in a precise location

-> Intersection point: location where the distances between all the rays is minimal: “forward ray intersection”

- Solved by least squares estimation, based on collinearity equations



Normal situation (multiple images)

Forward ray intersection 2

Collinearity equations + distortion terms:

$$
\begin{aligned}
x' &= x_p - c \cdot \frac{r_{11} \cdot (X - x_0) + r_{21} \cdot (Y - y_0) + r_{31} \cdot (Z - z_0)}{r_{13} \cdot (X - x_0) + r_{23} \cdot (Y - y_0) + r_{33} \cdot (Z - z_0)} + d\bar{x}' \\
&= \bar{x}' + d\bar{x}' \\
y' &= y_p - c \cdot \frac{r_{12} \cdot (X - x_0) + r_{22} \cdot (Y - y_0) + r_{32} \cdot (Z - z_0)}{r_{13} \cdot (X - x_0) + r_{23} \cdot (Y - y_0) + r_{33} \cdot (Z - z_0)} + d\bar{y}' \\
&= \bar{y}' + d\bar{y}'.
\end{aligned}
$$

equation (*)

$(\bar{x}', \bar{y}')$ uncorrected terms

$d\bar{x}', d\bar{y}'$ correcting terms

Least squares estimation:

- Given parameters: external orientation, internal orientation, distortion parameters
- Observations: image coordinates of corresponding points (x_i', y_i') in *n* images
- Unknowns: object space coordinates of observed points *(X,Y,Z)*

Forward ray intersection 3

Derivatives of the uncorrected terms $(\bar{x'}, \bar{y'})$ of the collinearity equations with respect to *(X,Y,Z)* :

$$\left(\frac{\partial \bar{x'}}{\partial X}\right)_i = -\frac{c_i}{{N_i}^2} \cdot (N_i r_{11i} - Z x_i r_{13i}) \quad \left(\frac{\partial \bar{y'}}{\partial X}\right)_i = -\frac{c_i}{{N_i}^2} \cdot (N_i r_{12i} - Z y_i r_{13i})$$
$$\left(\frac{\partial \bar{x'}}{\partial Y}\right)_i = -\frac{c_i}{{N_i}^2} \cdot (N_i r_{21i} - Z x_i r_{23i}) \quad \left(\frac{\partial \bar{y'}}{\partial Y}\right)_i = -\frac{c_i}{{N_i}^2} \cdot (N_i r_{22i} - Z y_i r_{23i})$$
$$\left(\frac{\partial \bar{x'}}{\partial Z}\right)_i = -\frac{c_i}{{N_i}^2} \cdot (N_i r_{31i} - Z x_i r_{33i}) \quad \left(\frac{\partial \bar{y'}}{\partial Z}\right)_i = -\frac{c_i}{{N_i}^2} \cdot (N_i r_{32i} - Z y_i r_{33i})$$

$$\begin{aligned} Z x_i &= r_{11i} \cdot (X - x_{0i}) + r_{21i} \cdot (Y - y_{0i}) + r_{31i} \cdot (Z - z_{0i}) \\ Z y_i &= r_{12i} \cdot (X - x_{0i}) + r_{22i} \cdot (Y - y_{0i}) + r_{32i} \cdot (Z - z_{0i}) \\ N_i &= r_{13i} \cdot (X - x_{0i}) + r_{23i} \cdot (Y - y_{0i}) + r_{33i} \cdot (Z - z_{0i}). \end{aligned}$$

i = 1..n image number

The derivatives of the correcting terms $d\bar{x'}, d\bar{y'}$ are very small and can be omitted, thus the derivatives of the observations are:

$$\left(\frac{\partial x'}{\partial X}\right)_i \cong \left(\frac{\partial \bar{x'}}{\partial X}\right)_i \quad \left(\frac{\partial y'}{\partial X}\right)_i \cong \left(\frac{\partial \bar{y'}}{\partial X}\right)_i$$
$$\left(\frac{\partial x'}{\partial Y}\right)_i \cong \left(\frac{\partial \bar{x'}}{\partial Y}\right)_i \quad \left(\frac{\partial y'}{\partial Y}\right)_i \cong \left(\frac{\partial \bar{y'}}{\partial Y}\right)_i$$
$$\left(\frac{\partial y'}{\partial Z}\right)_i \cong \left(\frac{\partial \bar{x'}}{\partial Z}\right)_i \quad \left(\frac{\partial y'}{\partial Z}\right)_i \cong \left(\frac{\partial \bar{y'}}{\partial Z}\right)_i .$$

Forward ray intersection 4

The unknown vector *x* , the design matrix *A* and the observation vector *l* are:

$$x = (dX, dY, dZ)^T$$

$$\mathbf{A} = \begin{bmatrix} \left(\frac{\partial \bar{x}'}{\partial X}\right)_1 & \left(\frac{\partial \bar{x}'}{\partial Y}\right)_1 & \left(\frac{\partial \bar{x}'}{\partial Z}\right)_1 \\ \left(\frac{\partial \bar{y}'}{\partial X}\right)_1 & \left(\frac{\partial \bar{y}'}{\partial Y}\right)_1 & \left(\frac{\partial \bar{y}'}{\partial Z}\right)_1 \\ \vdots & \vdots & \vdots \\ \left(\frac{\partial \bar{x}'}{\partial X}\right)_n & \left(\frac{\partial \bar{x}'}{\partial Y}\right)_n & \left(\frac{\partial \bar{x}'}{\partial Z}\right)_n \\ \left(\frac{\partial \bar{y}'}{\partial X}\right)_n & \left(\frac{\partial \bar{y}'}{\partial Y}\right)_n & \left(\frac{\partial \bar{y}'}{\partial Z}\right)_n \end{bmatrix}$$

$$l = (x'_1 - \hat{x}'_1, y'_1 - \hat{y}'_1, \cdots, x'_n - \hat{x}'_n, y'_n - \hat{y}'_n)^T$$

where

dX, dY, dZ	changes of the unknown object coordinates from the estimations,
x'_i, y'_i	observed (i.e., measured) image coordinates of the point in image i,
$\hat{x}'_i, \hat{y}'_i$	the estimated image coordinates, computed by backprojecting the estimated 3–D point onto the image i according to equation *(*)*
n	number of images.

Forward ray intersection 5

Gauss-Markov model of least squares:

$$
\begin{aligned}
\hat{x} &= \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T l && \text{solution vector,}\\
v &= \mathbf{A}\hat{x} - l && \text{residual vector,}\\
\hat{\sigma}_0^2 &= \frac{v^T v}{2n-3} && \text{variance factor,}\\
\mathbf{Q} &= \hat{\sigma}_0^2 \cdot \left(\mathbf{A}^T\mathbf{A}\right)^{-1} && \text{covariance matrix,}\\
\hat{\sigma}_i^2 &= \mathbf{Q}_{ii} && \text{variance factor of the single unknowns.}
\end{aligned}
$$

System is solved iteratively:

- Compute the estimated unknown vector $\hat{x}$
- Update design matrix *A* and observation vector *l*
  (backprojecting the new estimated *X,Y,Z* onto the images)
- Iteration until the changes of the unknowns are smaller than a threshold (e.g. 0.5%)

Useful:

the variance factor of the single unknowns $\hat{\sigma}_i$ expresses the theoretical Precision of the computed 3D coordinates in the 3 axis directions.

Point cloud processing, surface generation, texturing

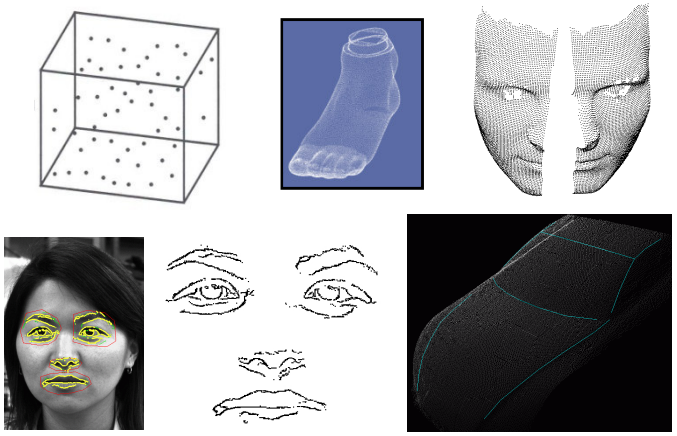
1. 3D point cloud
   (a) General introduction
   (b) Photogrammetric pipeline - forward ray intersection
   (c) Types of 3D point clouds (volume, surface, edges, static, dynamic)

2. Point cloud processing
   (a) Raw point cloud (background, outliers, noise)
   (b) Point could editing (manual, automatic, removal points)
   (c) Point cloud processing (fusion, decimating, etc.)

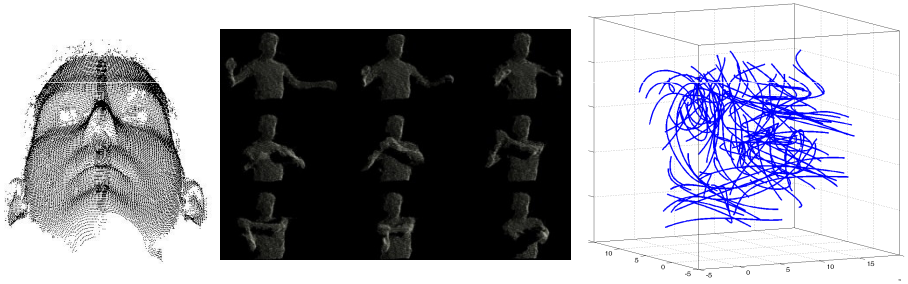
3. Surface generation and texturing
   (a) 3D meshing
   (b) Surface editing (hole filling, smoothing)
   (c) Texturing methods (projection, mesh texture, etc.)

Different types of 3D point clouds

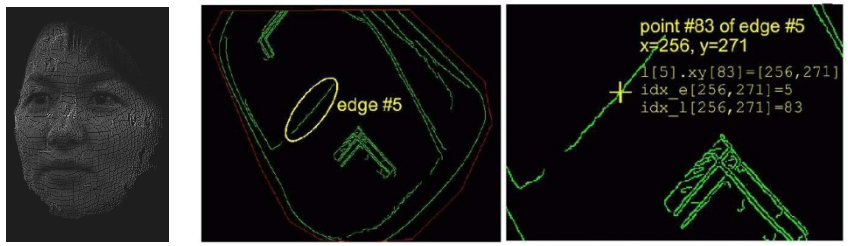
1. Representing different “objects” type
   (a) 3D volumes
   (b) Surfaces (continuous or with discontinuities)
   (c) 3D curves/lines in space
   (d) Combined structures



2. Static and dynamic point clouds
   (a) Static 3D point clouds
   (b) Dynamic 3D points clouds
   (c) 4D point clouds (tracked during time)

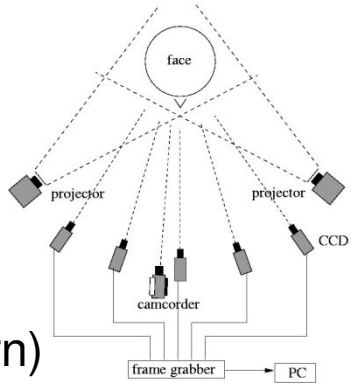


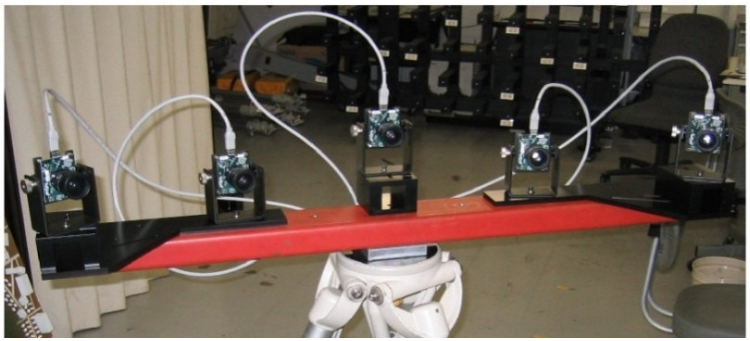
3. Additional information
   (a) Point number / ordering
   (b) Color information
   (c) Accuracy-matching information
   (d) Complex structures (linked information, point type, element number, etc.)



Example 1: surface, static, additional info color

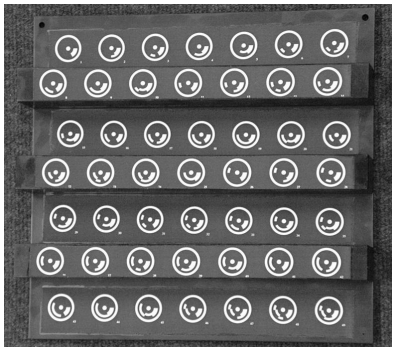
Human face measurement





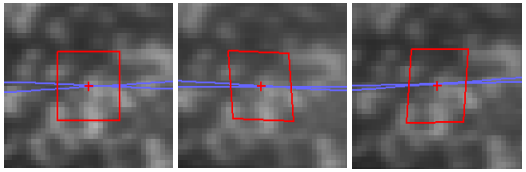
System:

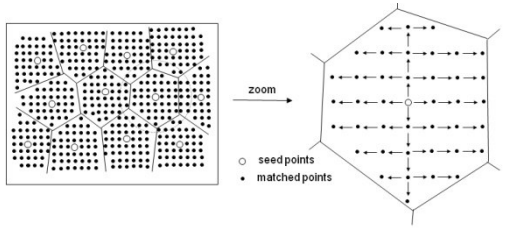
- 5 camera synchronized
  1 color camera
- 2 slides projectors (random pattern)
- calibration object with coded targets

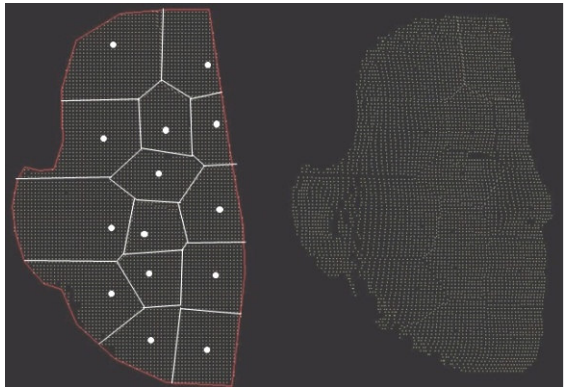


Method:

- calibration with reference object
- automatic multi-image matching
  -> dense set of corres. points



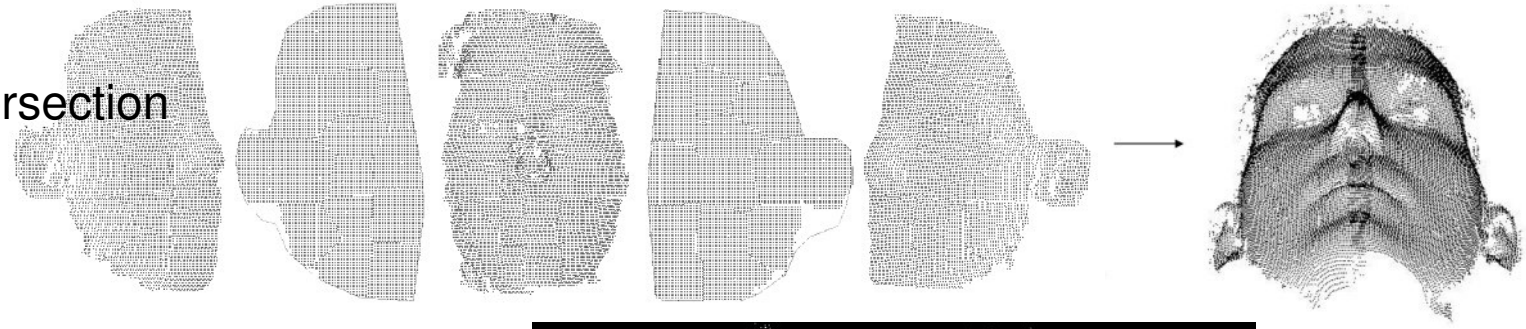




Example 1: surface, static, additional info color

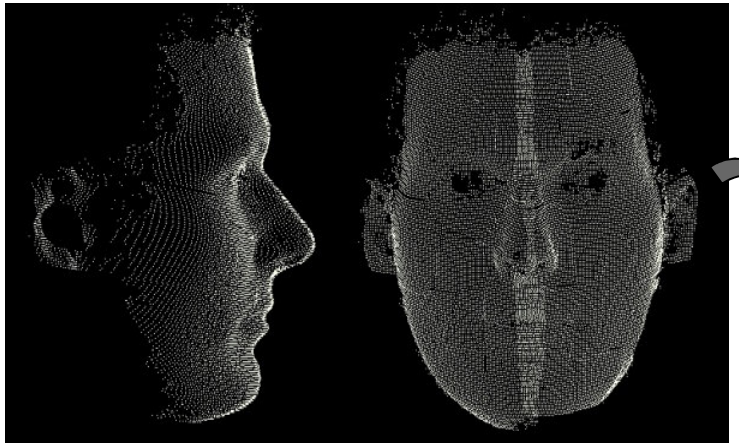
Method:

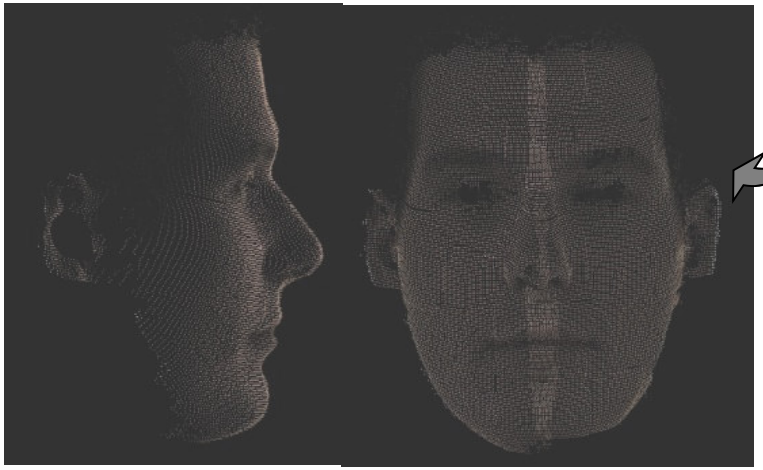
- forward ray intersection



Point cloud with color information:

- Backprojection of 3D points on calibrated color image, read of color data (RGB)
- Point cloud:
  - lookup table {color#, R G, B}
  - list of points {#, X, Y, Z, color#}



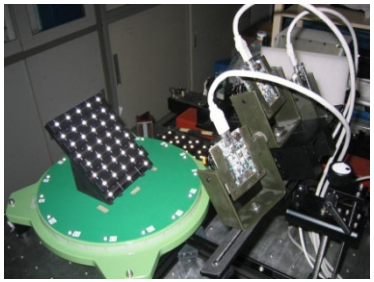


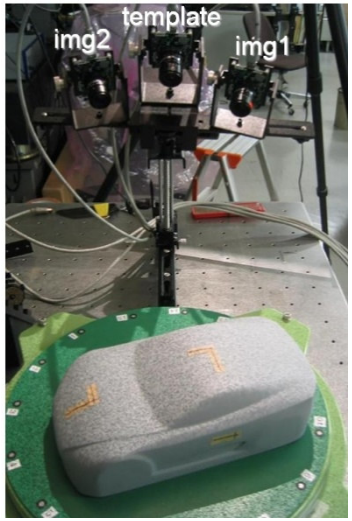
```
20253
256
4 3 4
100 115 101
34 60 30
156 170 154
103 59 19
156 116 84
60 25 6
106 87 50
111 142 111
66 86 59
11 31 9
164 143 116
140 97 61
73 59 34
131 115 90
197 171 146
103 89 85
38 33 24
103 61 59
189 145 128
64 40 24
```

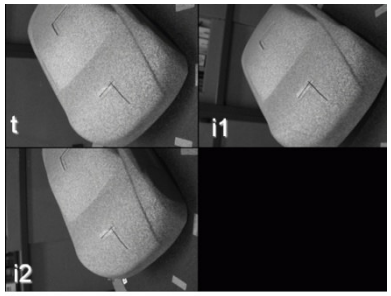
1	49.1197	131.7220	267.5411	184
2	49.1047	130.4994	266.4622	63
3	49.0964	129.4453	264.9246	11
4	49.0844	128.3091	263.5016	11
5	49.0760	127.0986	262.3006	119
6	49.0710	125.8547	261.2533	63
7	49.0579	124.6526	260.0439	119
8	49.0475	123.8343	257.7085	63
9	49.0447	122.7645	255.9879	199
10	49.0368	121.8031	253.9491	199
11	49.0487	120.6868	252.3029	49
12	49.0195	119.6967	250.2116	55
13	48.9867	118.7787	247.9207	123
14	48.9781	117.5208	246.5378	103
15	48.9670	116.5051	244.4916	7
16	48.9524	115.5155	242.5933	146
17	49.1282	134.0714	269.7939	63
18	49.1279	135.3336	270.5885	184
19	49.1325	136.4621	271.7486	63
20	49.1401	137.6223	272.9413	184
21	49.1333	138.9606	273.5245	184
22	49.1370	140.1906	274.3748	184
23	49.1403	141.4479	275.0374	62
24	49.1455	142.7421	275.5659	62

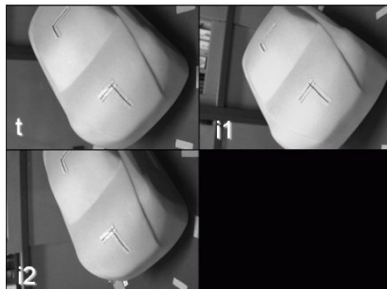
Example 2: 3D curves, static, linked structure (edges)

3D extraction of edges







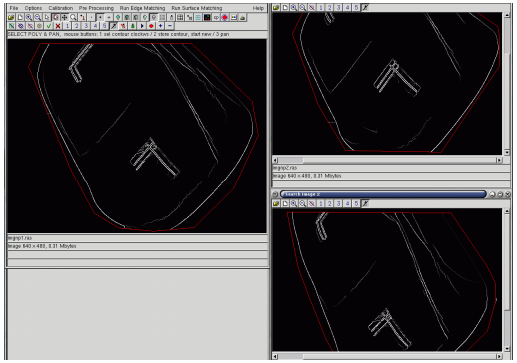


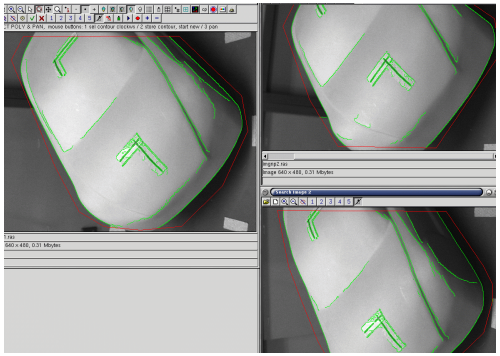
System:

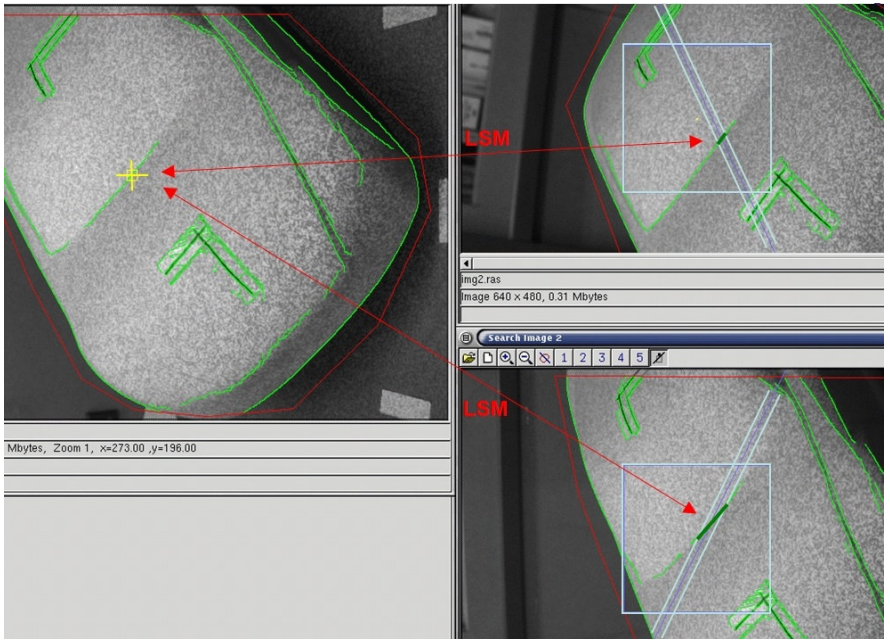
- 3 camera
- random pattern projection
- calibration object with
  regular distributed targets

Method:

- calibration with reference object
- edge (Canny, SUSAN) extraction in images
- matching of edge points by LSM







Example 2: 3D curves, static, linked structure (edges)

Point cloud:

Forward ray intersection -> 3D edges

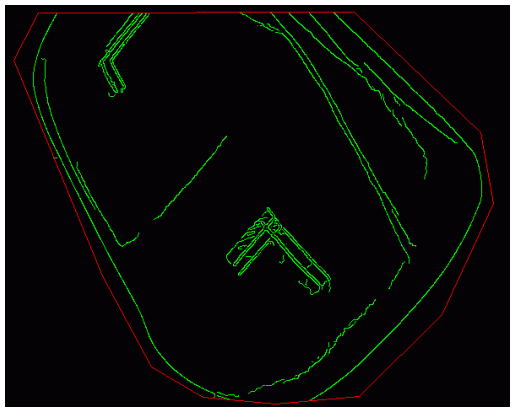
3D edges:

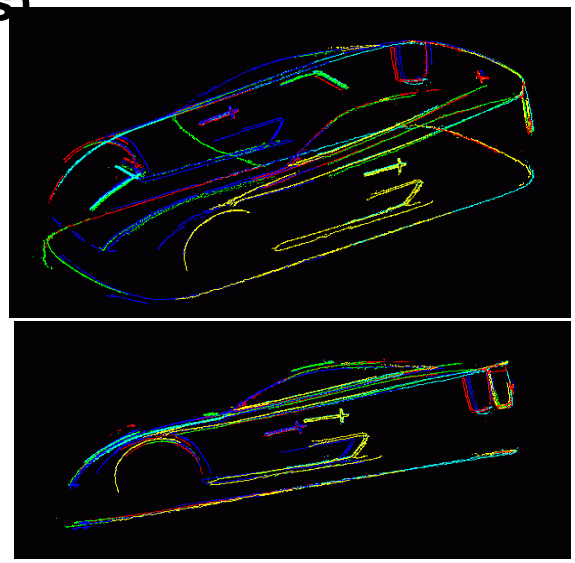
{edg#

{pt#,X,YZ}

}

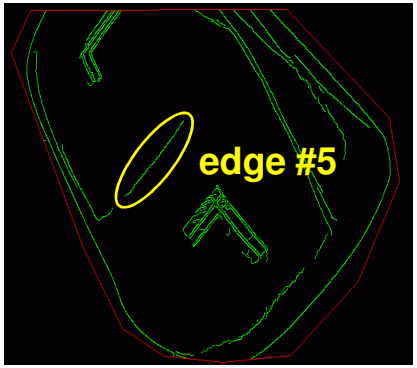
```
208
4
   4      -8.357961      86.432351      16.179710
   3      -8.394095      86.807865      18.326264
   2      -8.423295      86.445281      18.233076
   1      -8.458310      86.073040      18.130099
88
  17       4.795572     107.704972      37.667882
  16       4.616520     107.161324      37.158550
  19       5.022157     106.857872      36.939062
  22       5.427253     106.934678      37.538506
  23       5.818026     106.676428      37.929777
  25       6.202199     106.450813      38.435772
  28       6.585283     106.157558      38.651993
  32       6.950610     106.021796      39.433740
  37       7.335319     105.780109      39.838773
  36       7.345116     105.437097      39.674493
  42       7.727261     105.176056      39.761104
  41       7.736566     104.867786      39.827069
  40       7.737465     104.550337      39.883887
  44       8.108360     104.603236      40.203418
  46       8.479717     104.672375      40.598159
  48       8.832128     104.718034      40.767745
  52       9.194300     104.527145      41.418401
  51       9.189886     104.176308      41.256598
  50       9.184122     103.889600      41.427476
  55       9.566618     103.636324      41.752906
```

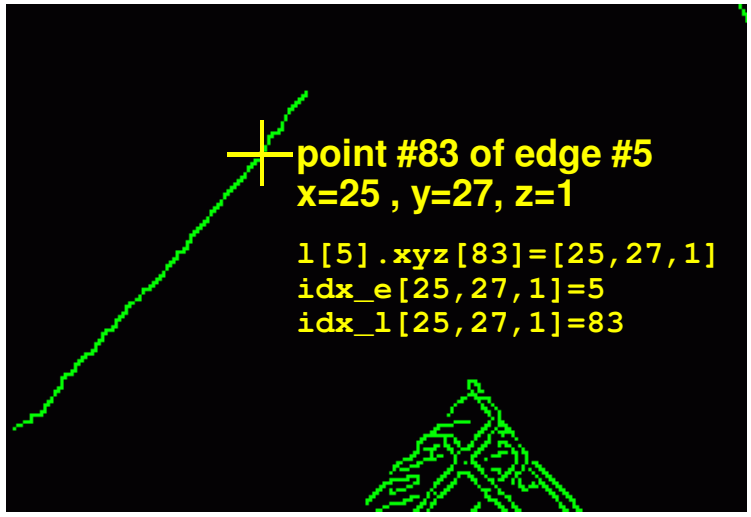




Complex linked structure for fast selection, analysis, etc.:

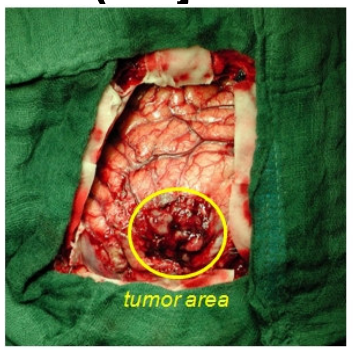
```
n:              number of edges i=1..n
n[i]:           number of points for edge i
l[i]:           list of points for edge i
  id[j]:        [id1,id2,..], point id numbers
  xyz[j]:       [x1,y1,z1,x2,y2,z2,..], list of edge pts
idx_e[x,y,z]:   x,y,z -> edge # (i)
idx_l[x,y,z]:   x,y,z -> pt # in point list (i)
```



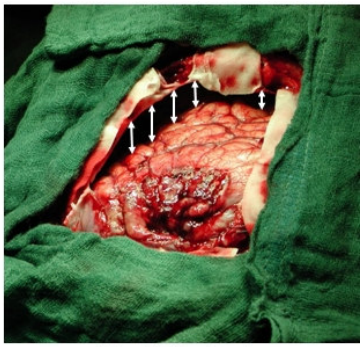


Example 3: 4D point cloud, dynamic, structure (trajectories)

3D tracking of brain surface



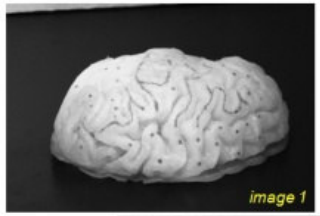
Craniotomy

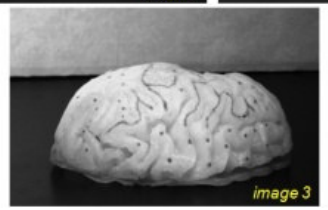


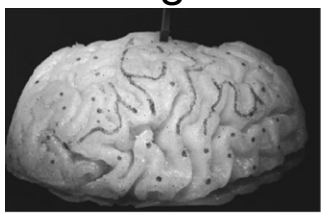
After 5-10 minutes

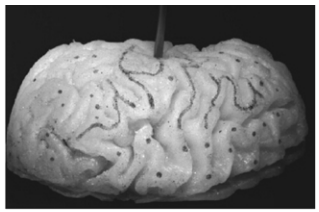
System:

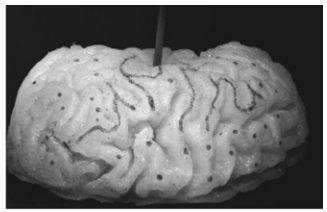
- 3 video camera synchronized
- image sequences
- calibration object with regular distributed targets

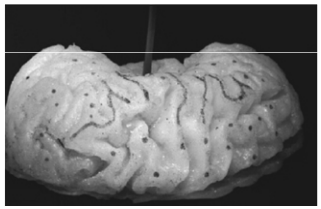


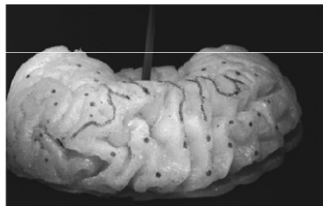








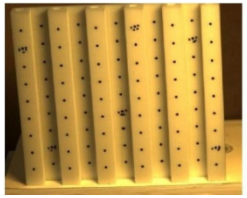


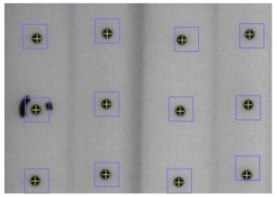


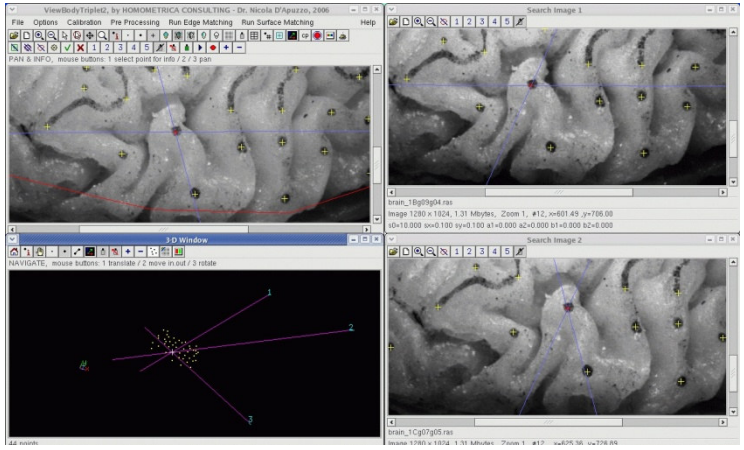
Method:

- calibration with reference object
- surface: multi-image matching

  + forward ray intersection



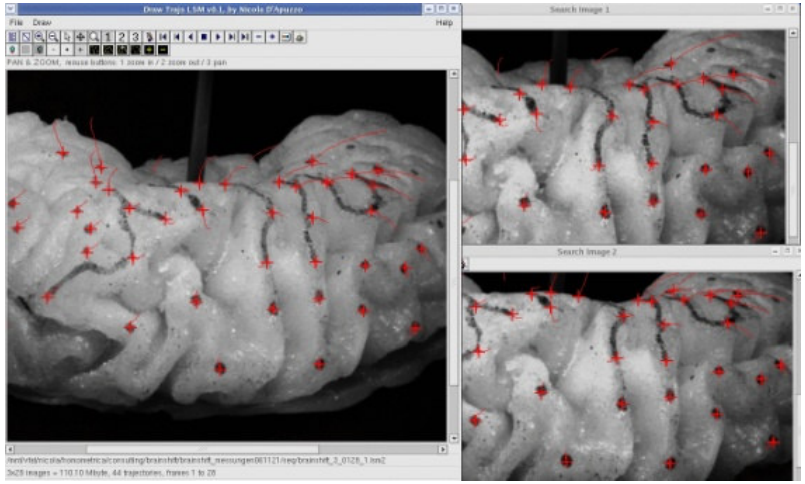


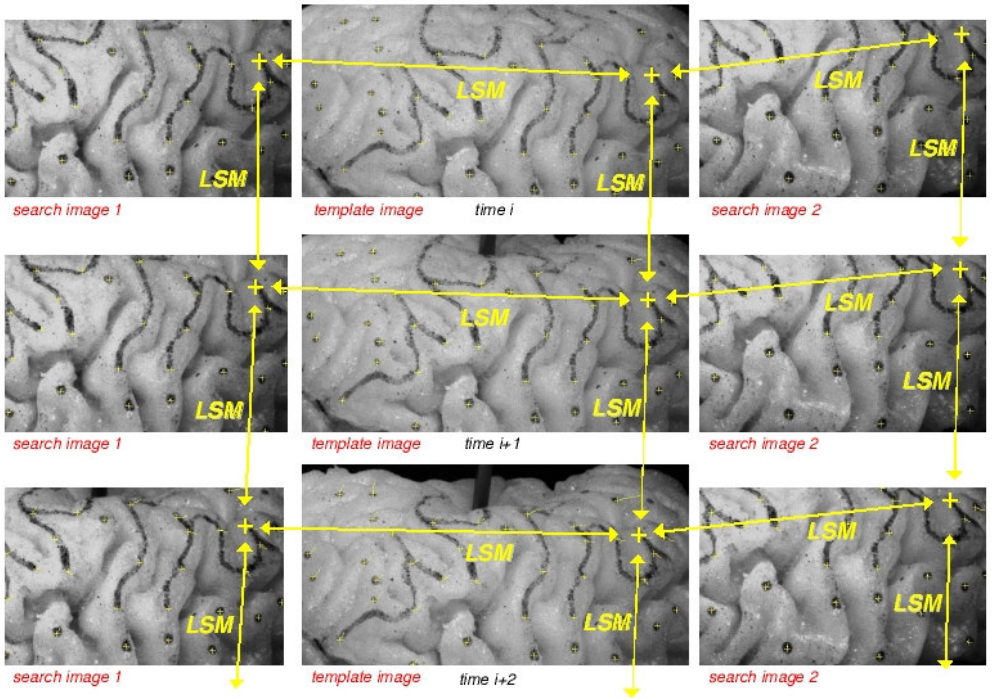


Example 3: 4D point cloud, dynamic, structure (trajectories)

Method:

- multi-image tracking of single points
  -> multi-2D trajectories
- forward ray intersection -> 3D traject.





Point cloud:

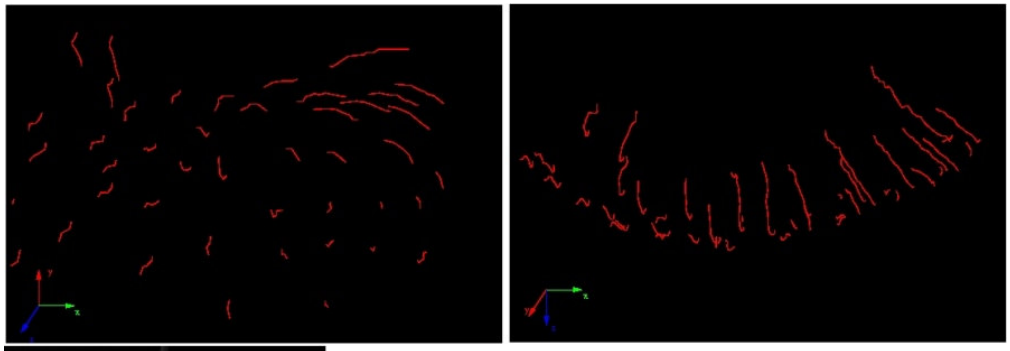
- 3D trajectories:

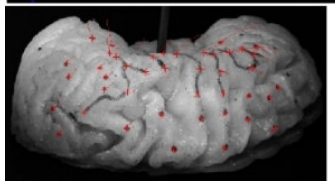
{tr#

{fr#, X,Y,Z}

}

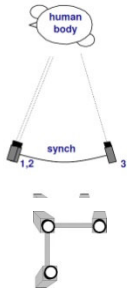
```
1
1    137.501   158.592    19.375       0.001
2    137.448   158.420    19.028       0.002
3    137.361   158.284    19.410       0.001
4    137.321   158.239    19.427       0.001
5    137.164   158.104    20.137       0.001
6    136.892   157.787    20.618       0.002
7    136.625   157.319    20.334       0.002
8    136.483   156.979    19.996       0.001
9    136.404   156.671    19.774       0.001
10   136.380   156.434    19.520       0.002
11   136.386   156.147    19.425       0.002
12   136.395   155.893    19.142       0.002
13   136.428   155.626    18.935       0.002
14   136.519   155.409    18.942       0.003
15   136.674   155.063    18.756       0.003
-1
2
1    146.015   160.335    24.869       0.000
2    145.981   160.159    24.447       0.001
3    145.906   160.036    24.759       0.001
4    145.870   159.986    24.847       0.001
5    145.737   159.858    25.550       0.001
6    145.522   159.506    25.933       0.002
7    145.379   158.921    25.474       0.002
8    145.338   158.480    25.217       0.002
9    145.370   158.030    24.812       0.002
10   145.453   157.627    24.467       0.002
```





Example 4: 4D dense point clouds, dynamic, surface + trajectories

3D human body surface tracking

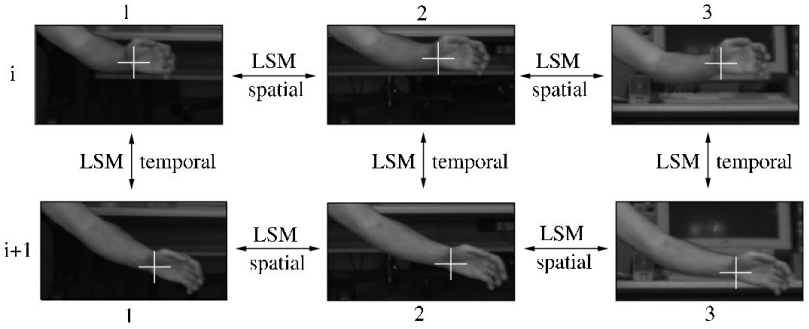


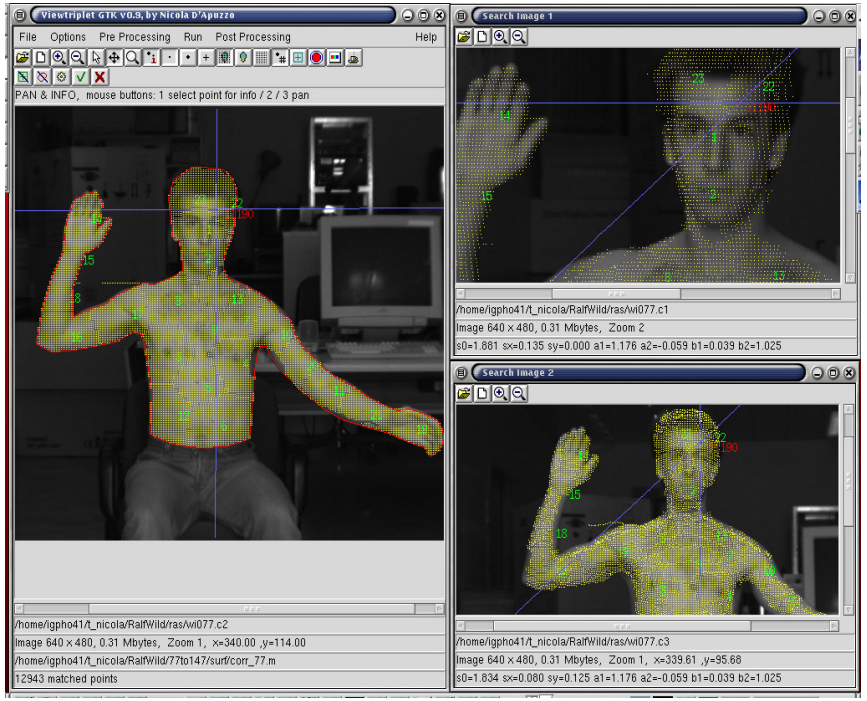
System:

- 3 video camera synchronized
- image sequences

Method:

- calibration with point cloud + scale
- Surface measurement and tracking:
  automatic multi-image matching
  + multi-image tracking of dense pt.set



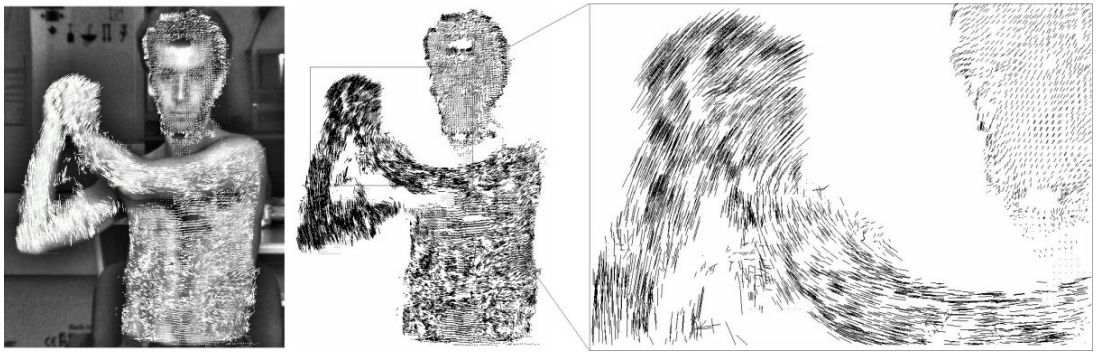


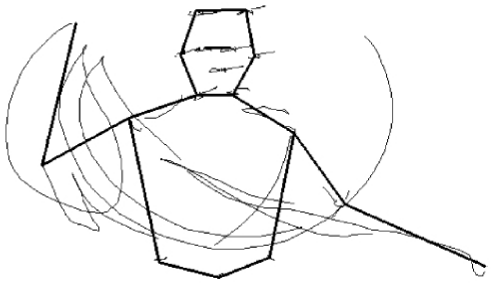
Example 4: 3D dense point clouds, dynamic, structure (trajectories)

Point clouds:

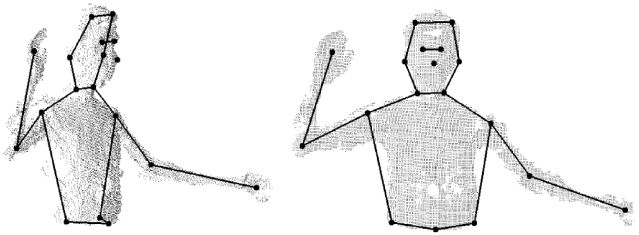
1. corresponding points for each frame
   -> forward ray intersection
   -> 3D point cloud for each frame

2. Dense set of multi-2D trajectories
   -> forward ray intersection
   -> dense 4D point cloud





3. Further processing
   e.g. tracking of key points
   in 4D point cloud



Point cloud processing, surface generation, texturing

1. 3D point cloud
   (a) General introduction
   (b) Photogrammetric pipeline - forward ray intersection
   (c) Types of 3D point clouds (volume, surface, edges, static, dynamic)

2. Point cloud processing
   (a) Raw point cloud (background, outliers, noise)
   (b) Point could editing (manual, automatic, removal points)
   (c) Point cloud processing (fusion, decimating, etc.)

3. Surface generation and texturing
   (a) 3D meshing
   (b) Surface editing (hole filling, smoothing)
   (c) Texturing methods (projection, mesh texture, etc.)

Photogrammetric measurement pipeline

ACQUISITION OF IMAGES

CALIBRATION / ORIENTATION

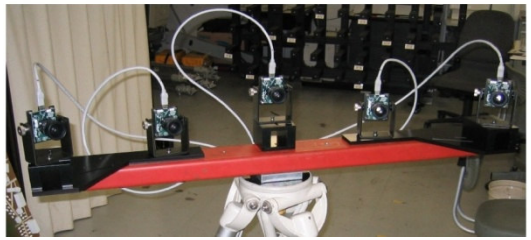
MEASUREMENT / MATCHING IN IMAGES

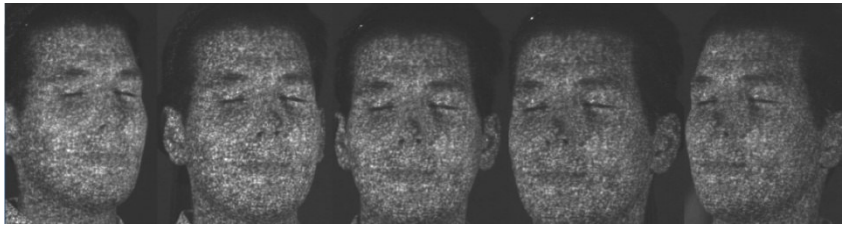
3D POINT CLOUD

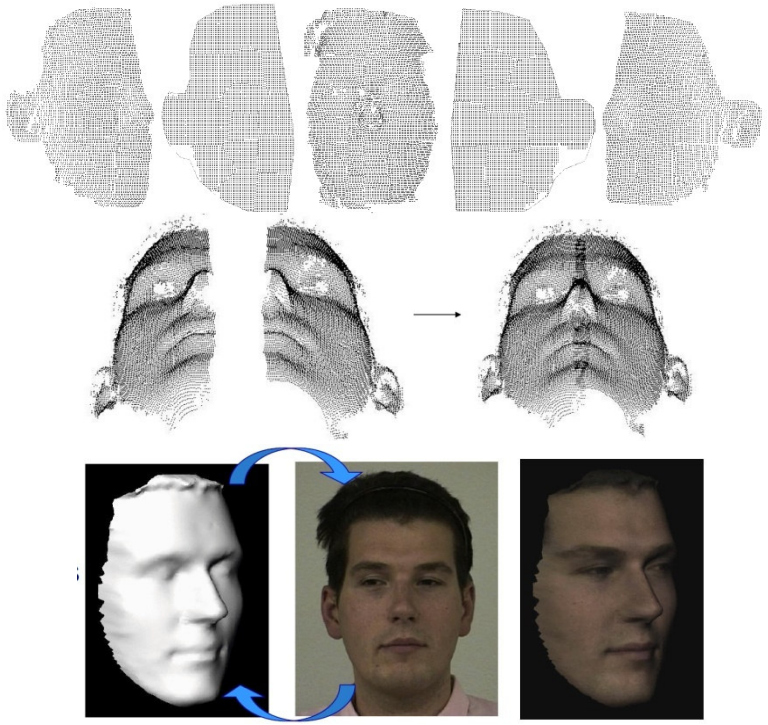
POINT CLOUD PROCESSING

SURFACE GENERATION / EDITING

TEXTURE MAPPING







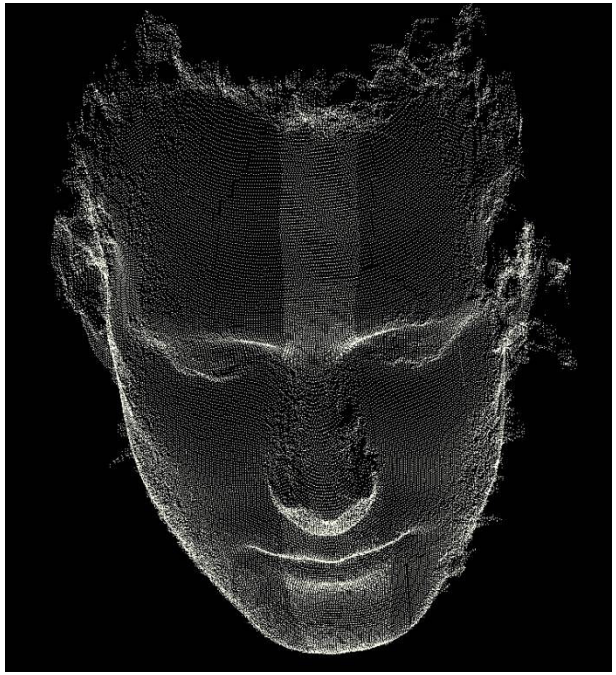
Raw point cloud

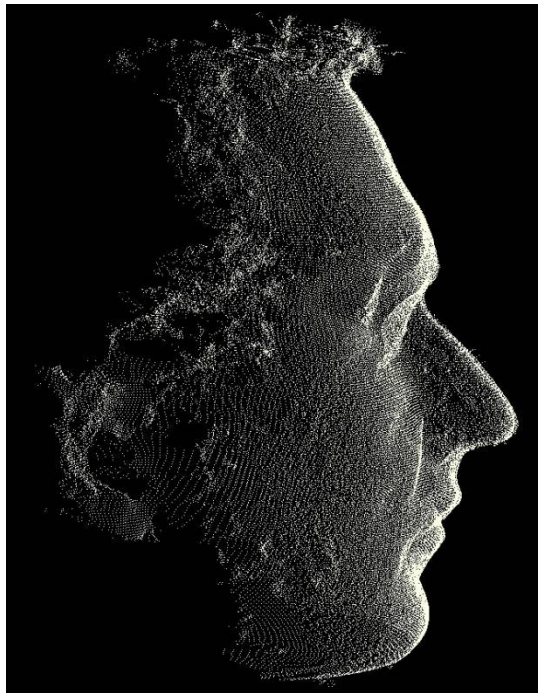
Point clouds resulting from the measurement process are not “clean”

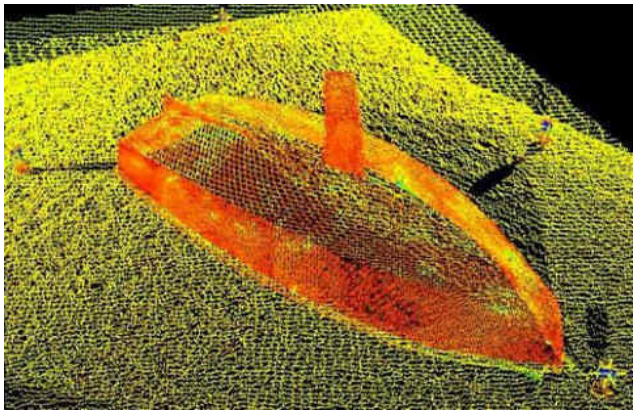
- Background points - not useful/interesting for the measurement task
- Outliers and noise caused by errors in the measurement (e.g. false match)
- Multiple point clouds - overlapping areas

-> point cloud has to be processed before generating a 3D mesh surface

a well-prepared point cloud leads to strong time saving in the further surface editing/modeling processes







Raw point cloud

Basic point cloud editing and processing:

- Manual selection of irrelevant points: e.g. background points, large outliers or manual selection of relevant points: e.g. interested area
- Automatic removal of outliers/scattered points or noise from the point cloud
- Fusion/merging of multiple point clouds
- Point cloud decimation / reduction of density
- Segmentation/separation of point clouds

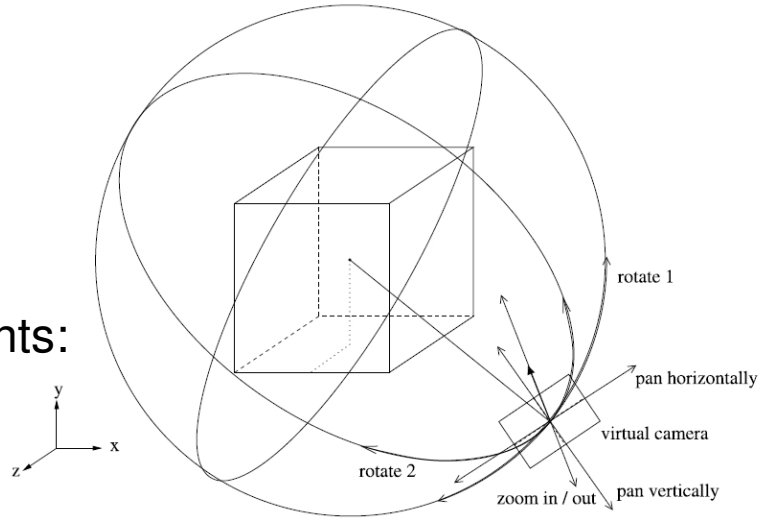
Processing for final result (no need to 3D meshing):

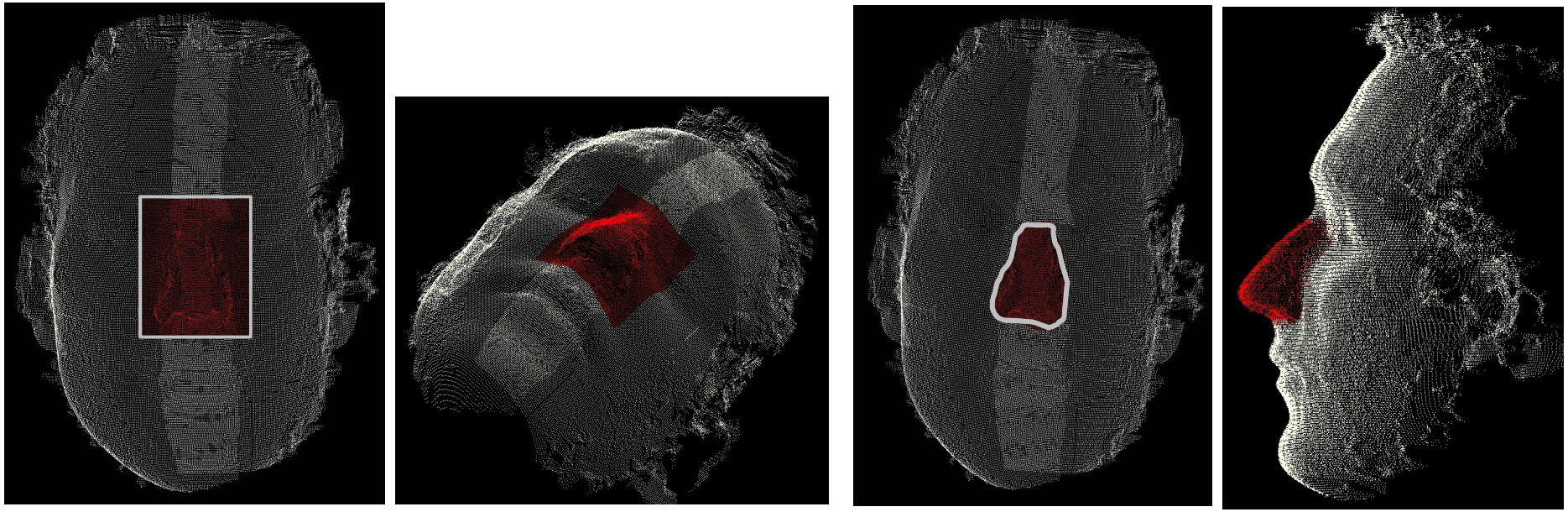
- Geometric features extraction / Fitting of geometric elements
- Extraction of 3D measures

Basic point could editing and processing 1

Manual selection of points in the 3D point cloud:

- Rectangle or contour selection
- 2D view=virtual camera, 3D points backprojected
- Problem: selection of points in 2D view -> 3D points:
  - not easy to fast select all the wished points
  - select & move/rotate/zoom

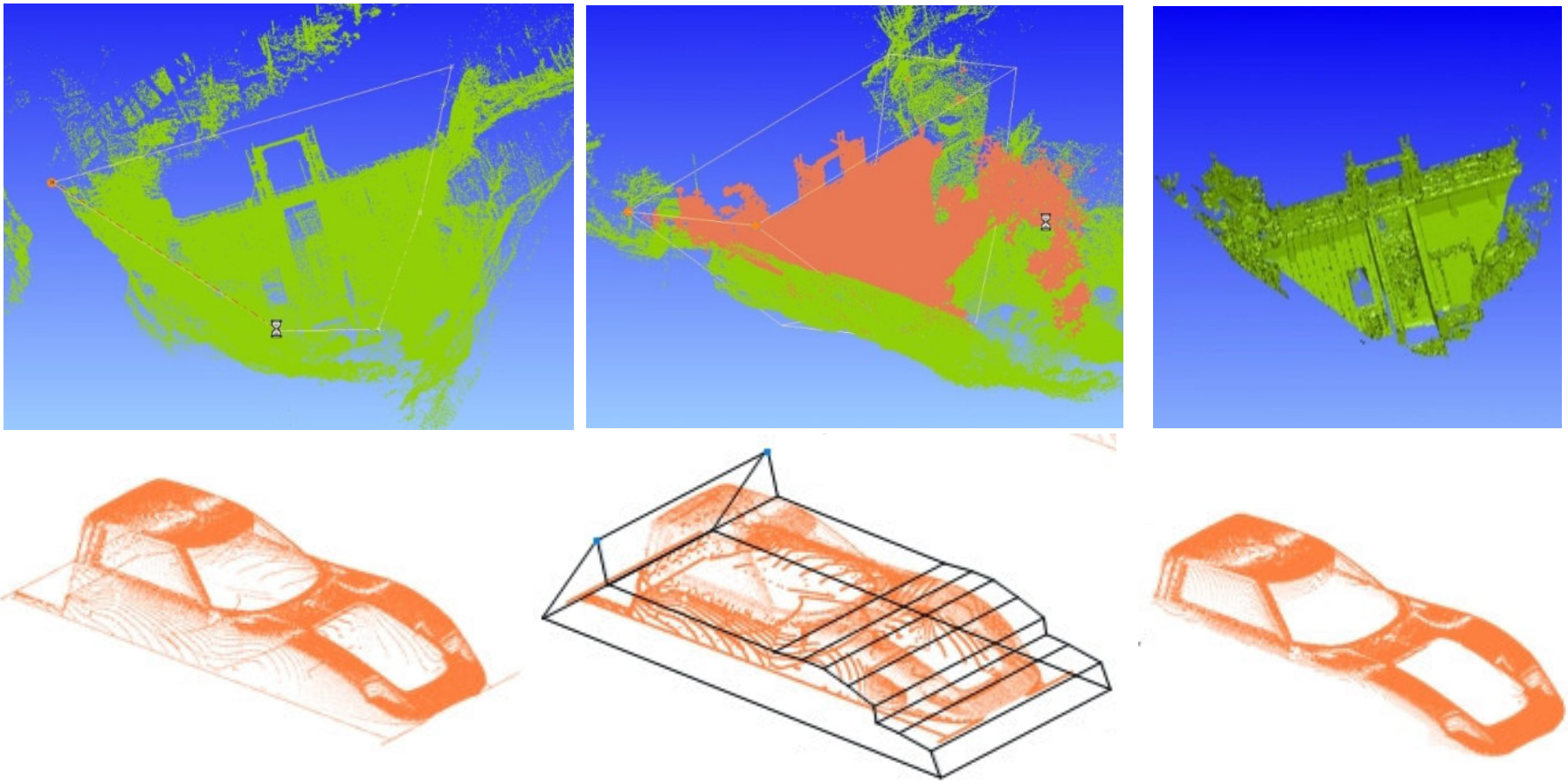




Basic point could editing and processing 1

Volumetric selection method:

- Select contour in 2D
- Extend contour in one direction: selection of more complex volumes



Basic point could editing and processing 2

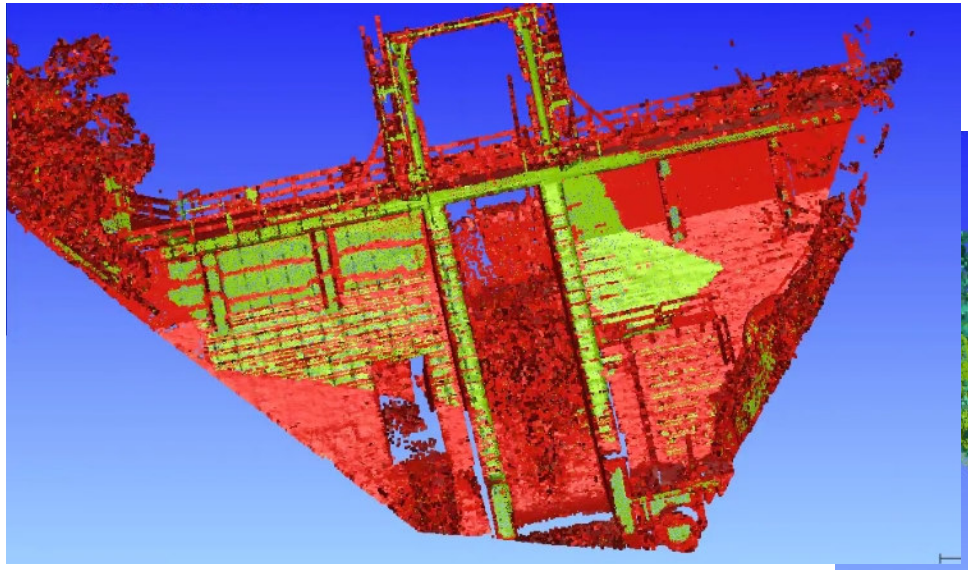
Automatic removal of outliers/scattered points or noise from the point cloud:

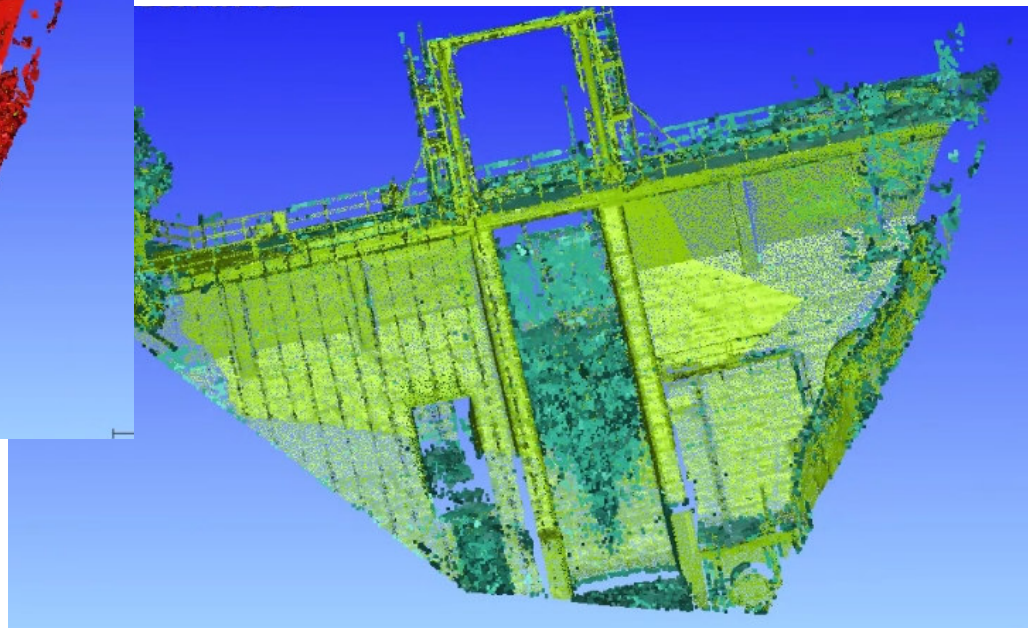
- Distance threshold:

  distance between adjacent points > threshold -> remove point

- Simple method: because, unorganized point cloud

  -> no information available yet about surface!

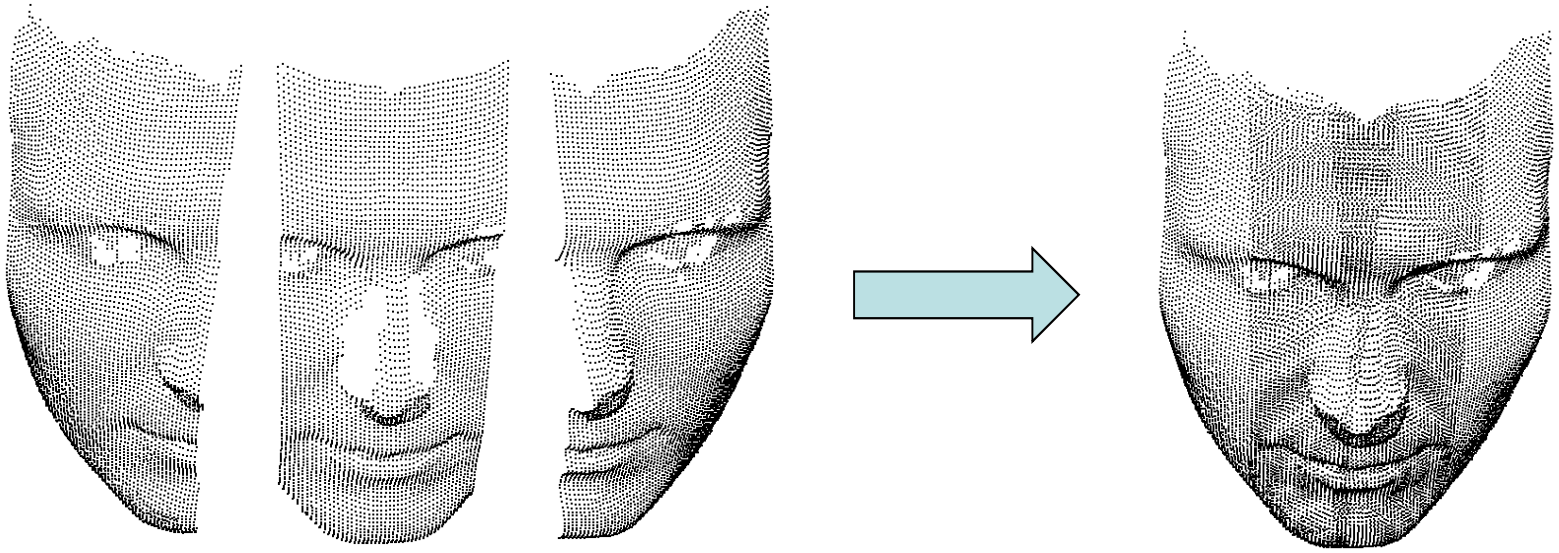




Basic point could editing and processing 3

Fusion/merging of multiple clouds:

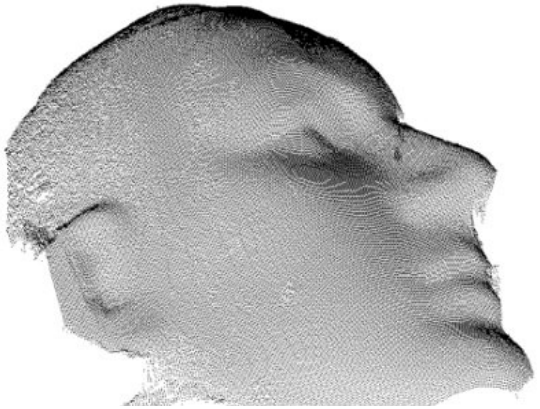
- By photogrammetric pipeline: multiple point clouds already “aligned”
  -> merging/fusion by simply adding list of points:



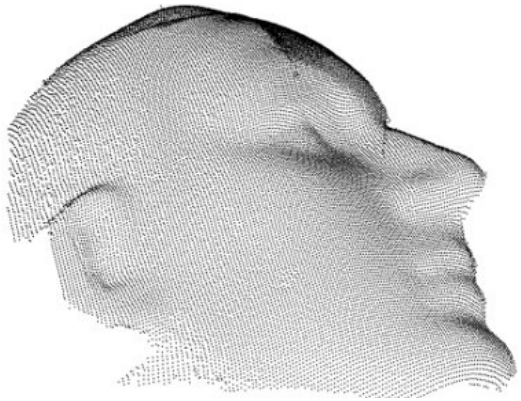
Basic point could editing and processing 4

Decimating point clouds:

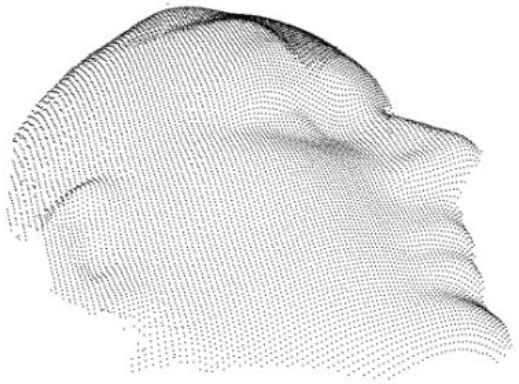
- Reduce point cloud density - regular reduction
- Problem: details also reduced -> better decimate after 3D meshing process, see later



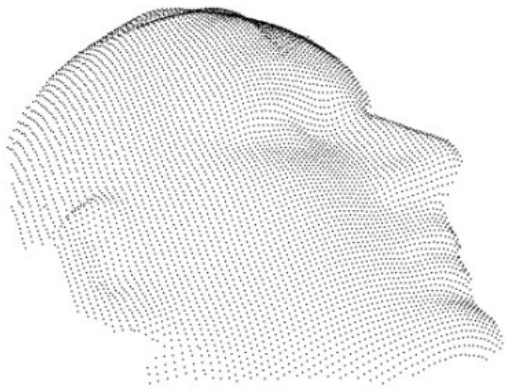
100,000 points



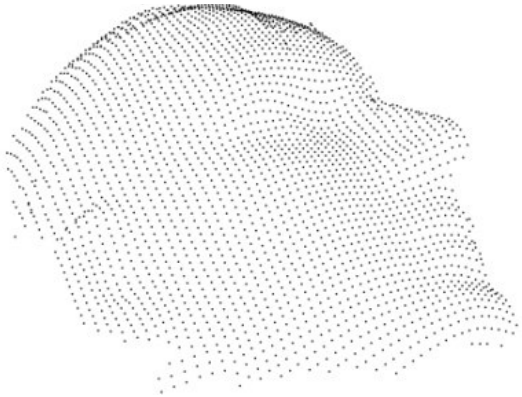
25,000 points



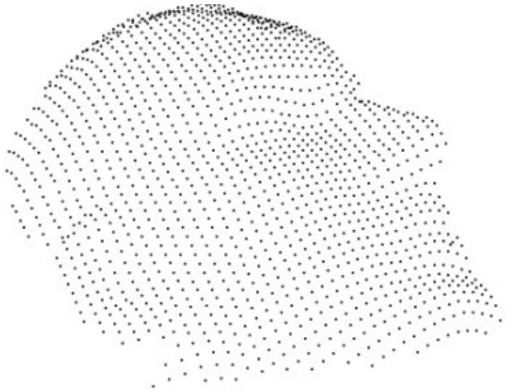
10,000 points



5,000 points



2,500 points



1,500 points

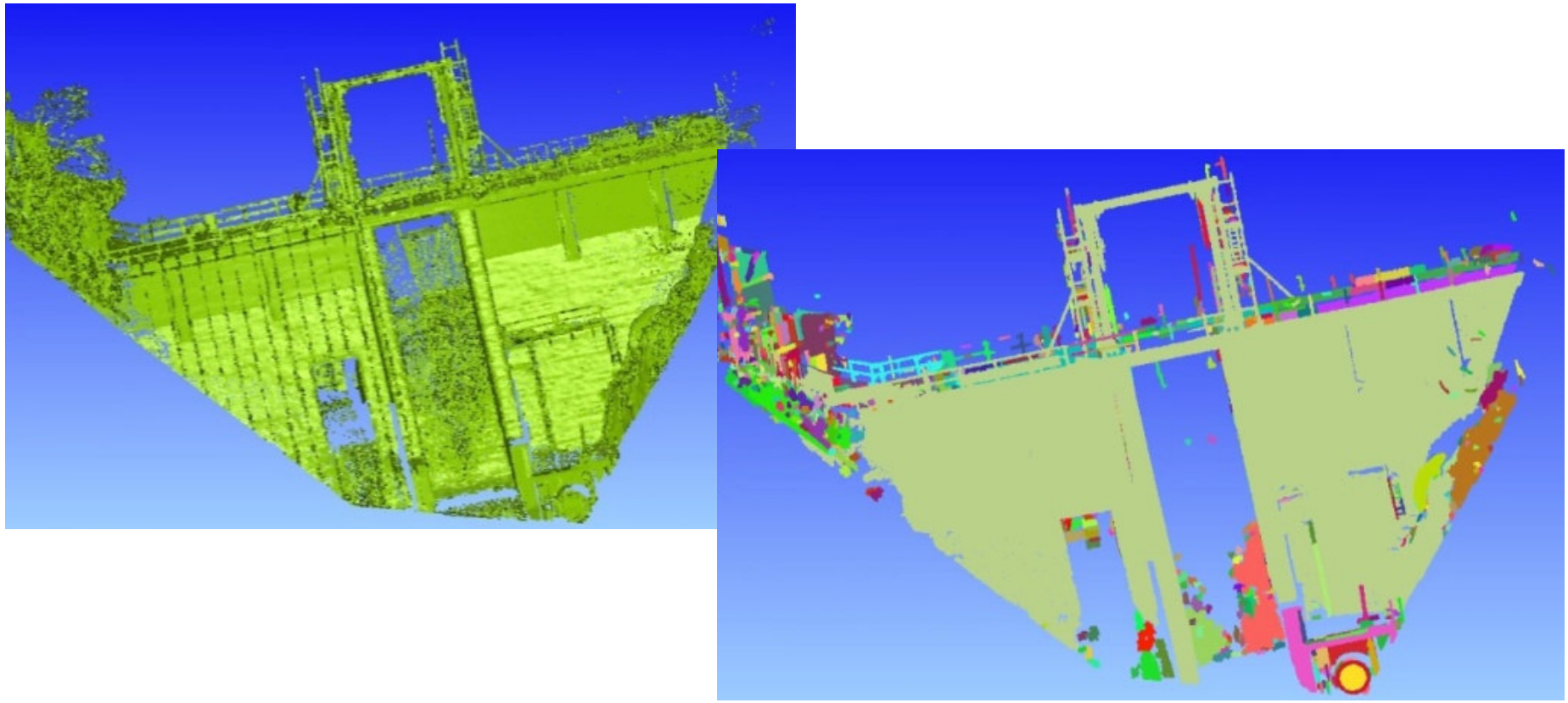
Basic point could editing and processing 5

Simple segmentation: “point cloud explosion”

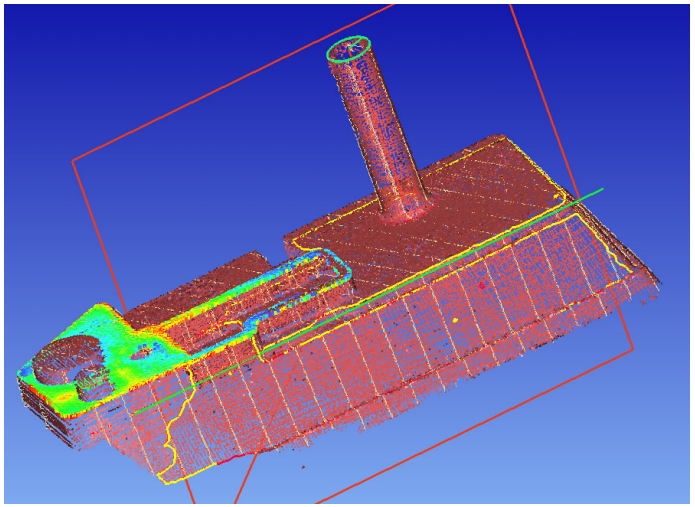
- Segment point cloud into clusters by two thresholds:

  Minimum distance between points – maximum distance between points

- Delete small clusters with less than *min* points



Point could processing for final result 1

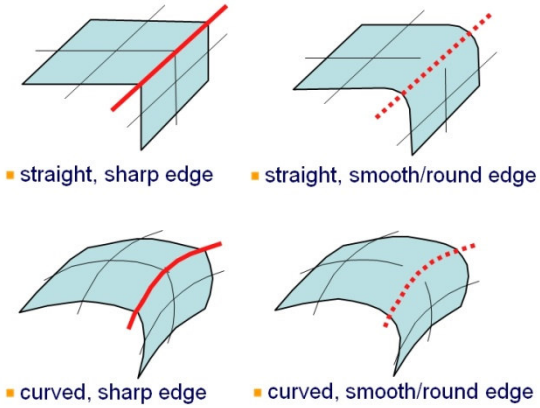


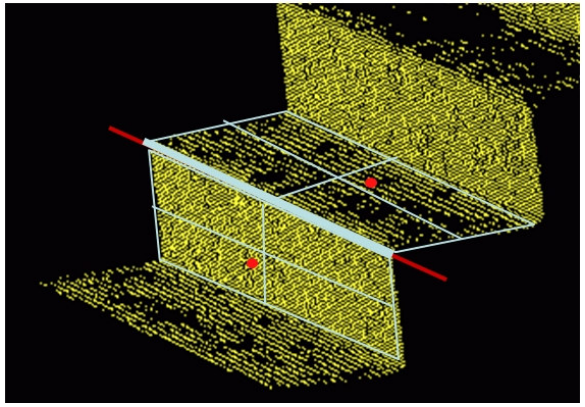
Geometric feature extraction:

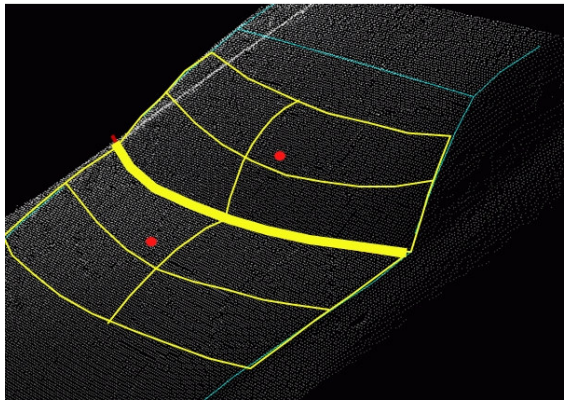
- Best shape extraction
- Typical processes based on local curvature analysis

Fitting of geometric elements:

- Least squares fitting of geometric elements to the point cloud: minimize distance local 3D point cloud to geometric element (plane, curves)
- Example: edge determination by plane/curve fitting:







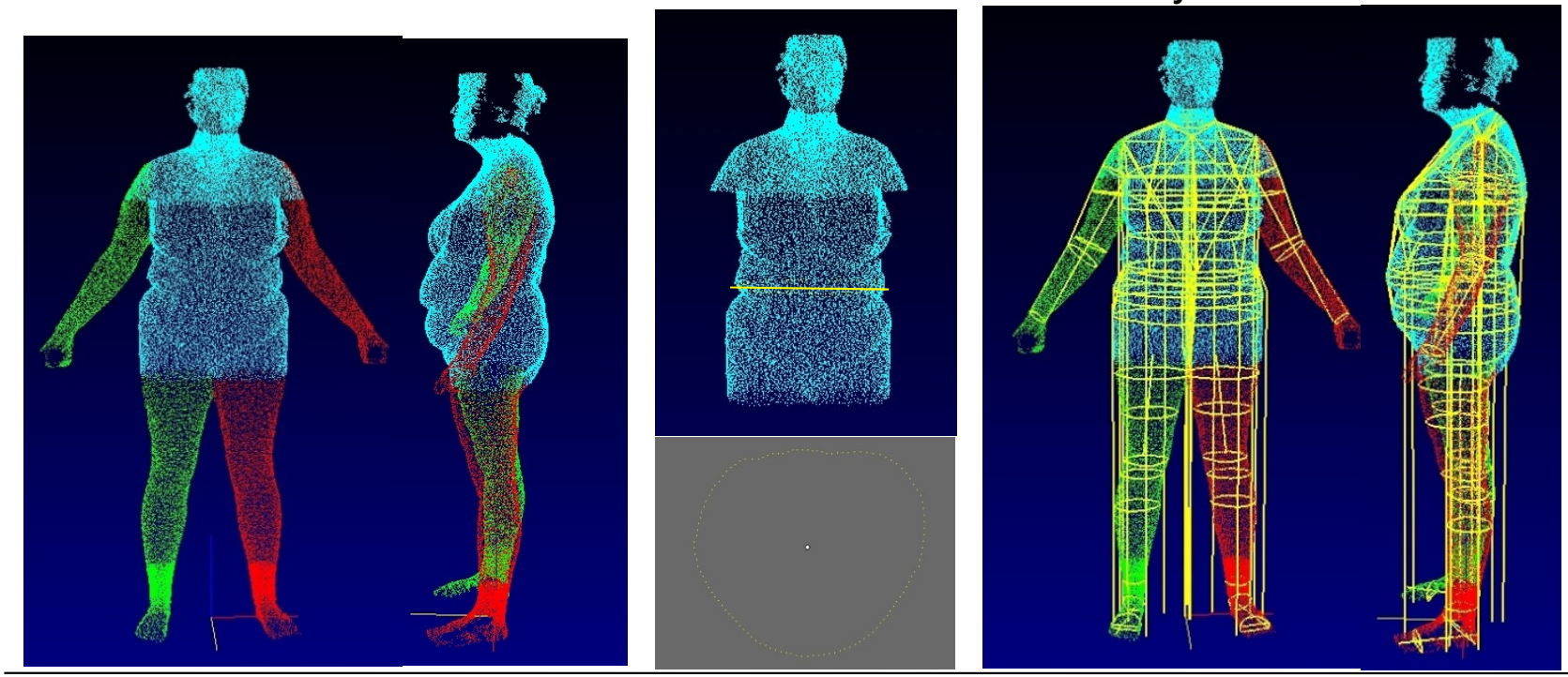
Point could processing for final result 2

Not always required to process the data for 3D mesh generation!

-> perform the required processed on point cloud only

Example: full body scan point cloud

- Segmentation of scan data into human body parts (arms, legs, “body”)
- Measurement of sections (e.g. measurement of waist on “body”)
- Automatic landmark extraction and measurement of body sizes



Point cloud processing, surface generation, texturing

1. 3D point cloud
   (a) General introduction
   (b) Photogrammetric pipeline - forward ray intersection
   (c) Types of 3D point clouds (volume, surface, edges, static, dynamic)

2. Point cloud processing
   (a) Raw point cloud (background, outliers, noise)
   (b) Point could editing (manual, automatic, removal points)
   (c) Point cloud processing (fusion, decimating, etc.)

3. Surface generation and texturing
   (a) 3D meshing
   (b) Surface editing (hole filling, smoothing)
   (c) Texturing methods (projection, mesh texture, etc.)

Photogrammetric measurement pipeline

ACQUISITION OF IMAGES

CALIBRATION / ORIENTATION

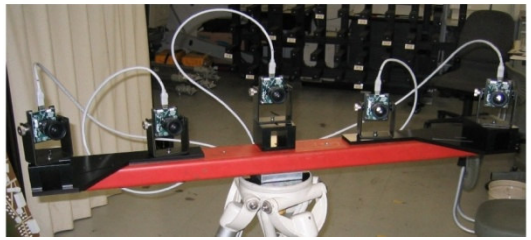
MEASUREMENT / MATCHING IN IMAGES

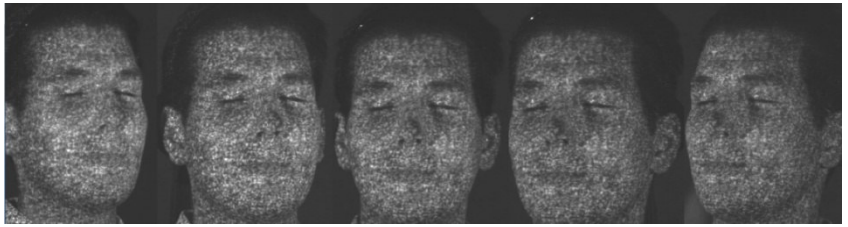
3D POINT CLOUD

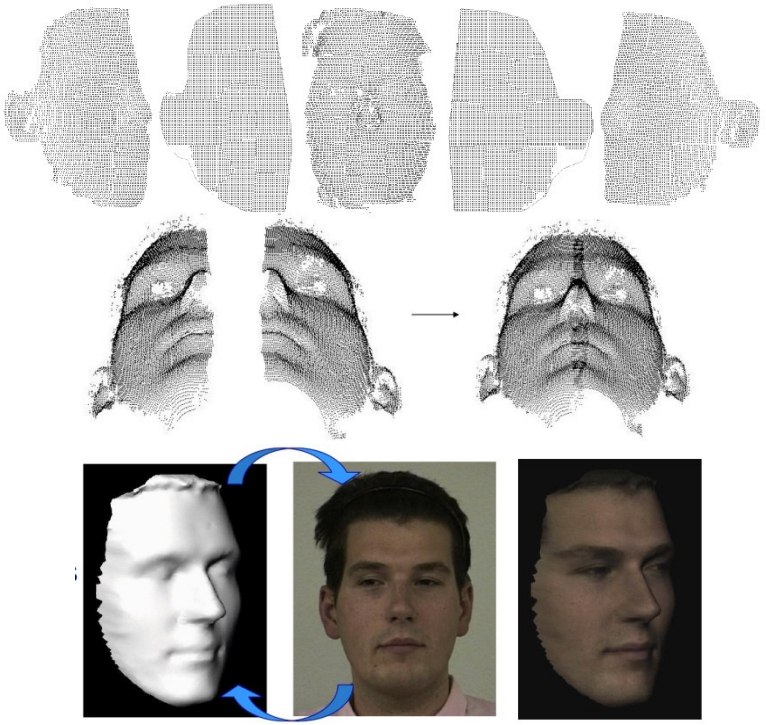
POINT CLOUD PROCESSING

SURFACE GENERATION / EDITING

TEXTURE MAPPING







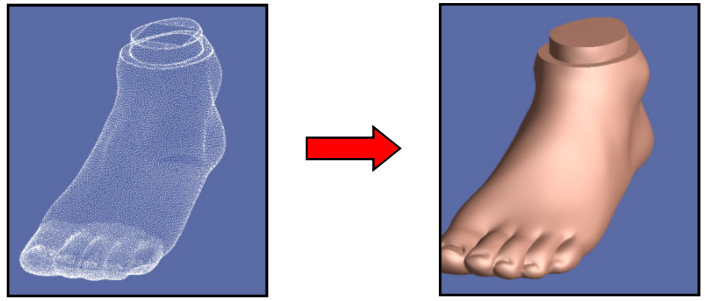
Surface generation

1. Mesh generation
   (a) General introduction
   (b) 2D/3D meshing
   (c) Adaptive meshing strategies

2. Surface editing and modeling
   (a) Data correction and completions (spikes removal, hole fillings, interpolation)
   (b) Data amelioration (smoothing, data reduction)
   (c) Data structuring (mesh structure, NURBS, parameterization)

3. Surface texturing
   (a) Texture mapping (point texturing, 2D texturing, 3D texture)
   (b) Photorealistic texture mapping (visibility analysis, radiometric correction)

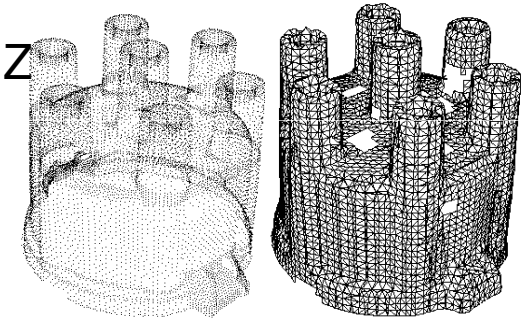
Mesh generation - from point cloud to surface



Meshing = polygonal surface generation:
from unorganized 3D point cloud
to set of polygons wich describe
the surface of the measured object

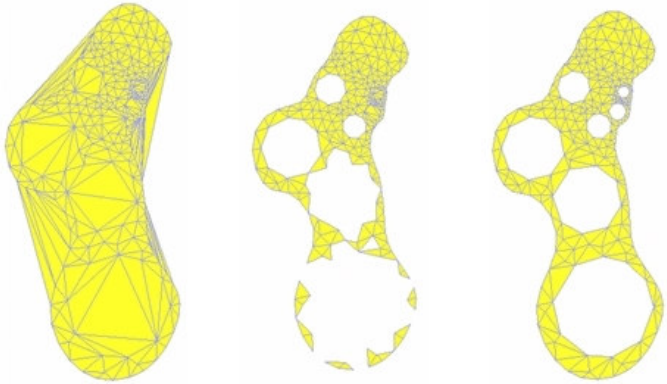
Point cloud vs. Surface model:

- Point cloud contains only surface digitization information: X,Y,Z
- Surface model contains:
  - geometrical information: vertices positions, surface normals
  - topological information: mesh connectivity, faces relations



Issues, problems:

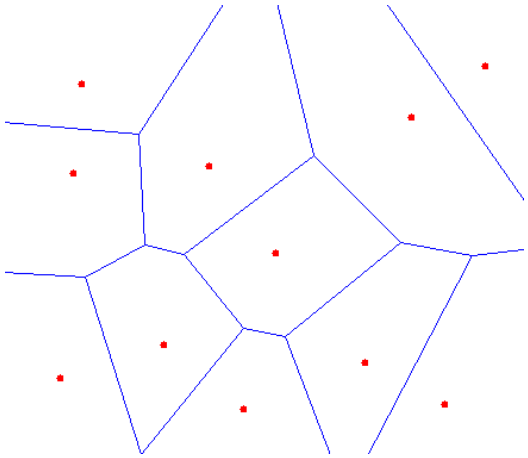
- Noise and measurement errors in the point cloud
- Unknown topology of the surface
- Number of polygons (computation problems)



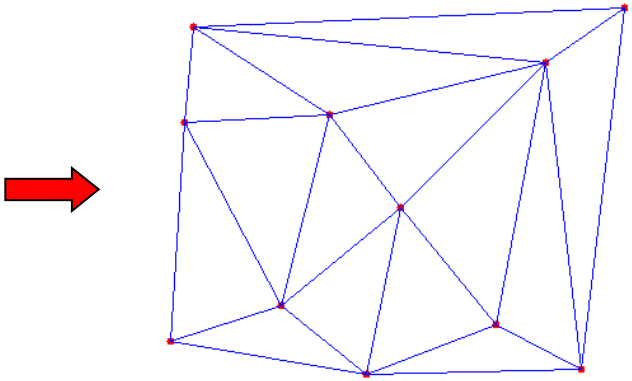
Mesh generation algorithm

Mostly employed method - Delauney triangulation

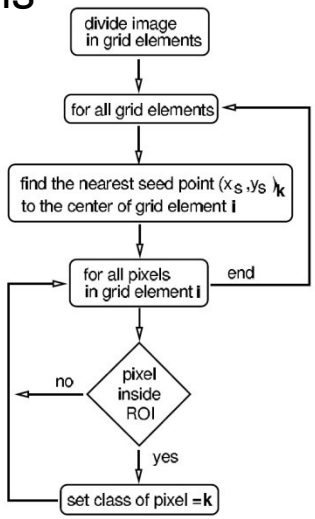
- It is a 2.5D method (extendible also to full 3D)
- Consider a set of points P on a plane (projection of 3D point onto a plane)
- Divide the space between individual points into polygonal regions such that the boundaries surrounding each point enclose an area that is closer to that point than to any other neighboring point (-> Voronoi tesselation)
- Build triangles by connecting the points of neighboring adjacent tessels



Voronoi tesselation



Delauney triangulation

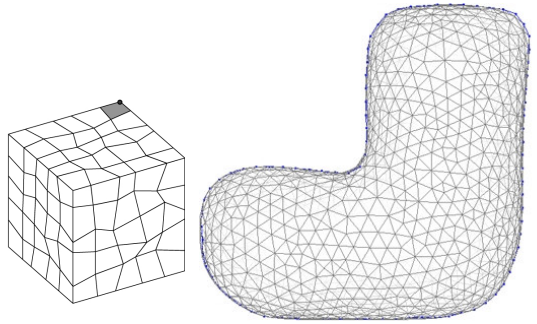


Flowchart of implemented Voronoi tesselation

Mesh generation strategies

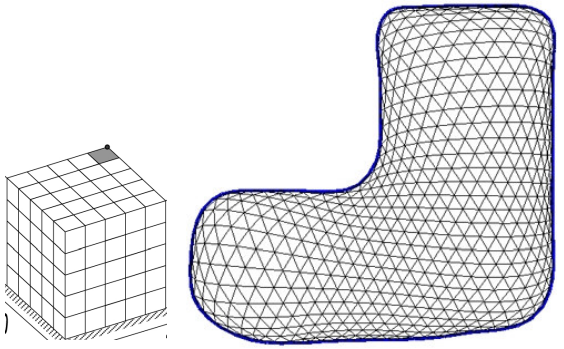
Original point cloud:

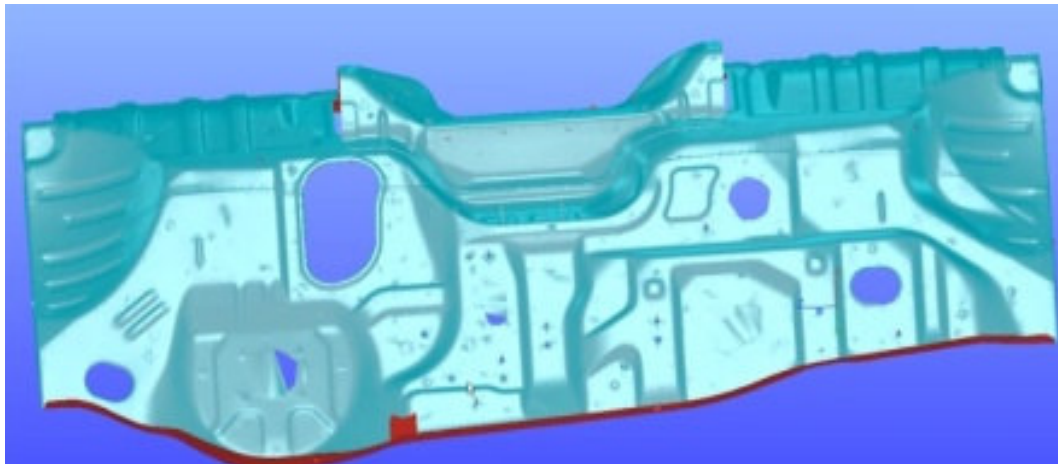
- A mesh is generated by triangulating the original point cloud
- Keep all points – no noise reduction – fast calculation



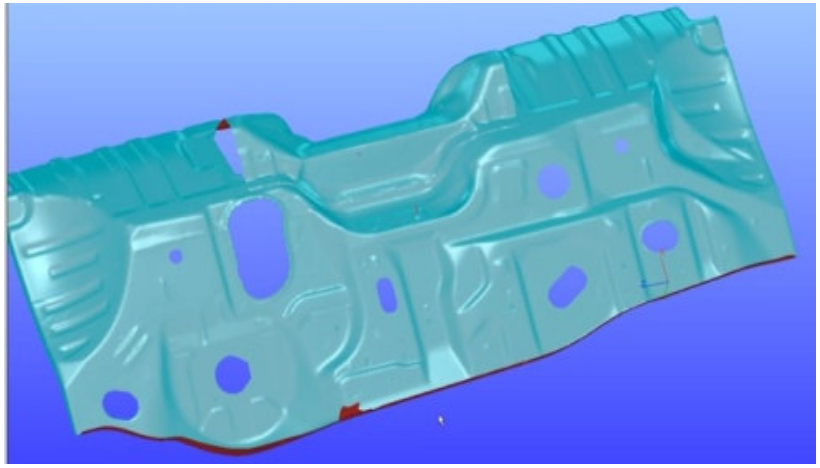
Regular resampling:

- A regular mesh is generated and points are resampled
- Noise reduction is possible by elmininating the points with deviation errors larger than a threshold
- Deviation error: distance original point to resulting 3D mesh





Original point cloud

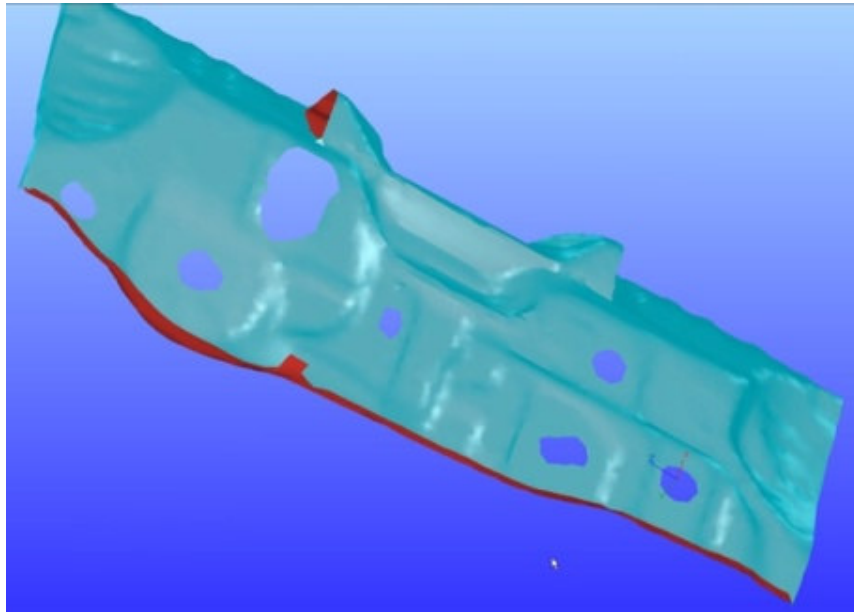


Regular resampling with noise reduction

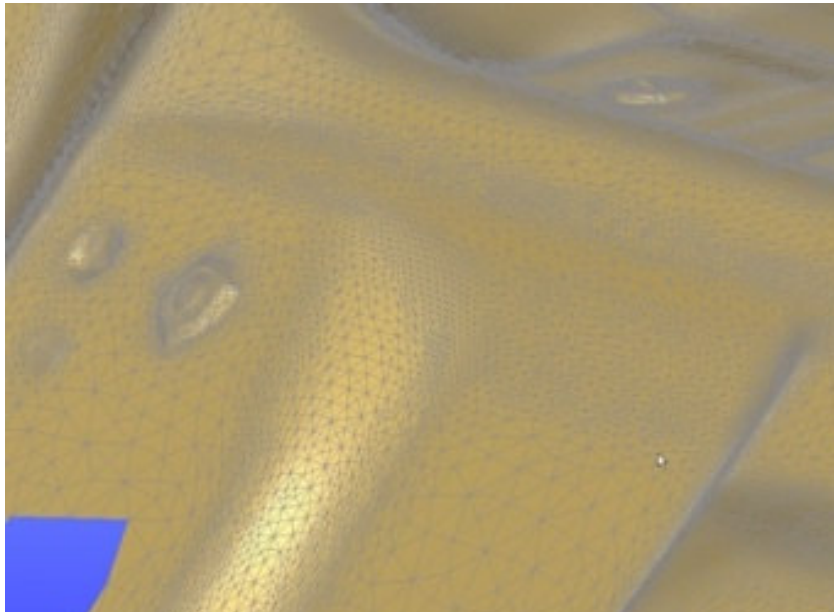
Mesh generation strategies

Meshing in 2 steps:

1. A rough mesh with regular triangles is created
   - the larger size of triangles, the more effective is the noise reduction
     (but the level of details will be lower)
2. A refinement meshing will be perfomed with variable triangle size:
   - deviation error criterion: error between point cloud and reconstructued mesh
   - fine details are preserved by more dense triangulation



Rough mesh

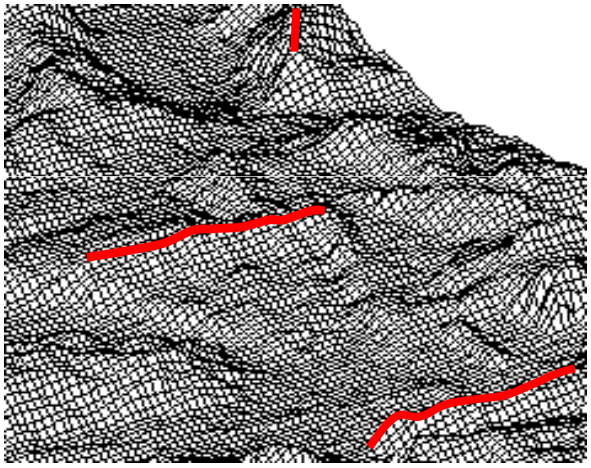


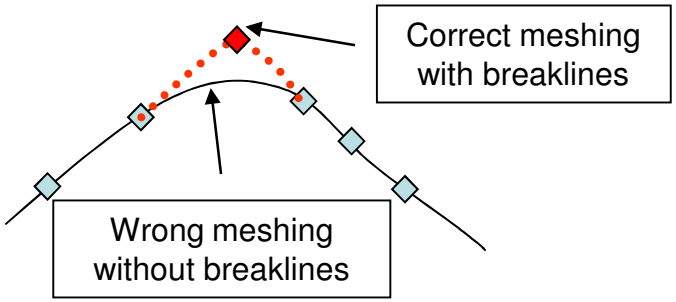
Detail of refined mesh

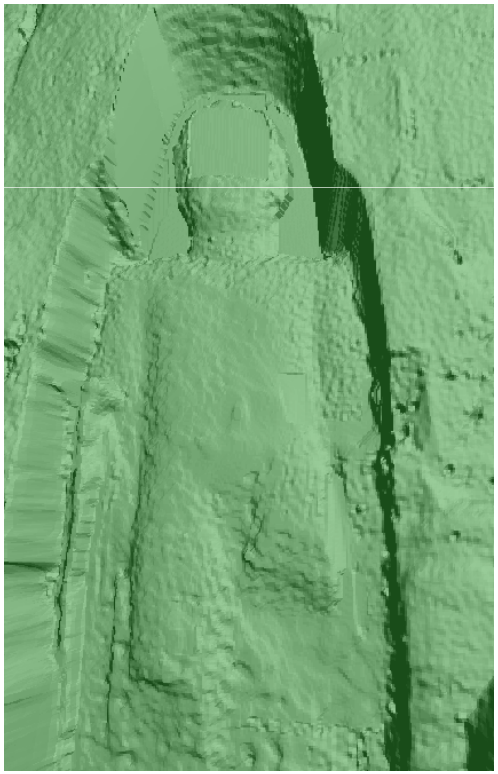
Mesh generation strategies

Constrained meshing:

- Include constrains in the meshing
- Force the triangles to be formed at specific locations
- Breaklines: surface topological information (curves, edges, ridges, etc.)
  -> specific details are kept (no smooth effect caused by resampling)







Without breaklines

With breaklines

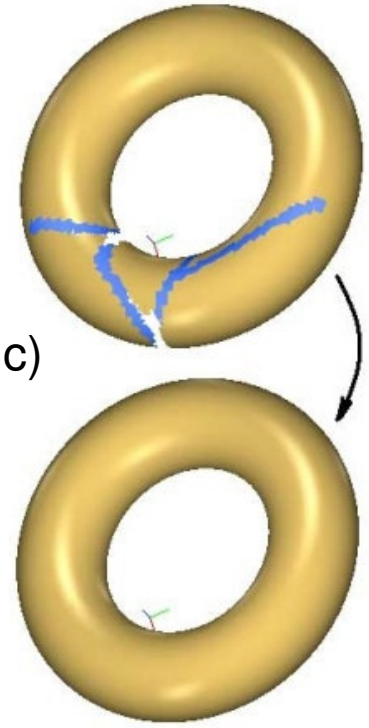
Surface generation

1. Mesh generation
   (a) General introduction
   (b) 2D/3D meshing
   (c) Adaptive meshing strategies

2. Surface editing and modeling
   (a) Data correction and completions (spikes removal, hole fillings, interpolation)
   (b) Data amelioration (smoothing, data reduction)
   (c) Data structuring (mesh structure, NURBS, parameterization)

3. Surface texturing
   (a) Texture mapping (point texturing, 2D texturing, 3D texture)
   (b) Photorealistic texture mapping (visibility analysis, radiometric correction)

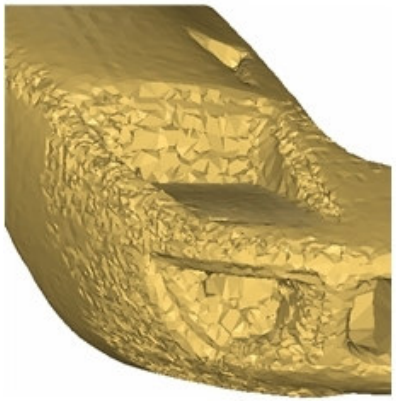
Data correction and completions



Two main tasks:

1. Hole filling
   - automatic procedures works well for small holes
   - interpolation of points based on local surfaces (usually quadratic)
   - large holes: manual editing usually required

2. Spikes removal (surface straighten)
   - automatic procedures perform accurate results only locally
   - manual correction achieve better results



Data amelioration

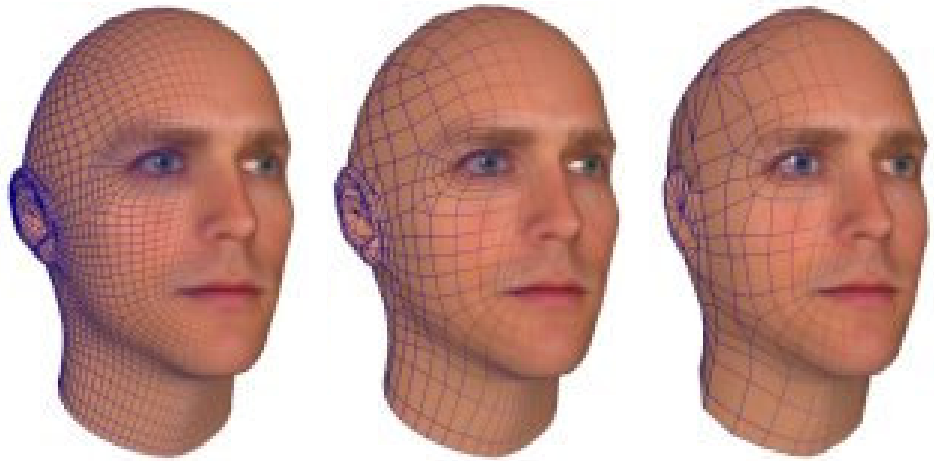
Smoothing

- require: pre-processing (hole filling, spikes removal)
- automatic procedures works very well
- deviation error threshold are considered
- small radii and sharp edges can be preserved

Before smoothing | After smoothing

Data reduction

- strong data reduction can also be achieved by similar algorithms
- Main goal: reduction of number of triangle, keep information of small details



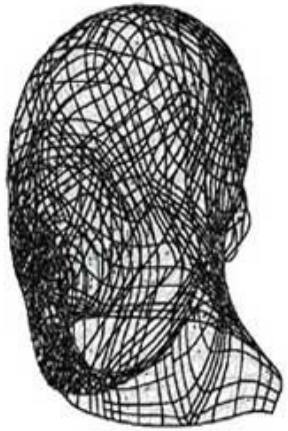
Data structuring

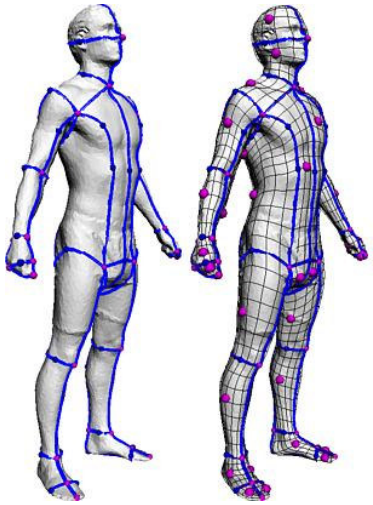
Task:

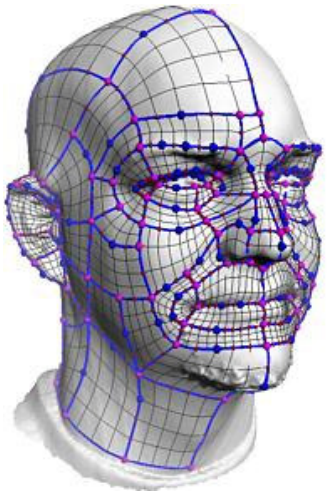
- Transform the 3D surface mesh into surface models with structure
  (e.g. NURBS Non-Uniform Rational B-Splines)


Point cloud


3D mesh


NURBS





Advantages of parameterized/structured surface model:

- Size of entire data (from several MB to few KB)
  List of polygons -> list of parameters
- Coordinate system independent
- “Matching” between different data is possible

Surface generation

1. Mesh generation
   (a) General introduction
   (b) 2D/3D meshing
   (c) Adaptive meshing strategies

2. Surface editing and modeling
   (a) Data correction and completions (spikes removal, hole fillings, interpolation)
   (b) Data amelioration (smoothing, data reduction)
   (c) Data structuring (mesh structure, NURBS, parameterization)

3. Surface texturing
   (a) Texture mapping (point texturing, 2D texturing, 3D texture)
   (b) Photorealistic texture mapping (visibility analysis, radiometric correction)

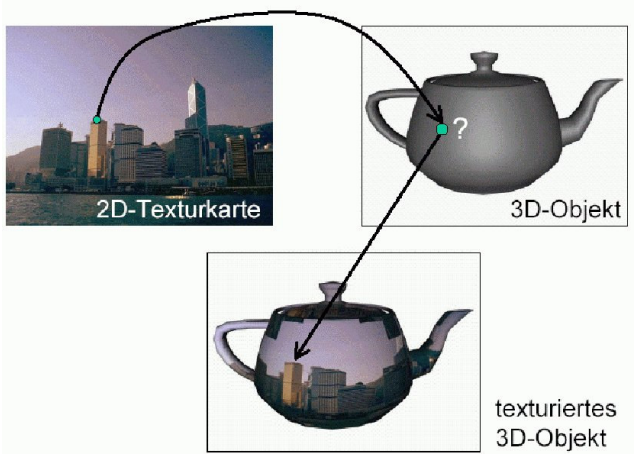
Surface texturing

Task:

Add color information to the 3D surface model

Usually real image data is used for texture mapping

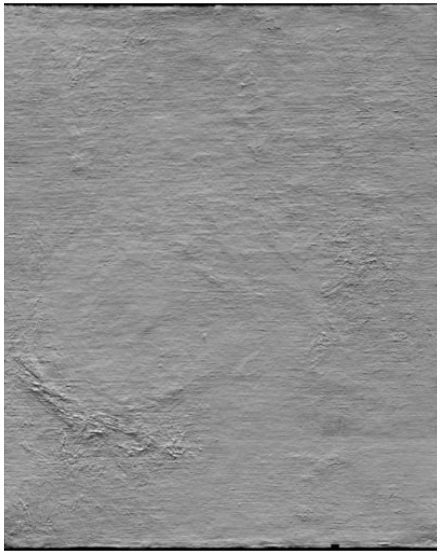
-> result: realistic representation of the object

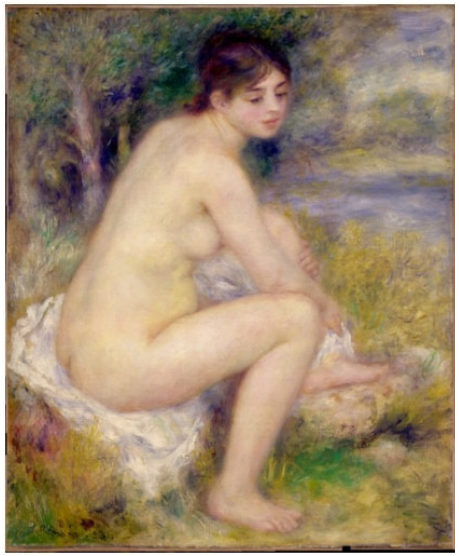


Texture mapping

Motivations/reasons for texture mapping

- Full representation of reconstructed object
- The information included in the texture may be more relevant than the geometrical information (=shape) of the 3D model
- The application of a texture can give to the model much more information, also “pseudo” geometrical information, e.g. the details and roughness of an object can be simulated without geometrical information, the shadows of the mapped texture will give the appearance of “3D”



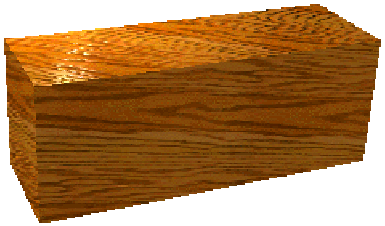


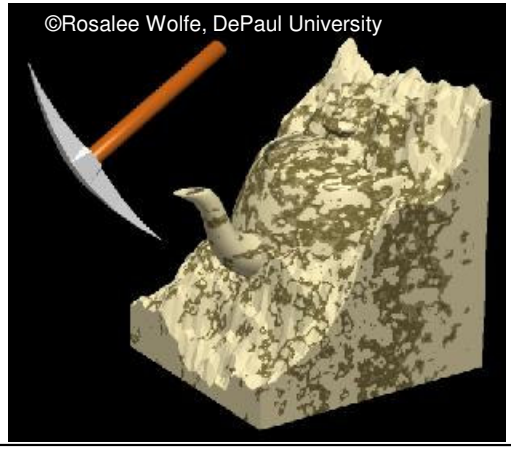
Different type of textures mapping methods

Point based texturing
- Coloration of point clouds

2D Texture
- Place a 2D image onto an object, like pasting wallpapers onto a wall

3D Texture
- Analog to carving the object from a block of marble
- Used for object with volumes-structure, e.g. wood

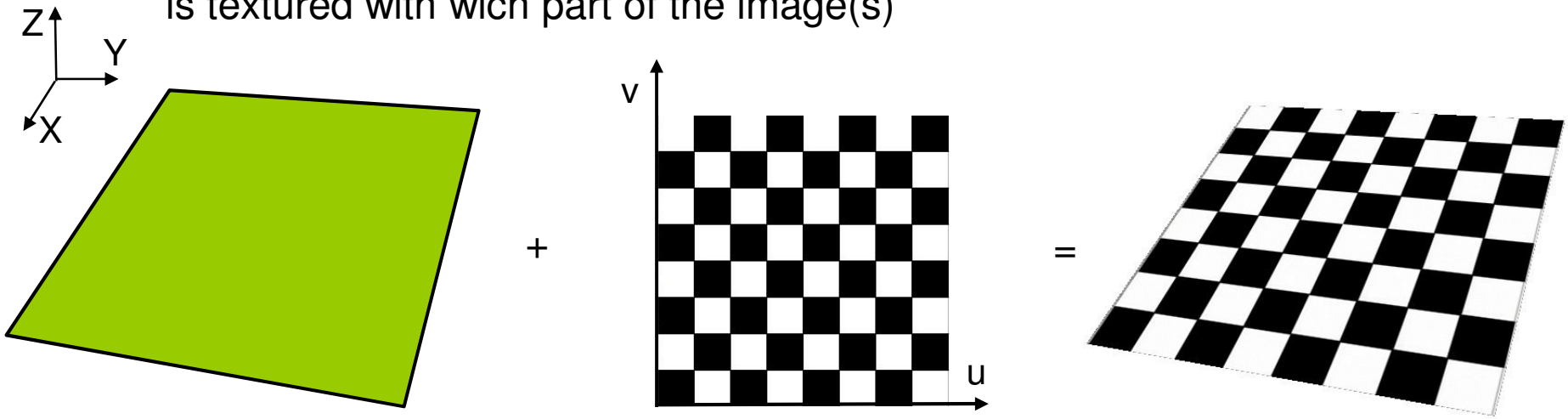




2D Texture mapping

Basic functionality of texture mapping:

- Given data:
  - Geometry: 3D surface model of the object
  - Texture data: 2D image(s)
  - “Connection between images and object”, e.g. orientation of the images
- Goal:
  - mapp 2D texture onto 3D geometry,
    i.e. determine which part of the object (area, triangle, vertex)
    is textured with wich part of the image(s)

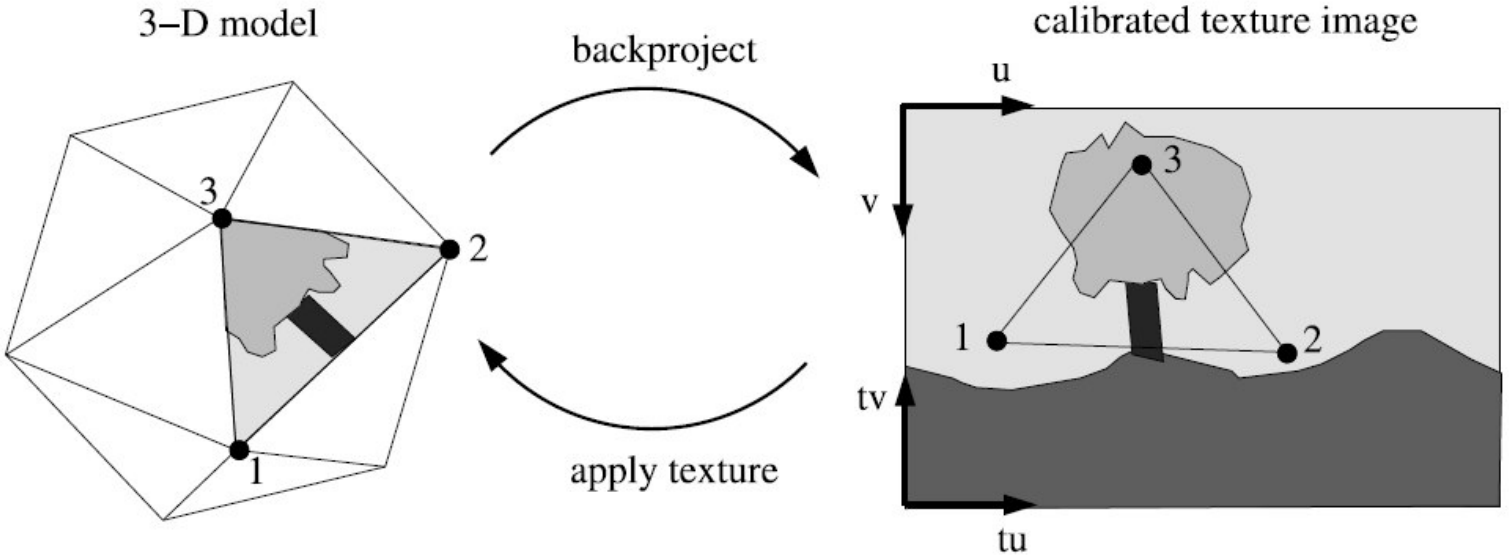


Geometry | 2D Texture | Textured Model

2D Texture mapping

Texture mapping usually consists of two transformation:

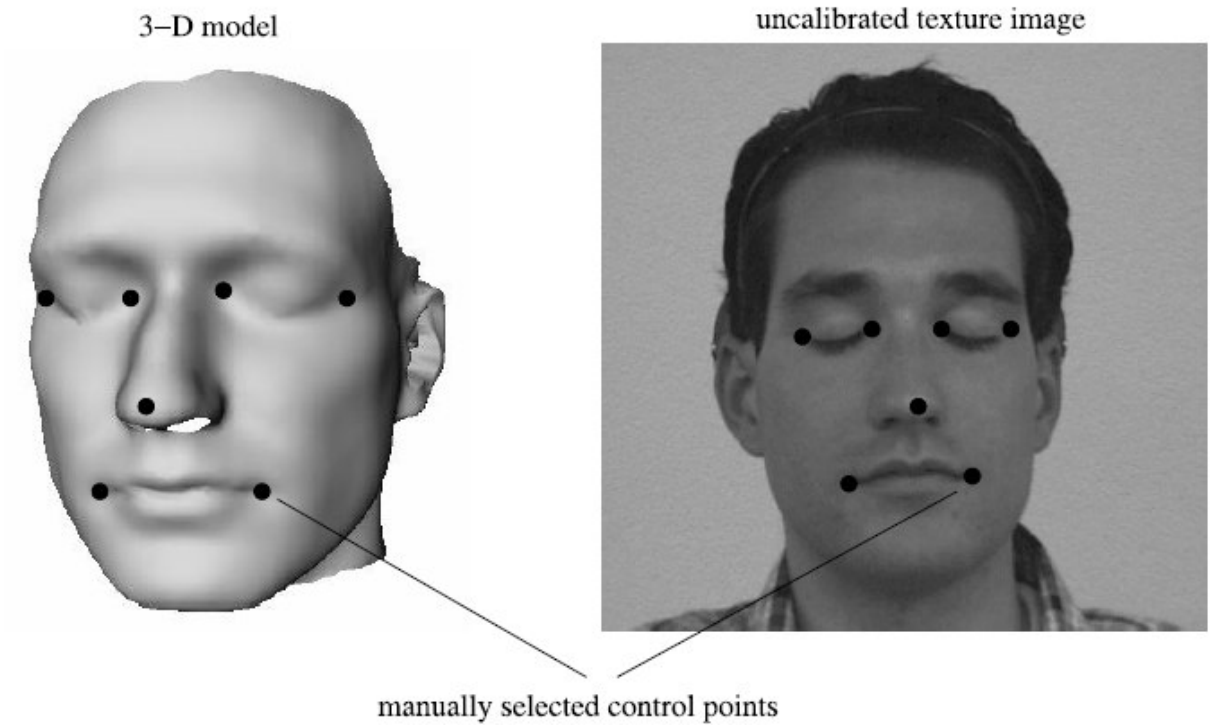
- Backprojection of each verteces of the 3D model onto the corrispondent (calibrated) texture image
- Application of the single “fractions” of texture to the triangles of the 3D model
- This information is usually stored in 3D files as “texture image coordinates”



2D Texture mapping

For uncalibrated image data:

- Select manually few corresponding points in the 3D model
  and in the uncalibrated texture image
- Calibrate the camera by “spatial resection” using the 3D points as control points



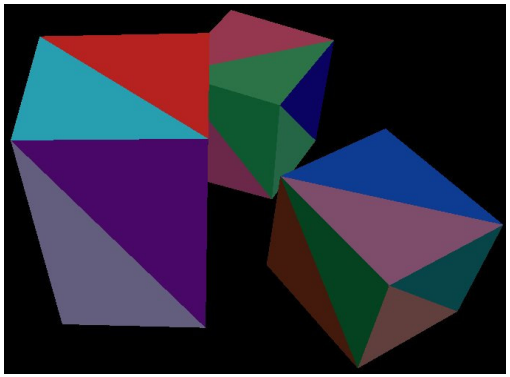
Photorealistic Texture Mapping

Additional problems to consider:

- Visibility analysis

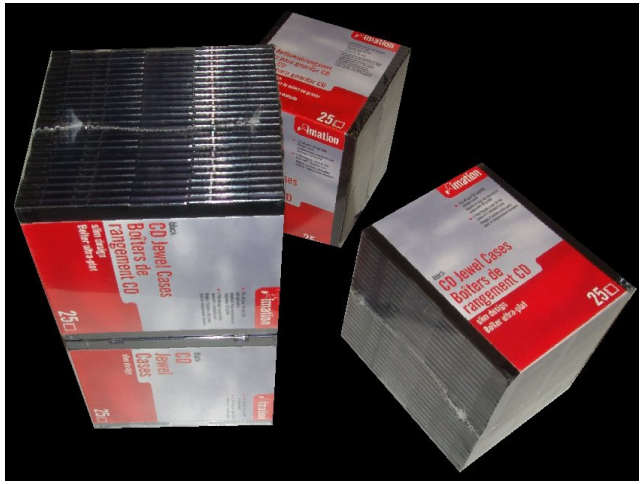
  - determination of the visible and invisible parts of objects in the texture image(s)

  -> to avoid mapping of false texture

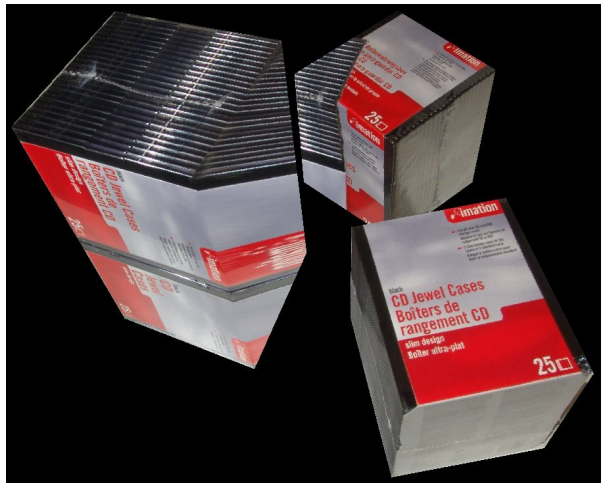


3D model

Texture image



Textured 3D model
viewed frontally
-> correct visualization



Textured 3D model
viewed from side
-> wrong texture is visible

Photorealistic Texture Mapping

After visibility analysis, complete texture images can be generated

For simple objects, a singular texture image is sufficient for satisfactory results

3D surface model

Cylindrical texture image

Texturized surface model

Photorealistic Texture Mapping

For more complex objects, multiple texture images are required

However, it is possible to group the different texture elements into composed images

Texture image for face

Texture image for full body

Photorealistic Texture Mapping

Additional problems to consider:

How to combine the information from different images?

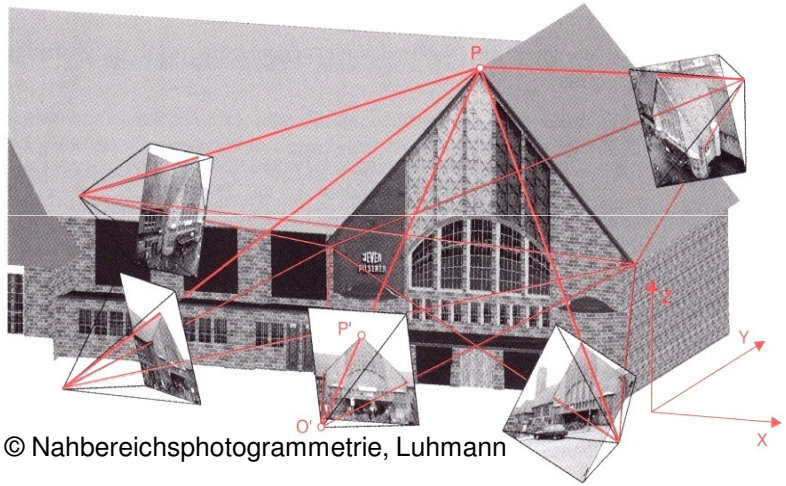
- differences in color and illumination, differences in resolution

Possible solutions:

- Calculation of an average texture value:

  the final color value for the object-texel is calculated by the average value of all the image grey values.

$$GreyValue = Average_{RGB} = \frac{\sum_{A=1}^{n} I_{A(RGB)}}{n}$$



© Nahbereichsphotogrammetrie, Luhmann

- Determination of the “best” texture for each object part:
  - the color value for each object space texel should be taken from only one image
  - Different parameters for the decision making,

    e.g. quality and resolution of the image-texture (angle&distance camera-object)

A pdf of this presentation is available to download at:

www.hometrica.ch/eth/PMV03.pdf

Visibility analysis

- **Using "Thresholding"**
  - Via thresholding the calculation of visible and invisible parts is possible using a high number of overlapping images

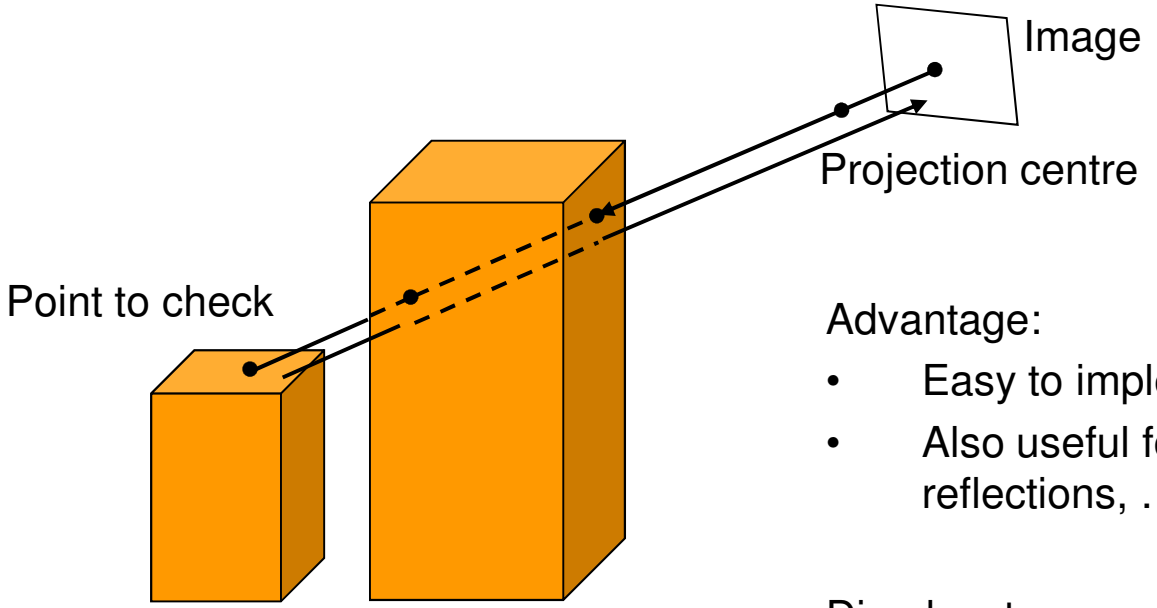
- **Ray-tracing**
  - Calculates the visibility of an object-point by tracing the ray from the image point through the projection centre to the object point or reverse

- **Z-Buffer**
  - Visibility analysis of an object-point by calculation of the distance from the image point to the object point

- **Surface-based**
  - Calculation of visibilities using vector-geometry

Visibility analysis – Ray-tracing:

- Ray-tracing:
  - Calculate the visibility of an object-point by tracing the ray from image point through the projection centre to the object point or reverse



Advantage:

- Easy to implement
- Also useful for rendering of shadows, reflections, …

Disadvantage:

- High amount of calculation time
- Calculation works point-wise

Visibility analysis – Z-Buffer:

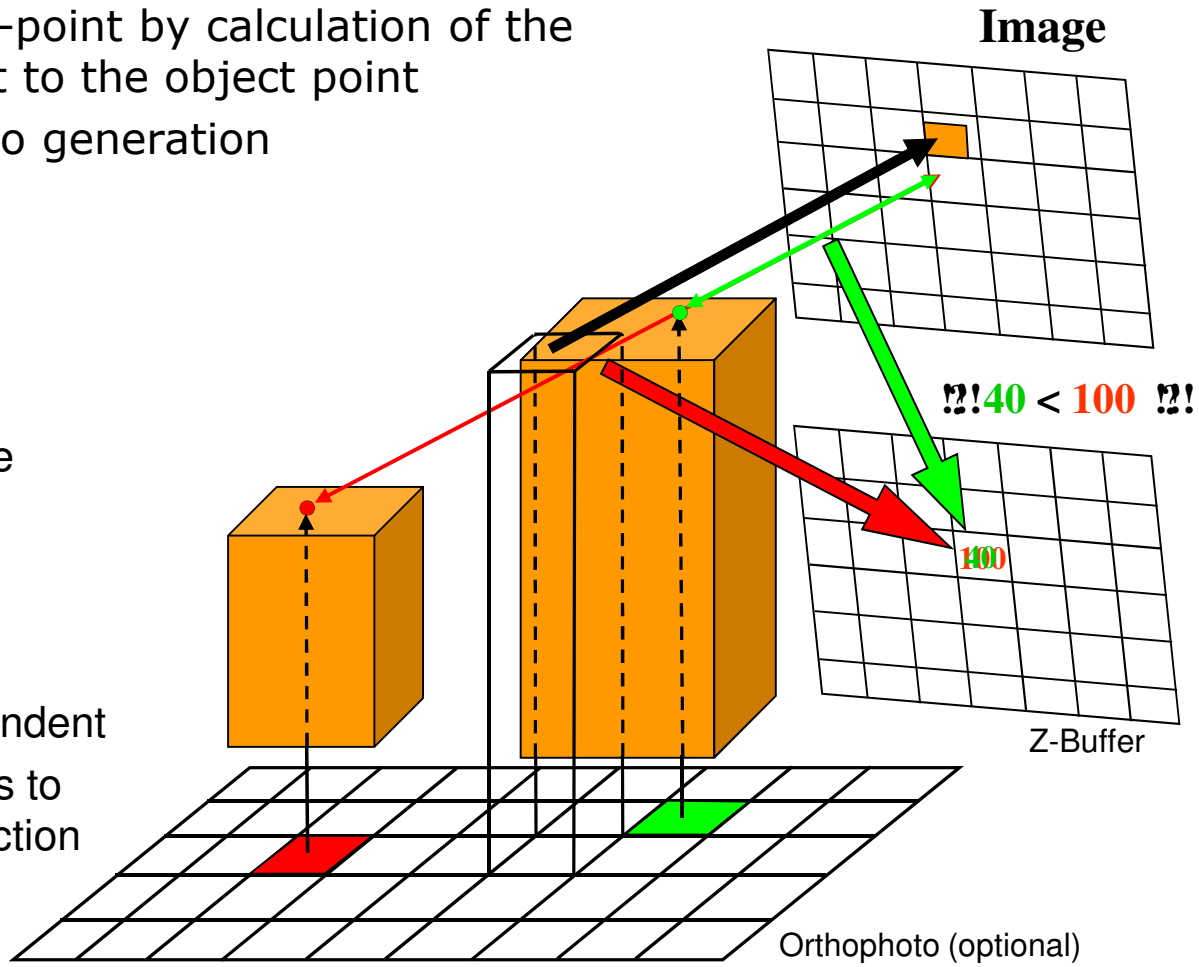
- Visibility analysis of an object-point by calculation of the distance from the image point to the object point
- Often used for true-orthophoto generation

Advantage:

- Easy to implement

Disadvantage:

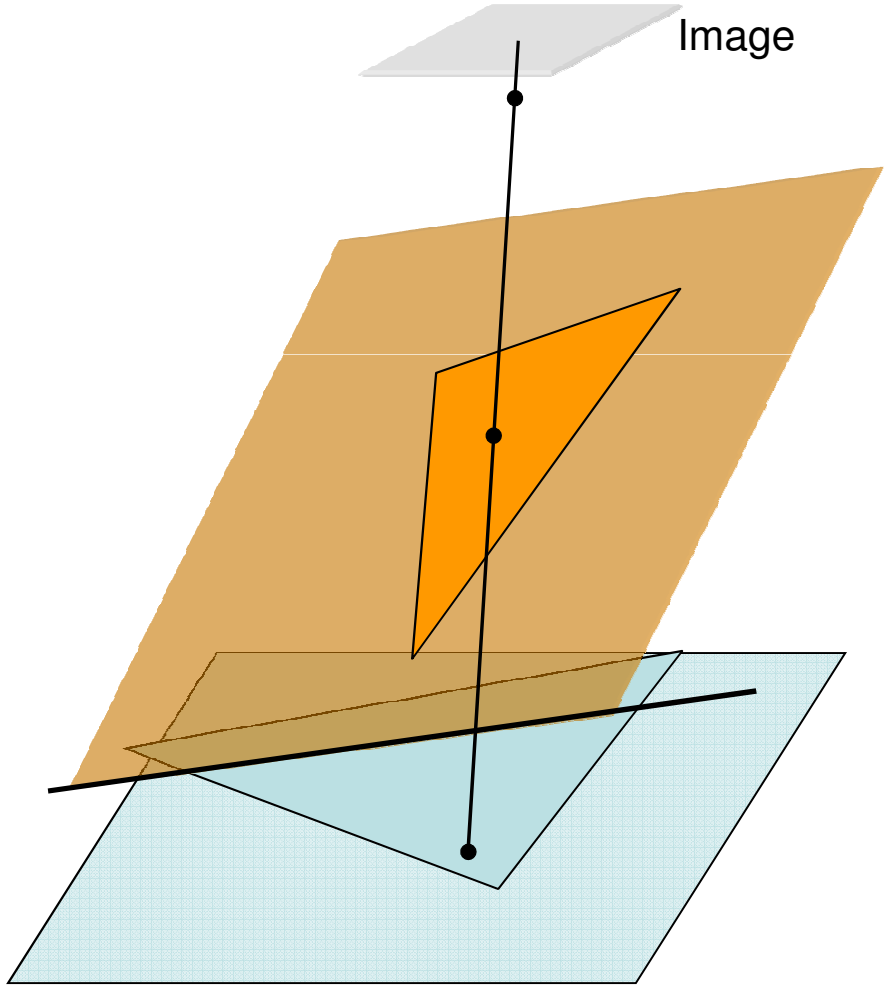
- High amount of calculation time
- High amount of memory
  - Aerial image: 23 x 23 cm^2 ; 15µm ; double precision (8 byte) → 1.8 GByte
- Pixel-based → resolution dependent
  - Mixed pixels on edges leads to aliasing and occlusion detection problems



Surface-based visibility analysis

Detection of the front triangle

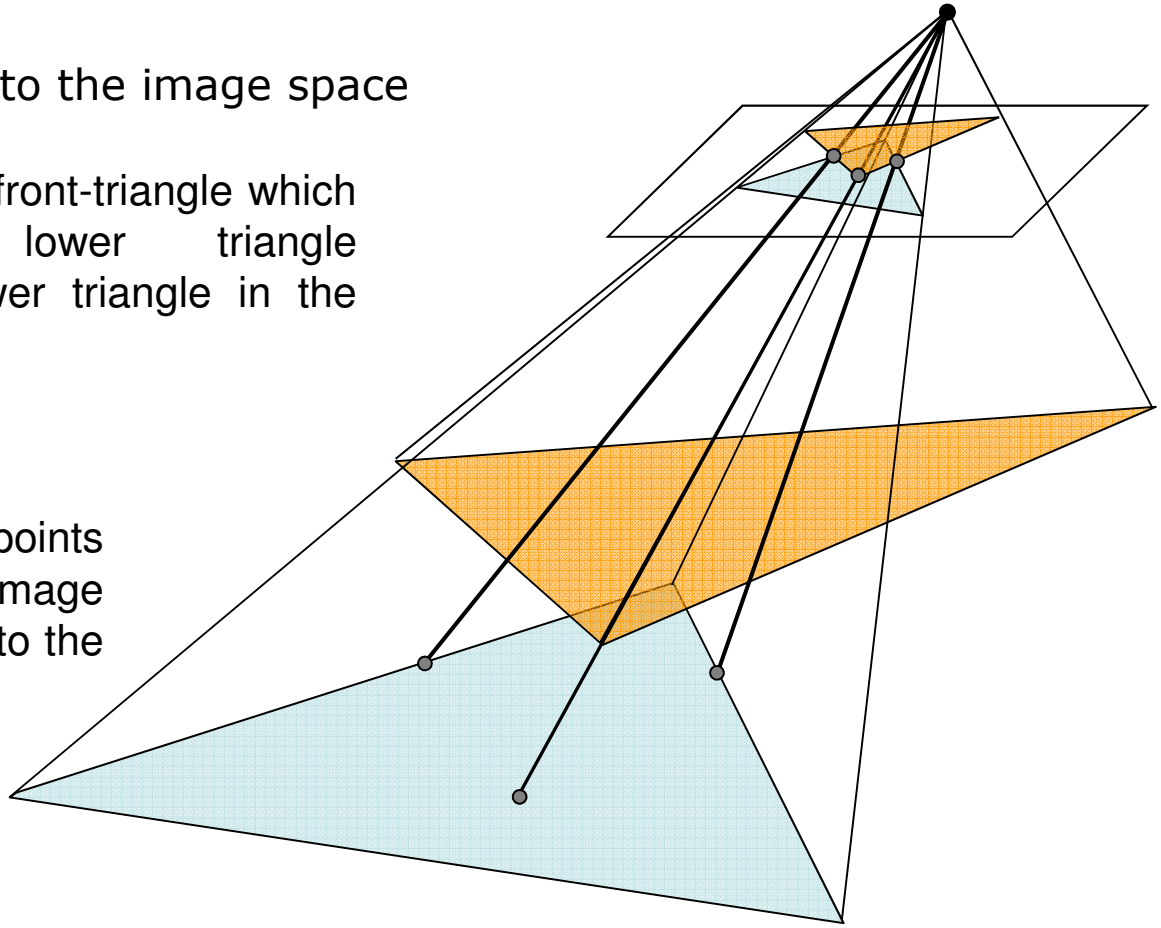
- Calculate the intersection line of the two planes defined by the triangles
- Calculate the triangle which is fully on one site of the intersection line and define a point in this triangle
- This point defines a intersection ray to calculate the corresponding point in the second (triangle) plane
- According to the distance from the projection centre to the respective point in the two planes, the front-triangle can be calculated



Surface-based visibility analysis

Points for re-triangulation:

- Projection of the triangles into the image space
- Projection of all vertices of the front-triangle which are inside of the lower triangle (in image space) into the lower triangle in the object space
- Calculation of the intersection points between the two triangles in image space and projection of them into the object space



Surface-based visibility analysis

Re-triangulation of the lower triangle:

- In a separate triangle-coordinate system
- Constrained Delaunay Triangulation

Visibility analysis:

- Definition of points in every new triangle
- Check: if the ray between projection centre and point intersects the upper triangle, the lower triangle is occluded