HALCON

the Power of Machine Vision

HALCON三维视觉方法

段德山
产品总监
中国大恒（集团）有限公司
北京图像视觉技术分公司

MVTEC

IMAVISION

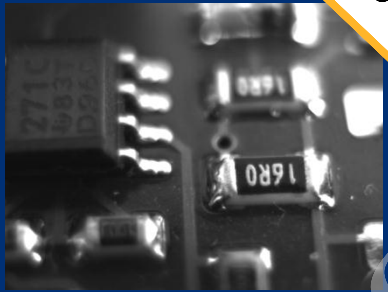
HALCON offers various methods for 3D machine vision:

- Camera calibration, Self-calibration
- Hand-eye calibration
- Rectification and 2D measurement
- Pose estimation
  - 3D matching (surface-based or shape-based)
  - Pose from rectangles or circles
- Reconstruction
  - Sheet-of-light methods
  - Binocular and multi-view stereo
  - Depth from focus
- 3D object processing
  - 3D surface comparison
  - 3D registration
  - Triangulation
  - 3D-primitive fitting

Select Your Solution: 3D Methods I

Height required?

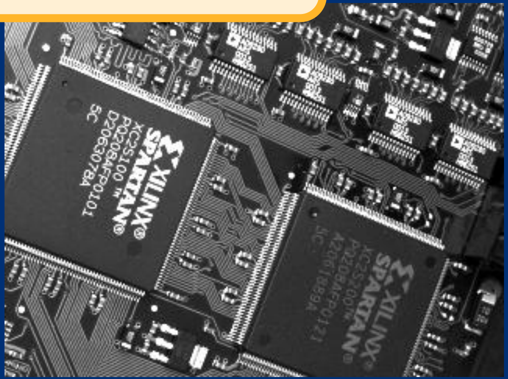
Yes



Large depth of field?

Yes

3D Methods II

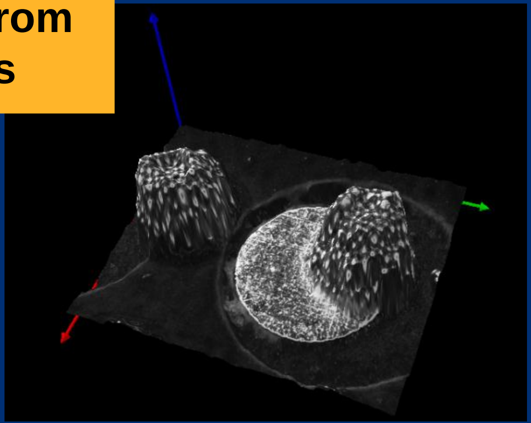


No

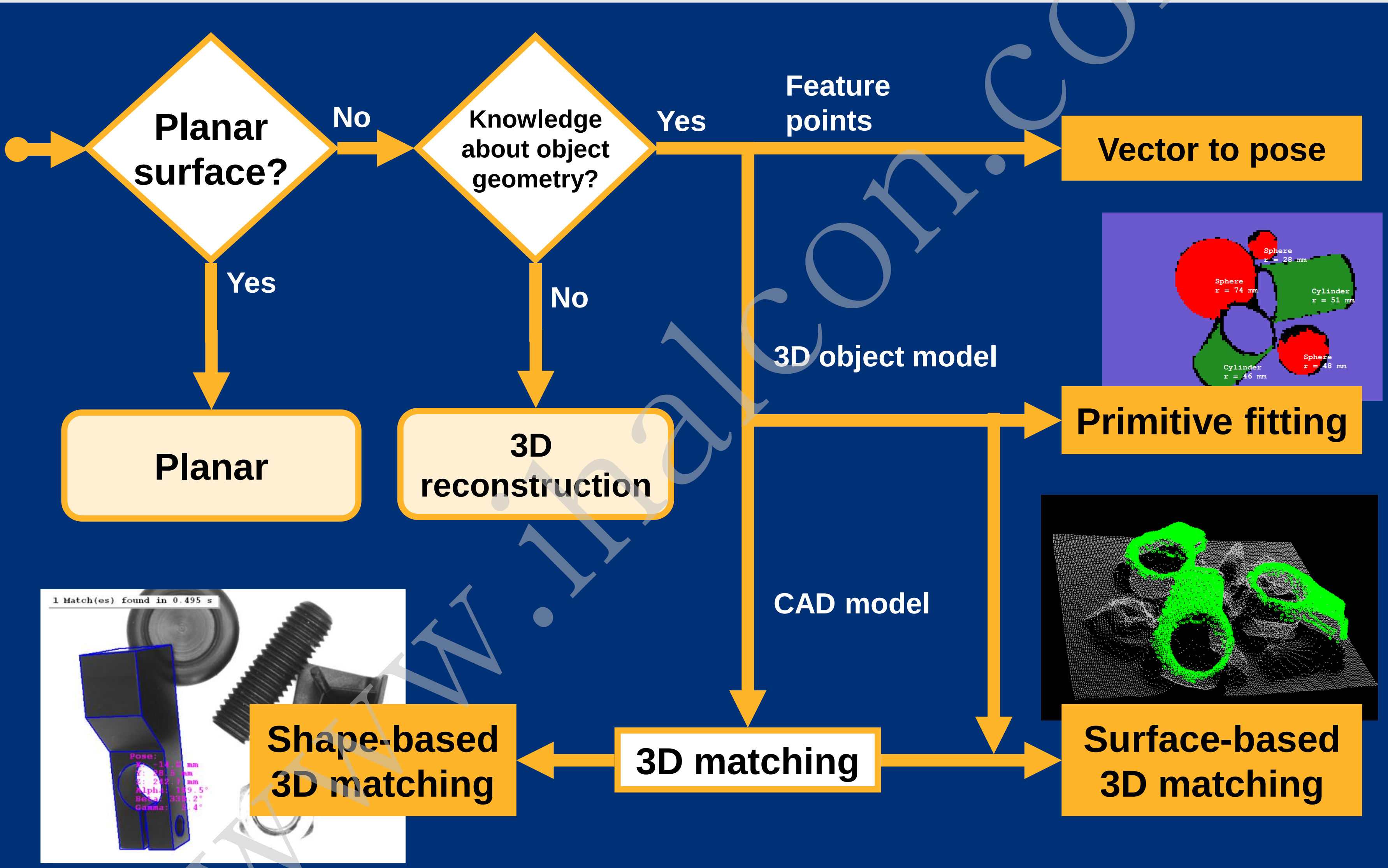
2D Rectification and Measurement

No

Depth from focus



Select Your Solution: 3D Methods II



Planar surface?

No

Yes

Planar

Knowledge about object geometry?

Yes

No

3D reconstruction

Feature points

Vector to pose

3D object model

Primitive fitting

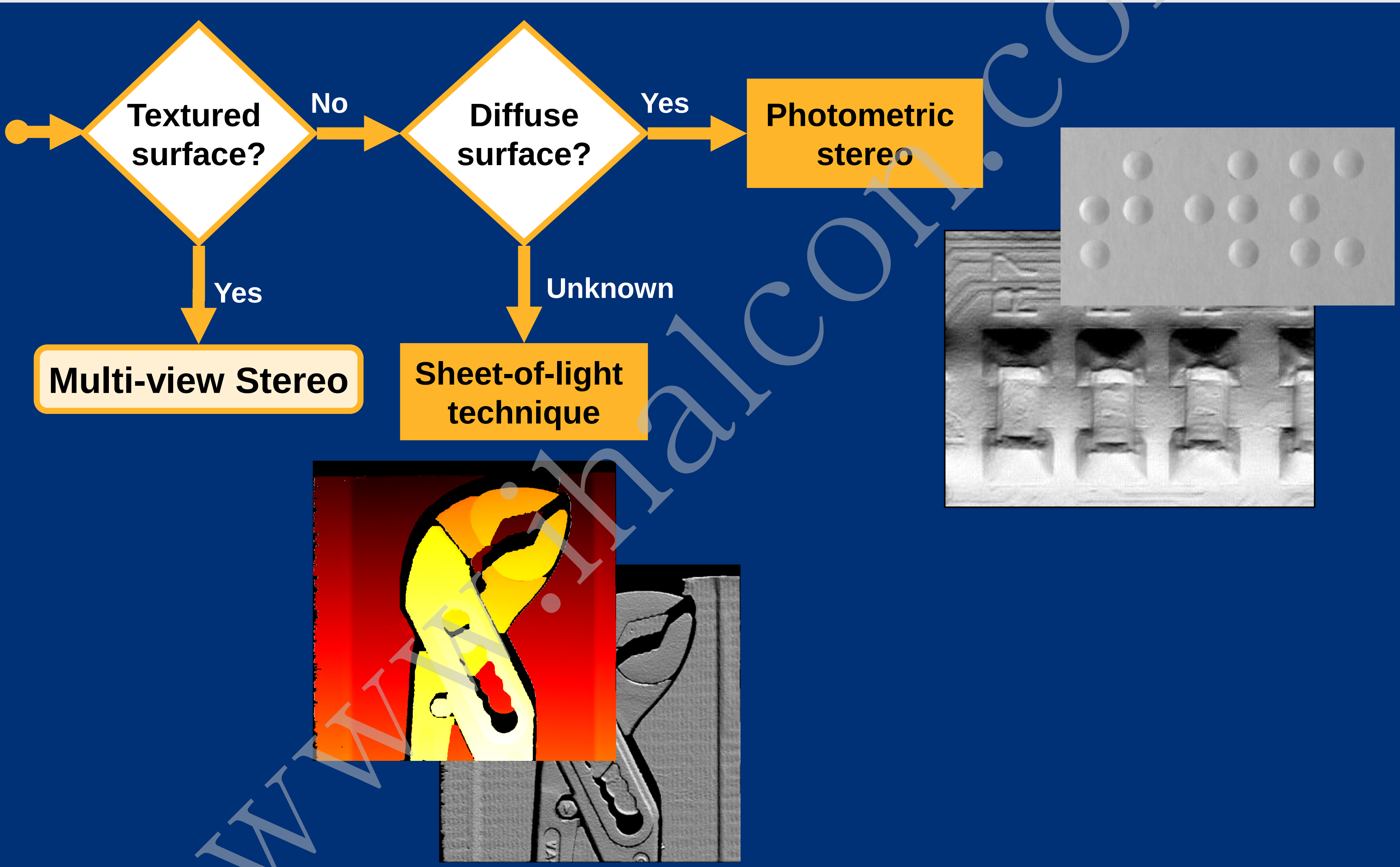
CAD model

3D matching

Shape-based 3D matching

Surface-based 3D matching

Select Your Solution: 3D Reconstruction



Textured surface?

No

Diffuse surface?

Yes

Photometric stereo

Yes

Multi-view Stereo

Unknown

Sheet-of-light technique

Select Your Solution: Planar

Geometric primitives?

No

Yes

3D position of circles or rectangles

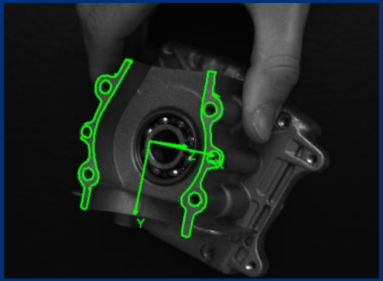
Clear edges?

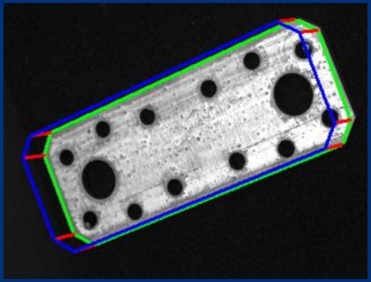
Yes

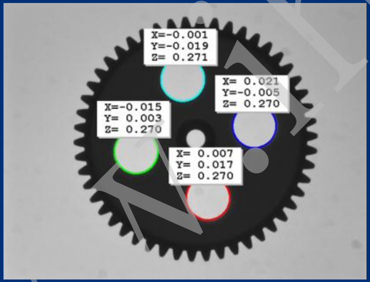
Perspective, deformable matching

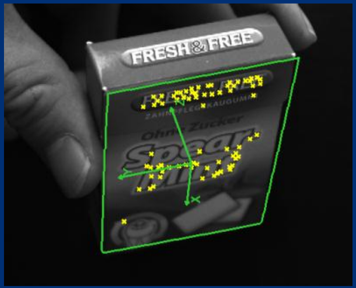
No

Descriptor-based matching

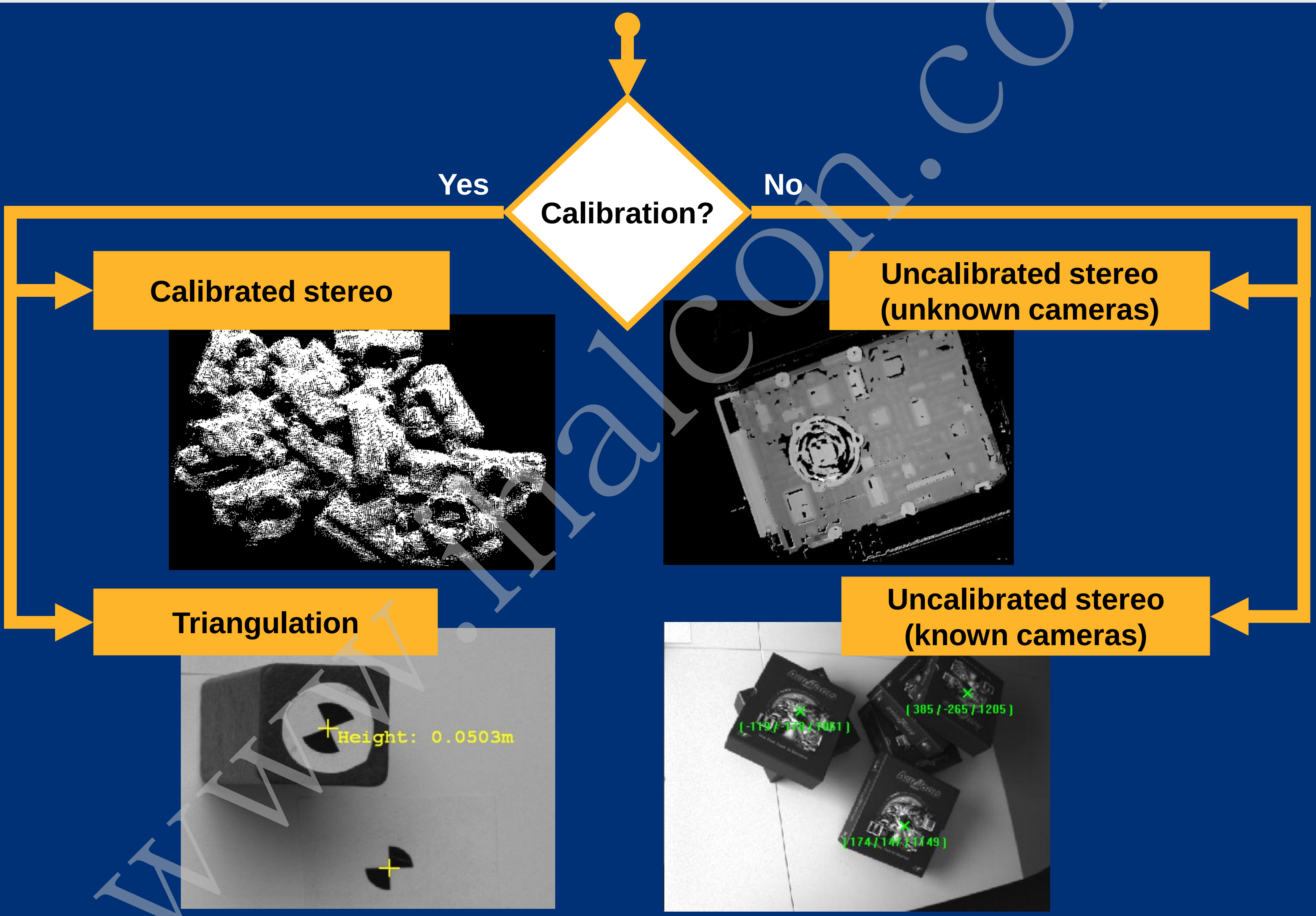




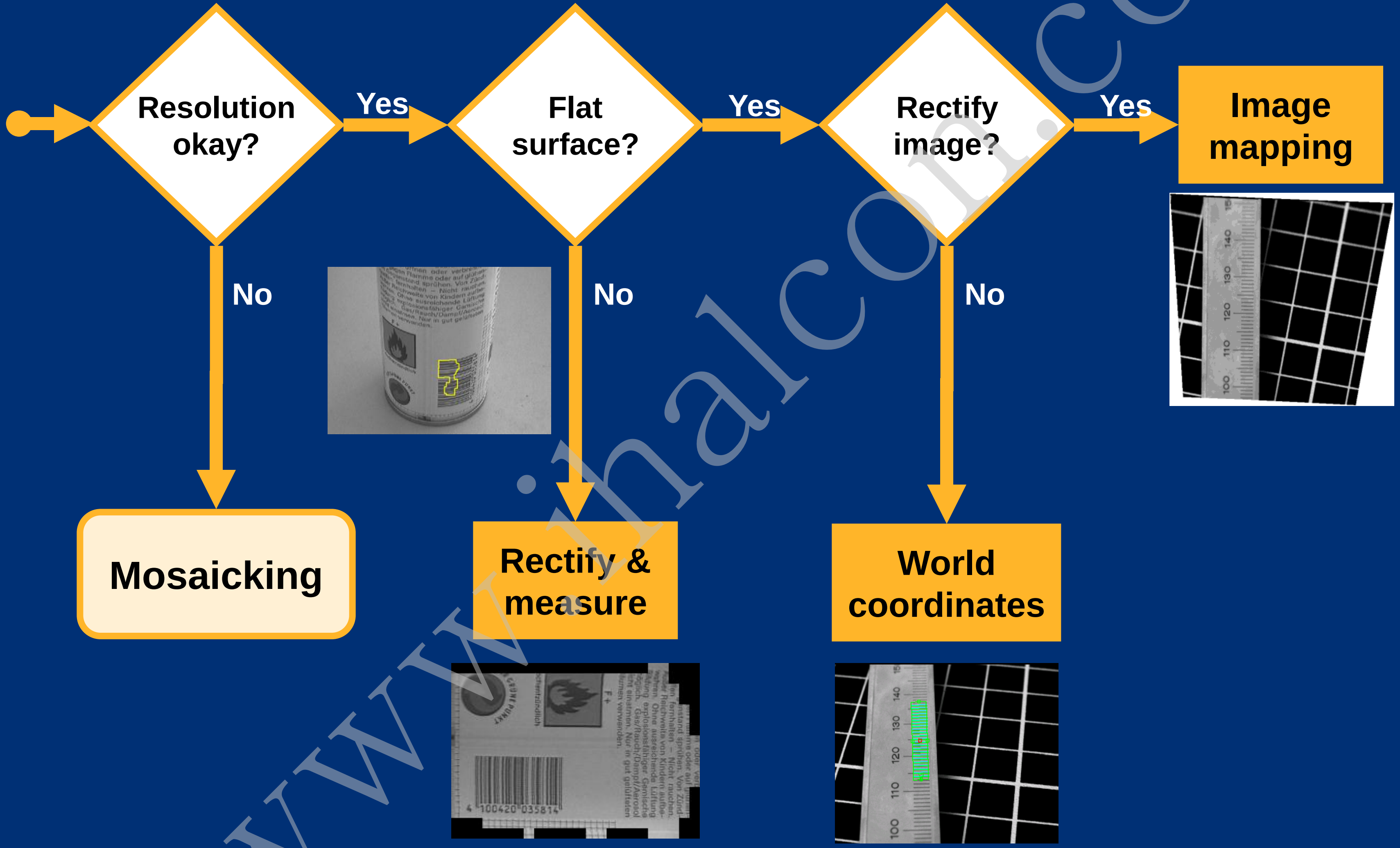




Select Your Solution: Stereo



Select Your Solution: 2D Rectification



Resolution okay?

Yes

No

Flat surface?

Yes

No

Rectify image?

Yes

No

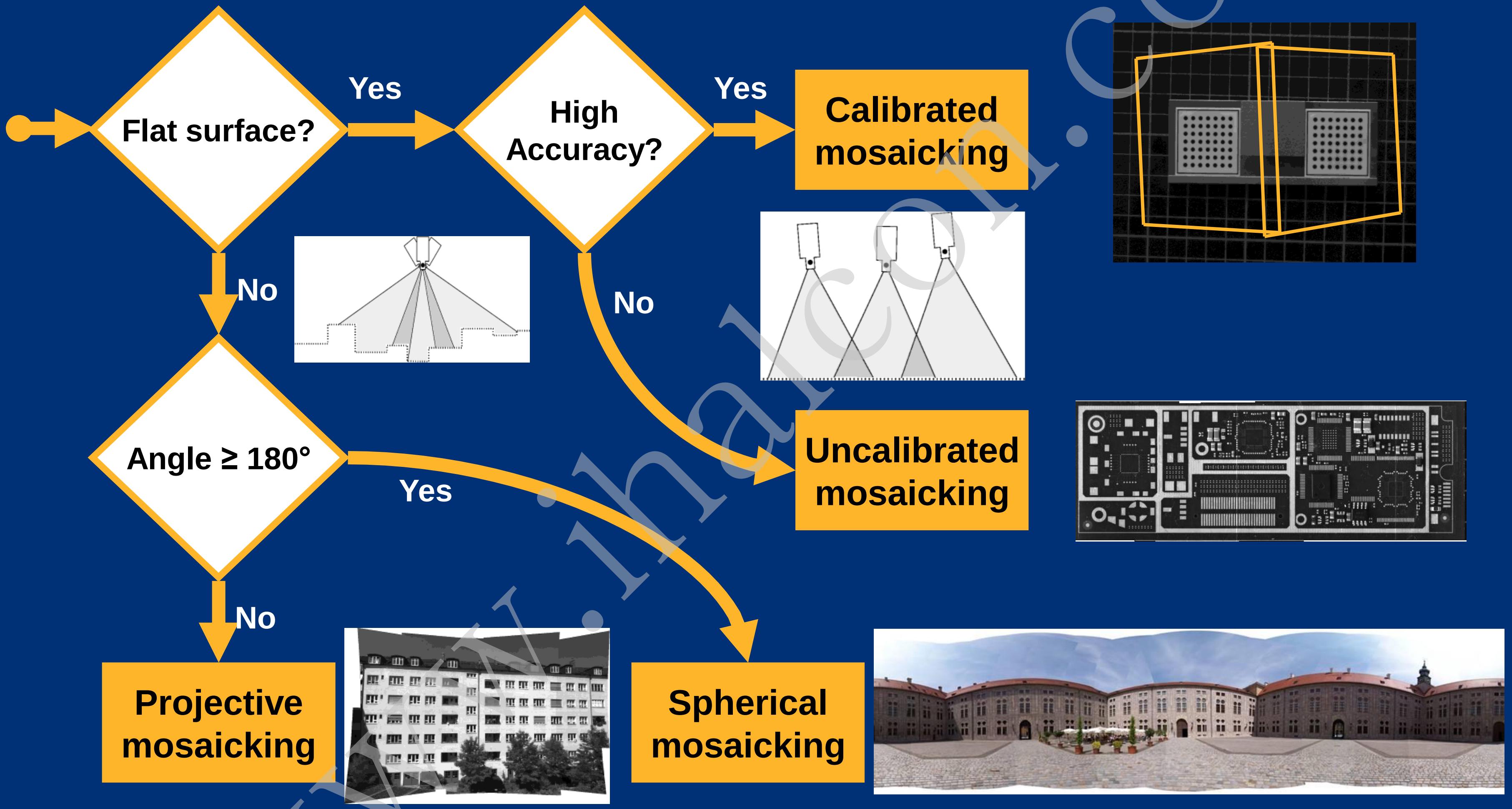
Image mapping

Mosaicking

Rectify & measure

World coordinates

Select Your Solution: Mosaicking



Flat surface?

Yes

High Accuracy?

Yes

Calibrated mosaicking

No

Uncalibrated mosaicking

No

Angle ≥ 180°

Yes

Spherical mosaicking

No

Projective mosaicking

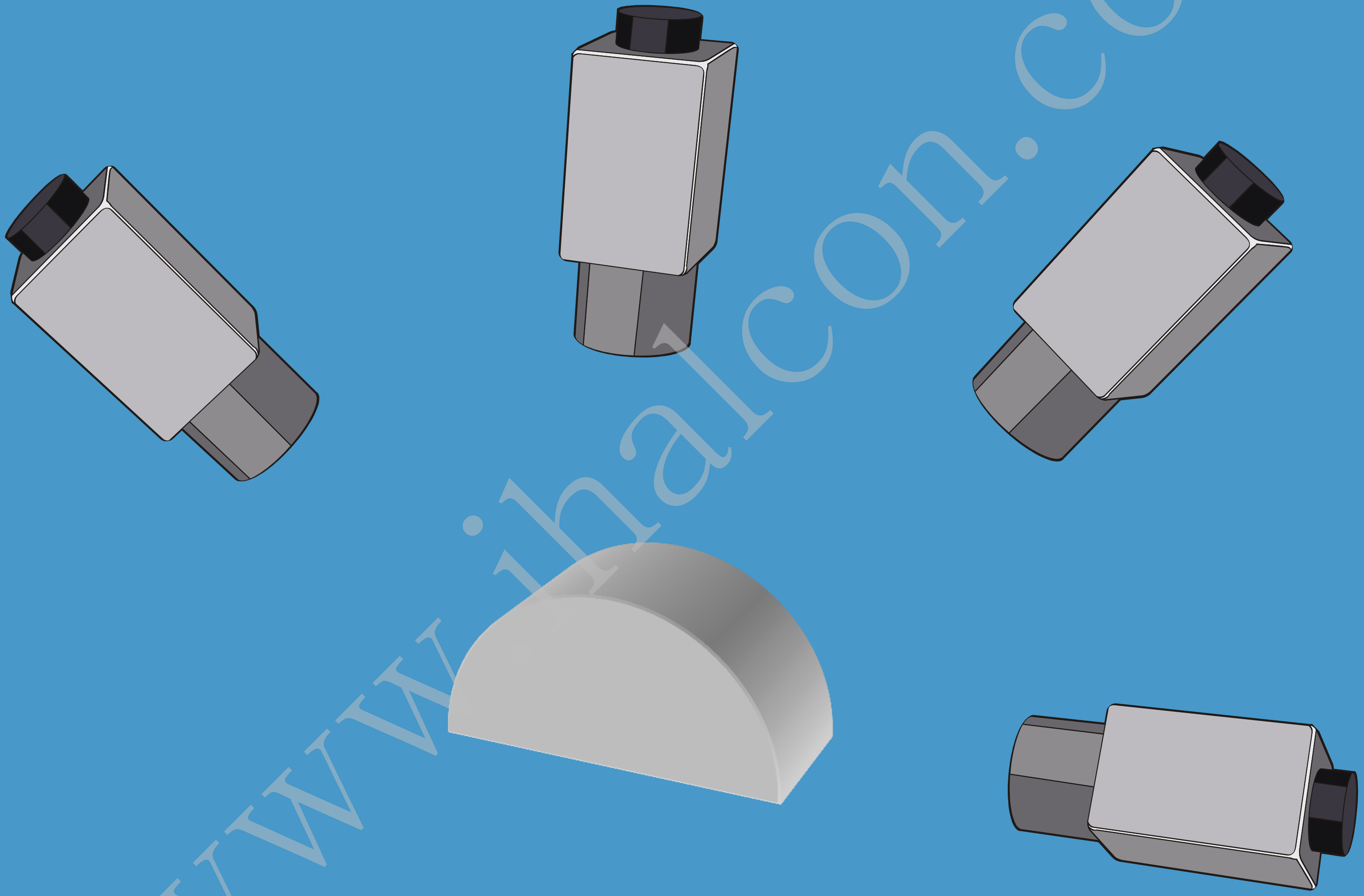
HALCON

the Power of Machine Vision

MVTEC

Multi-view Stereo

HALCON supports multi-view stereo

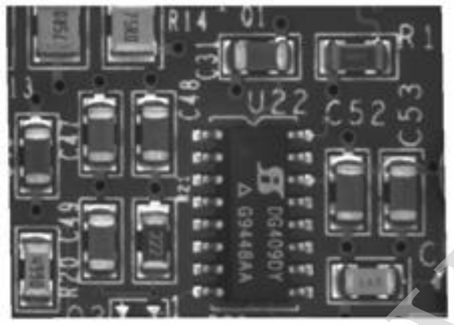


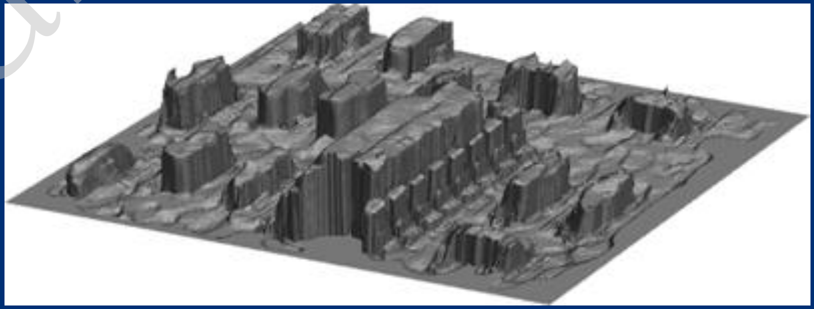
Overview

- Concept
- Stereo calibration process
- Reconstruction of points (triangulation)
- Epipolar contraint
- Image rectification
- Dense stereo reconstruction

Overview: Stereo — What for?

- Reconstruct the 3rd dimension
- Convert image coordinates to world coordinates
- Measure accurately in the real world
- Segment structures of unknown shape



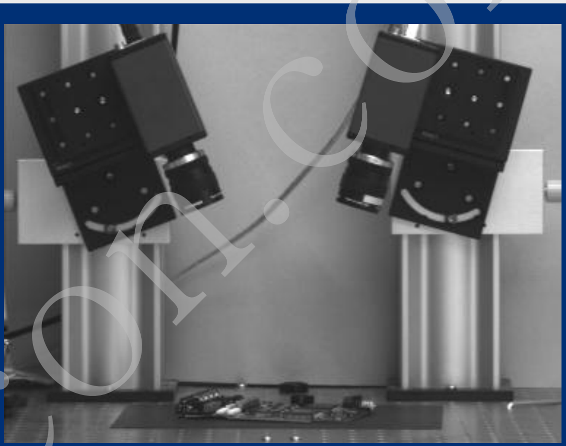


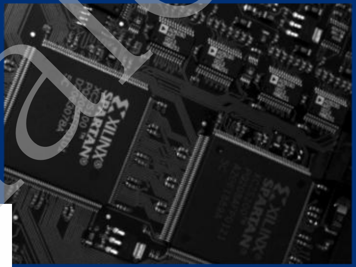
Overview: Typical Applications

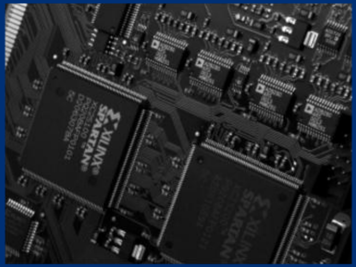
- Photogrammetry
- Micro electronics
- Quality inspections
  - Surface inspection
  - 3D measurements
- Robotics

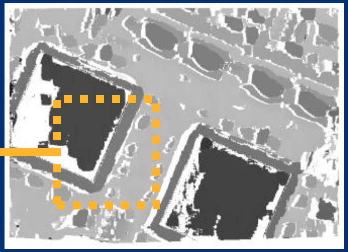
Overview: Conditions

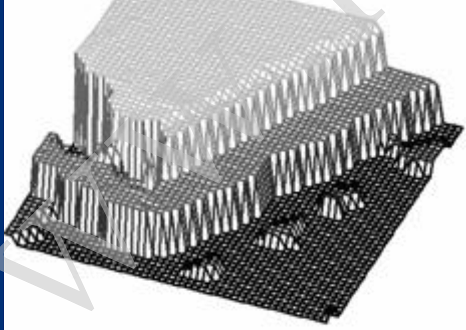
- Two or more projective cameras
- Measured objects visible in both images
- Similarly illuminated and exposed images
- The objects to be measured have significant texture











Idea of Binocular Reconstruction

Geometry of a binocular stereo set-up

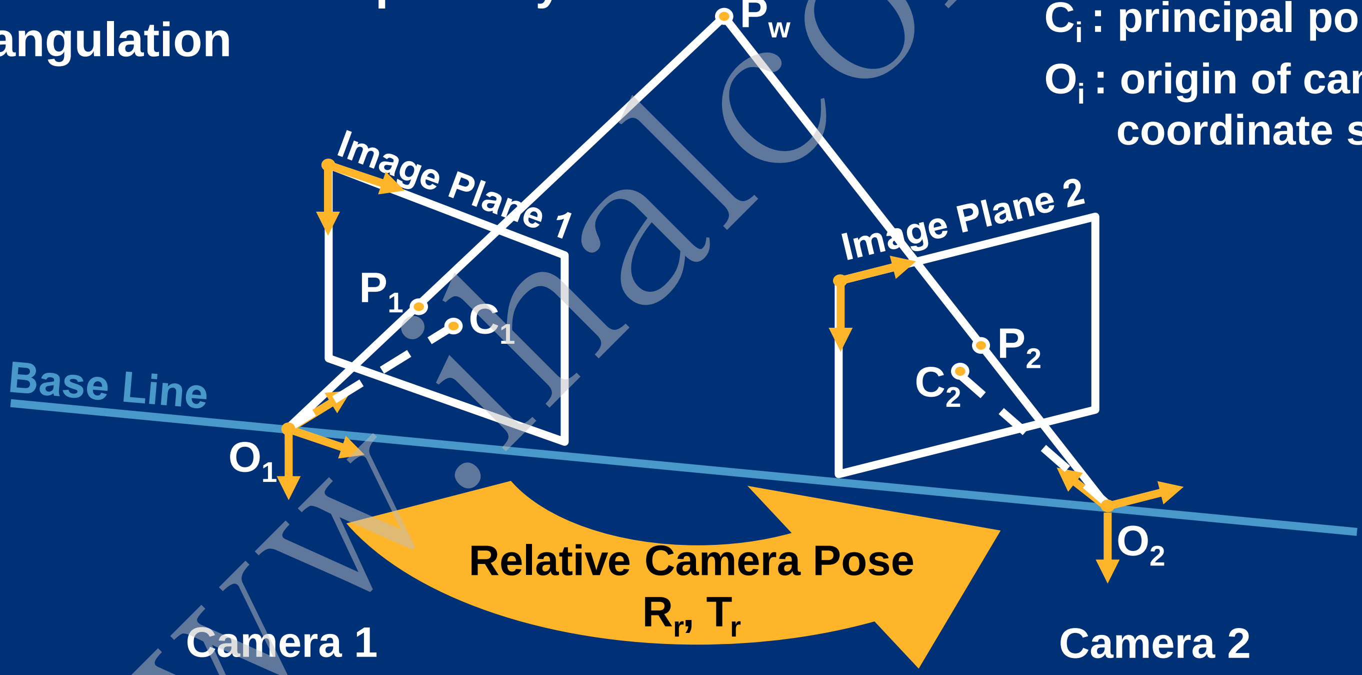
- Known image point correspondence
- Known camera parameters
- ⇨ Reconstruct world point by triangulation

P_w: world point

P_i : image point

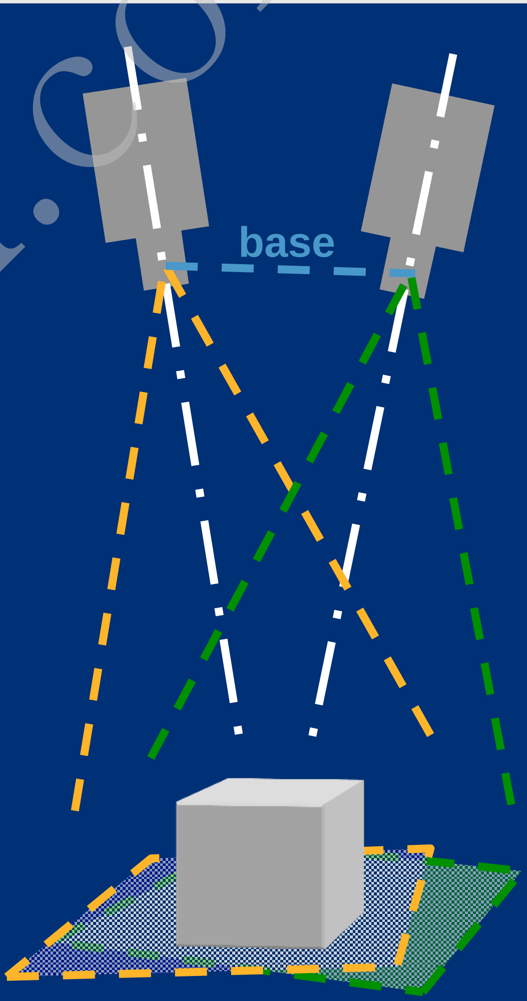
C_i : principal point

O_i : origin of camera coordinate system

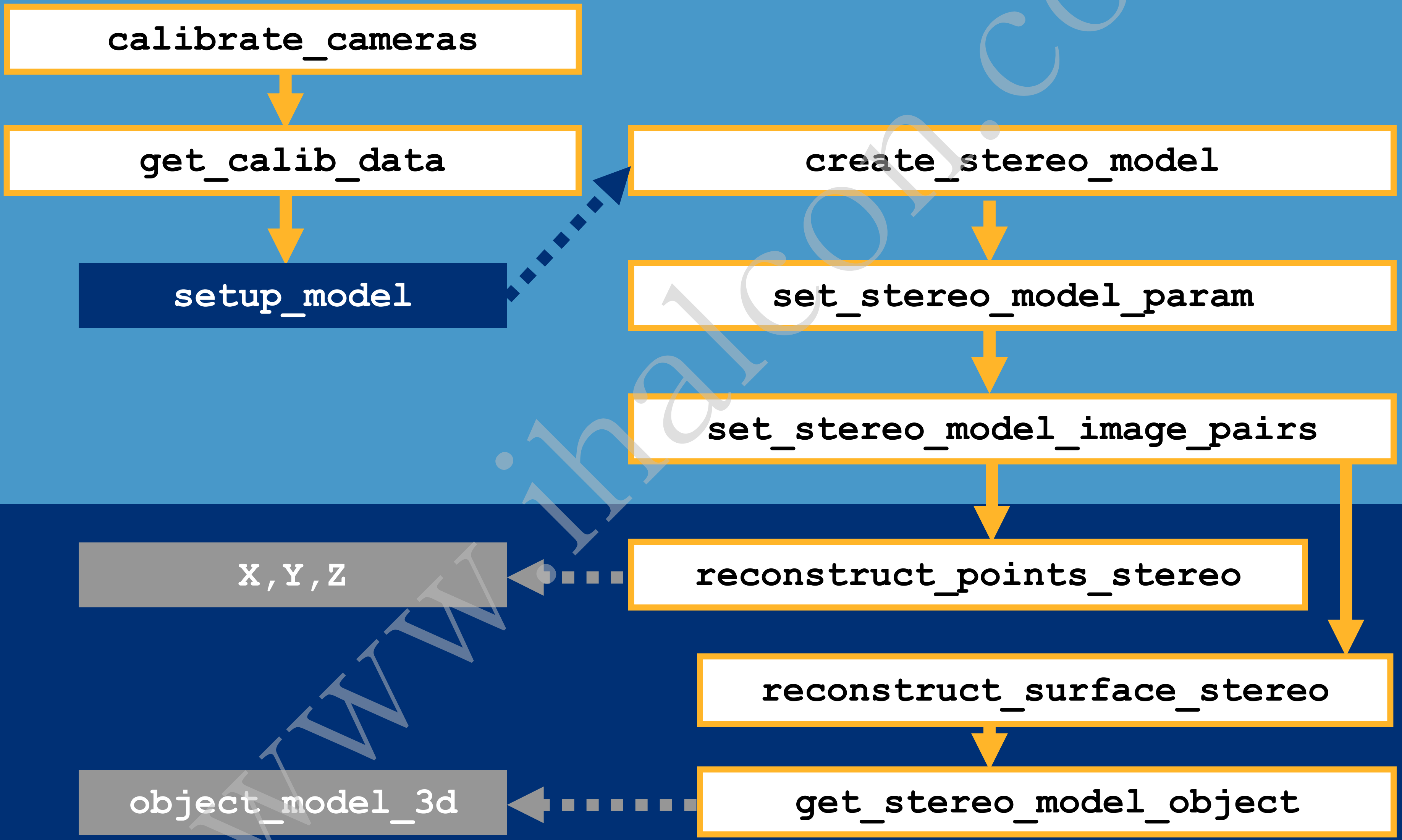


Rules to Arrange a Stereo Set-Up

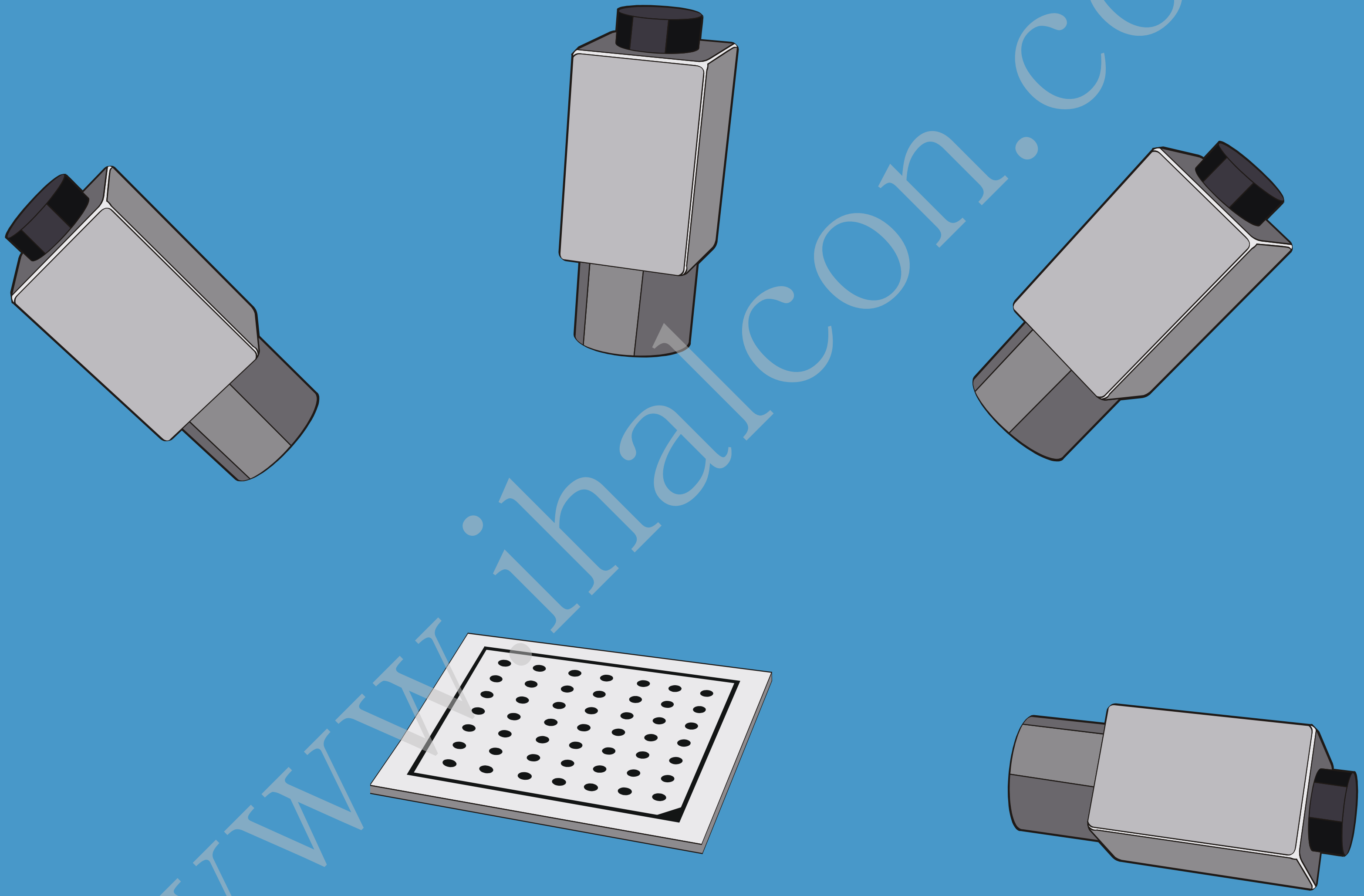
- Mount the cameras on a stable platform
- Maximize the overlapping image parts
- Find a trade off for setting the distance of the cameras (base):
  - the larger the base the higher the resolution of the distance
  - the smaller the base the less the occlusions and the more accurate the correspondences
- Set up the base line parallel to the objects' surfaces to be measured
- Provide illumination without reflections
- Use similar image resolutions for better results (focus, sensor)



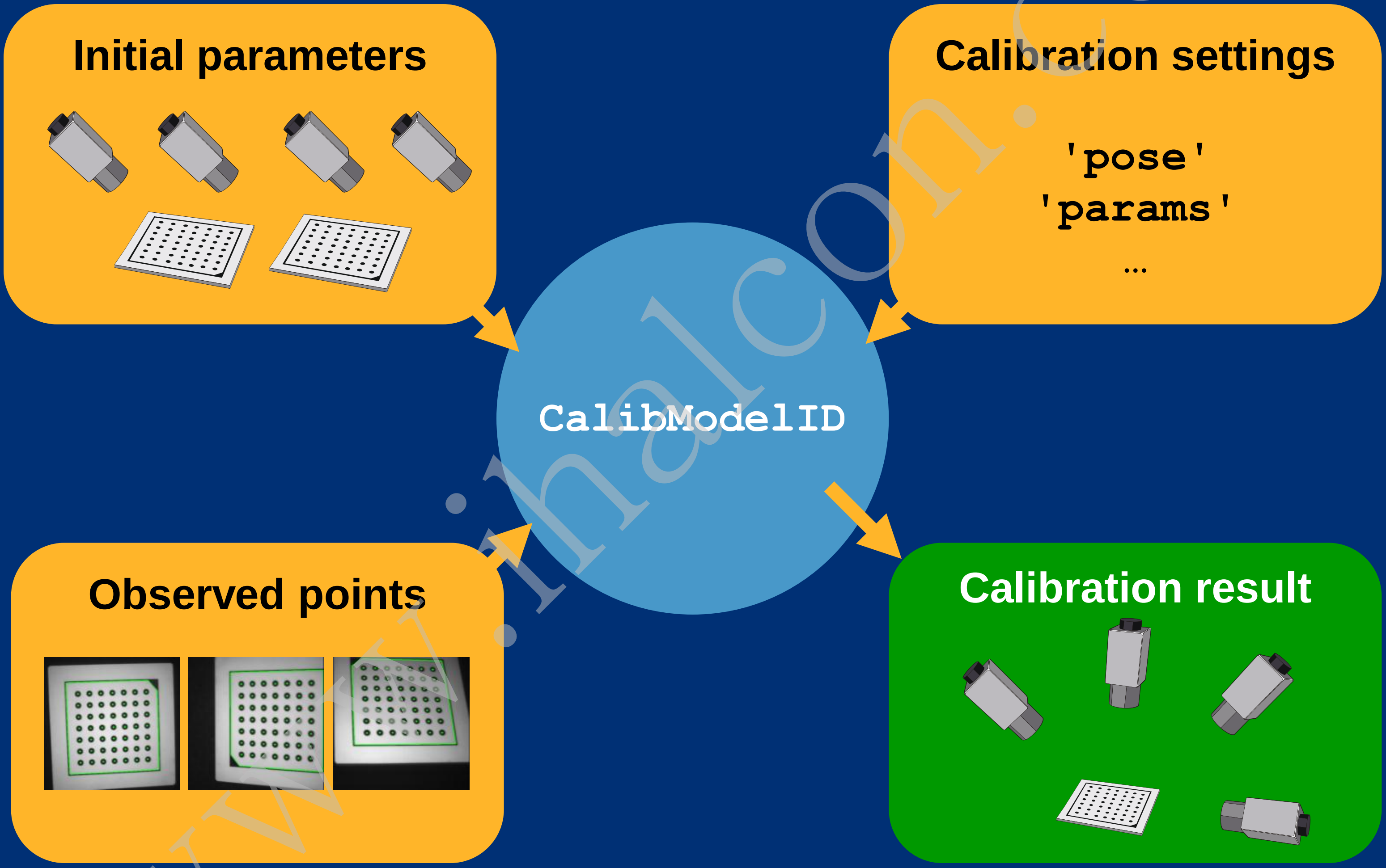
HALCON supports multi-view stereo



HALCON supports multi-view 3D calibration



HALCON offers a new calibration interface



Initial parameters

Calibration settings

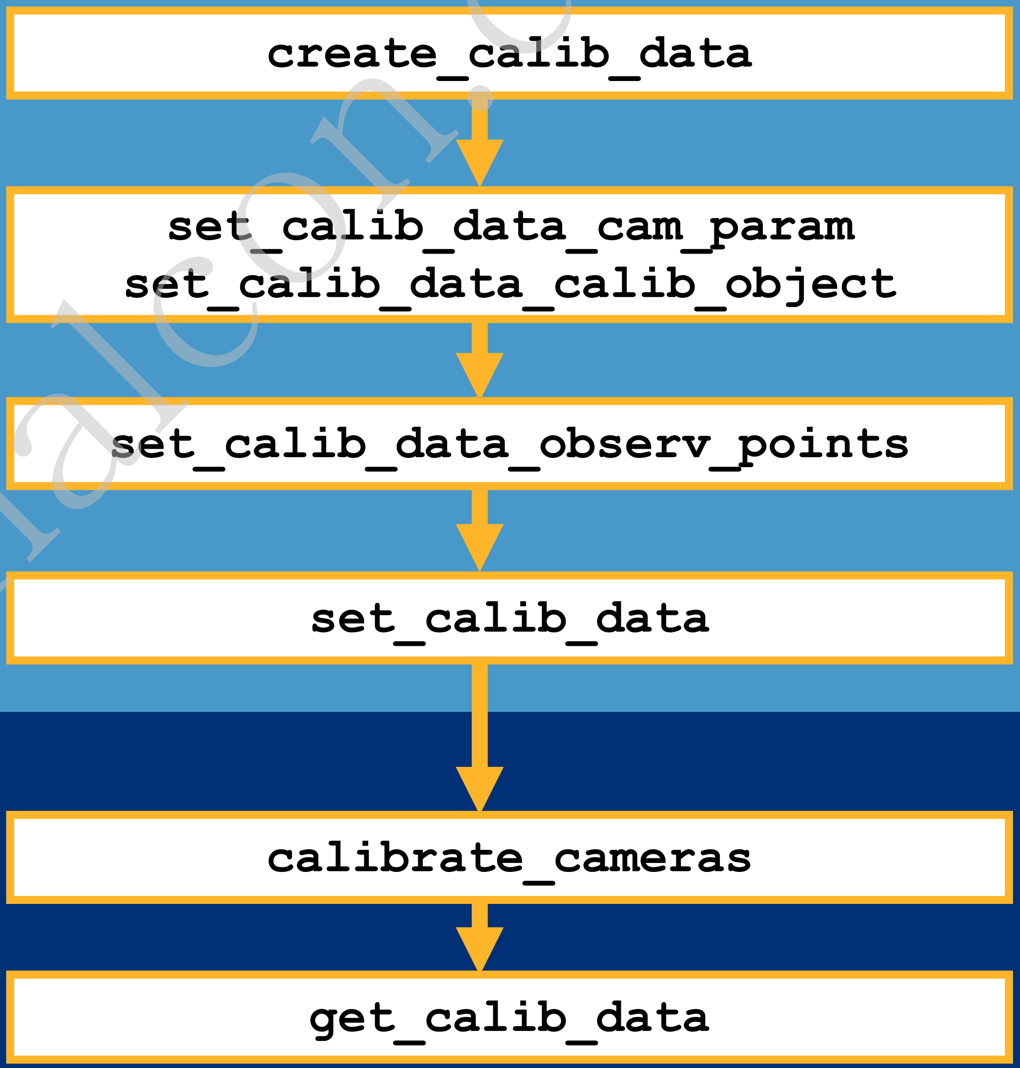
'pose'
'params'
...

CalibModelID

Observed points

Calibration result

Data gathering and parameter setup are handled by flexible operators



Configuration

Create model: `create_calib_data`

Configure calibration setup: `set_calib_data_cam_param`, `set_calib_data_calib_object`

Gather calibration data: `set_calib_data_observ_points`

Configure parameters: `set_calib_data`

Evaluation

Perform calibration: `calibrate_cameras`

Get results: `get_calib_data`

Configure calibration setup

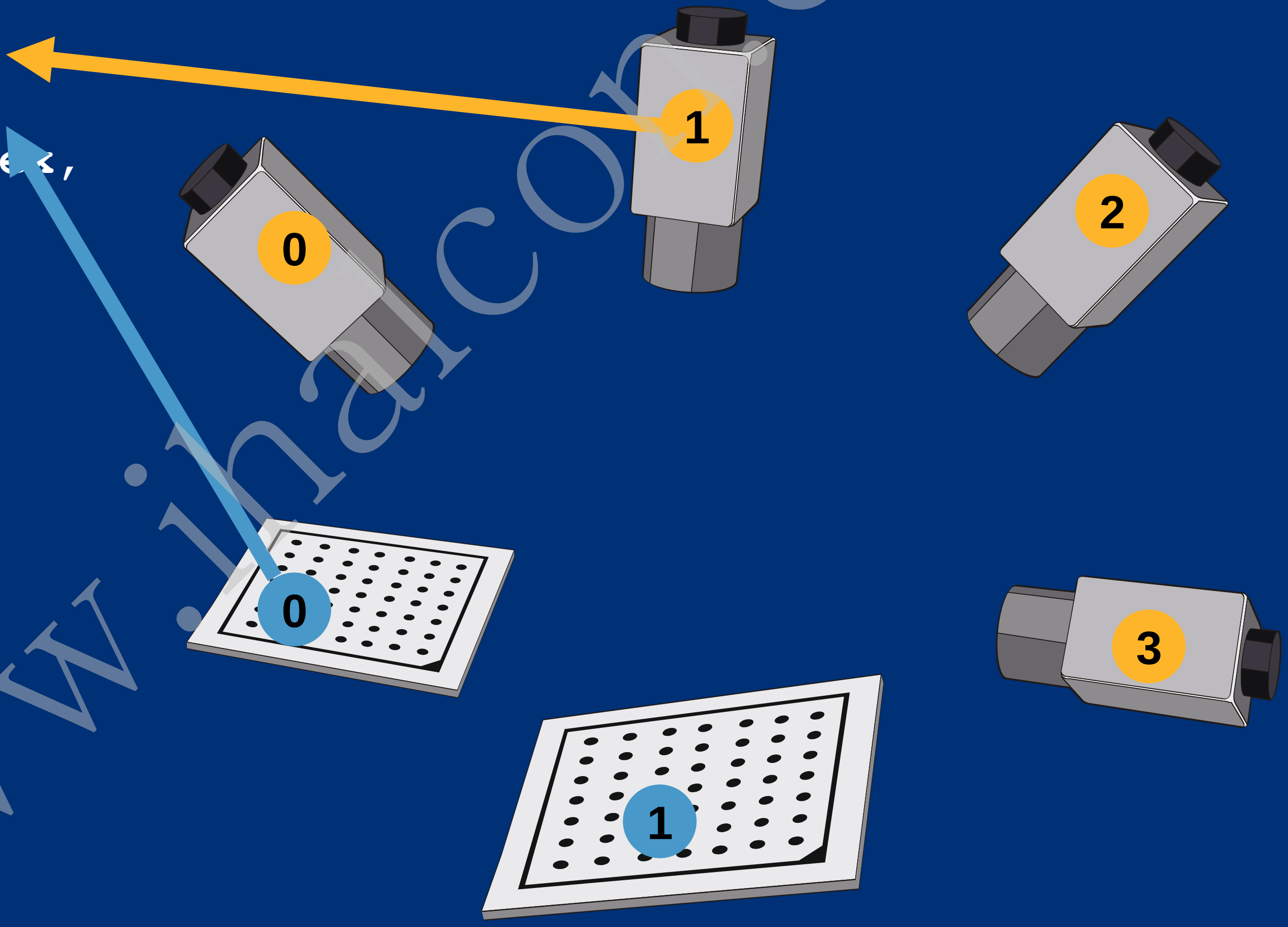
```
set_calib_data_calib_object ( ::
▶ CalibDataID,
▶ CalibObjIndex,
▶ CalibObjDescr :)
```

Different calibration objects possible

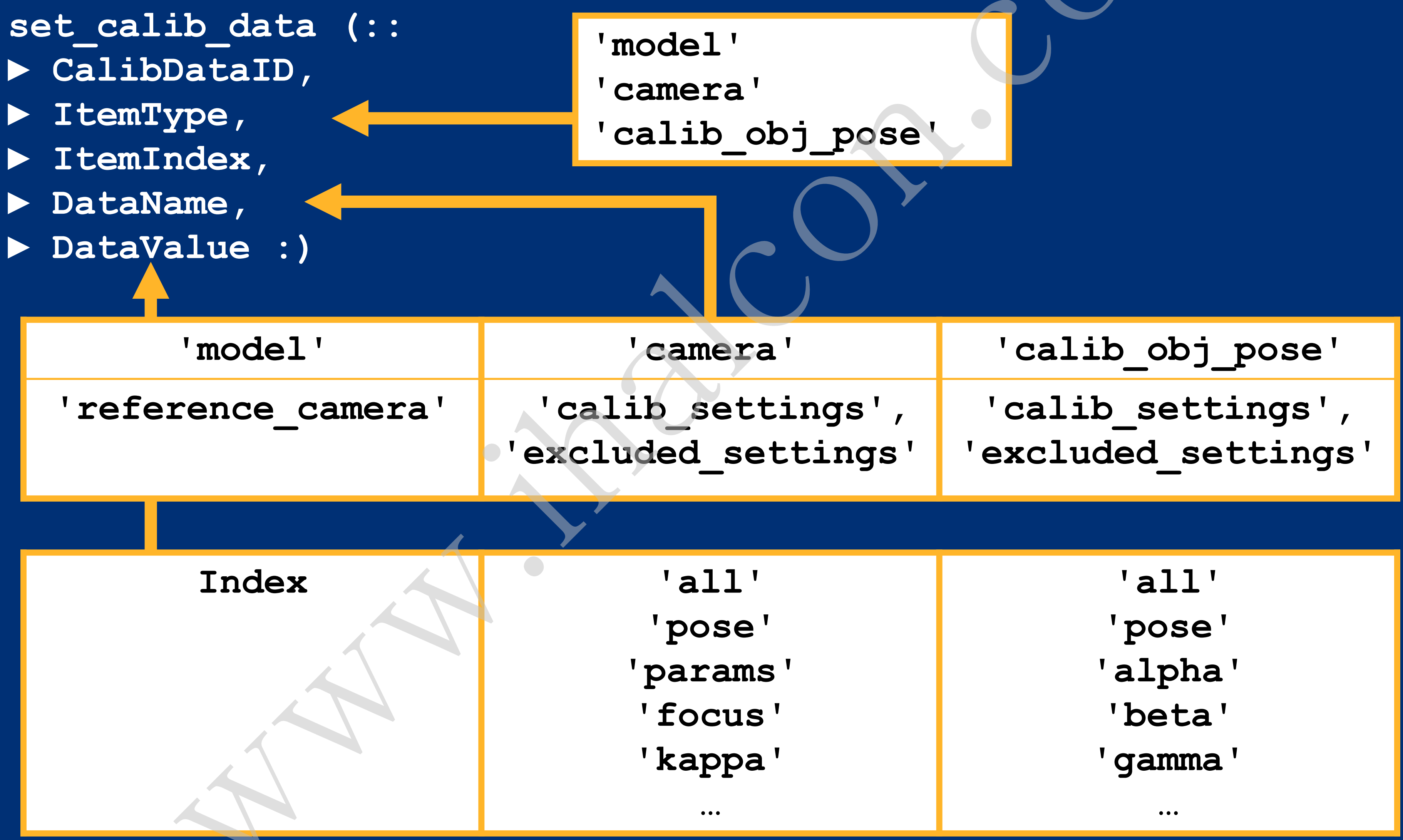
Description file

Gather calibration data

```
set_calib_data_observ_points (::
▶ CalibDataID,
▶ CameraIndex,
▶ CalibObjIndex,
▶ CalibObjPoseIndex,
▶ Row, Column,
▶ Index,
▶ Pose :)
```



Configure optimization parameters

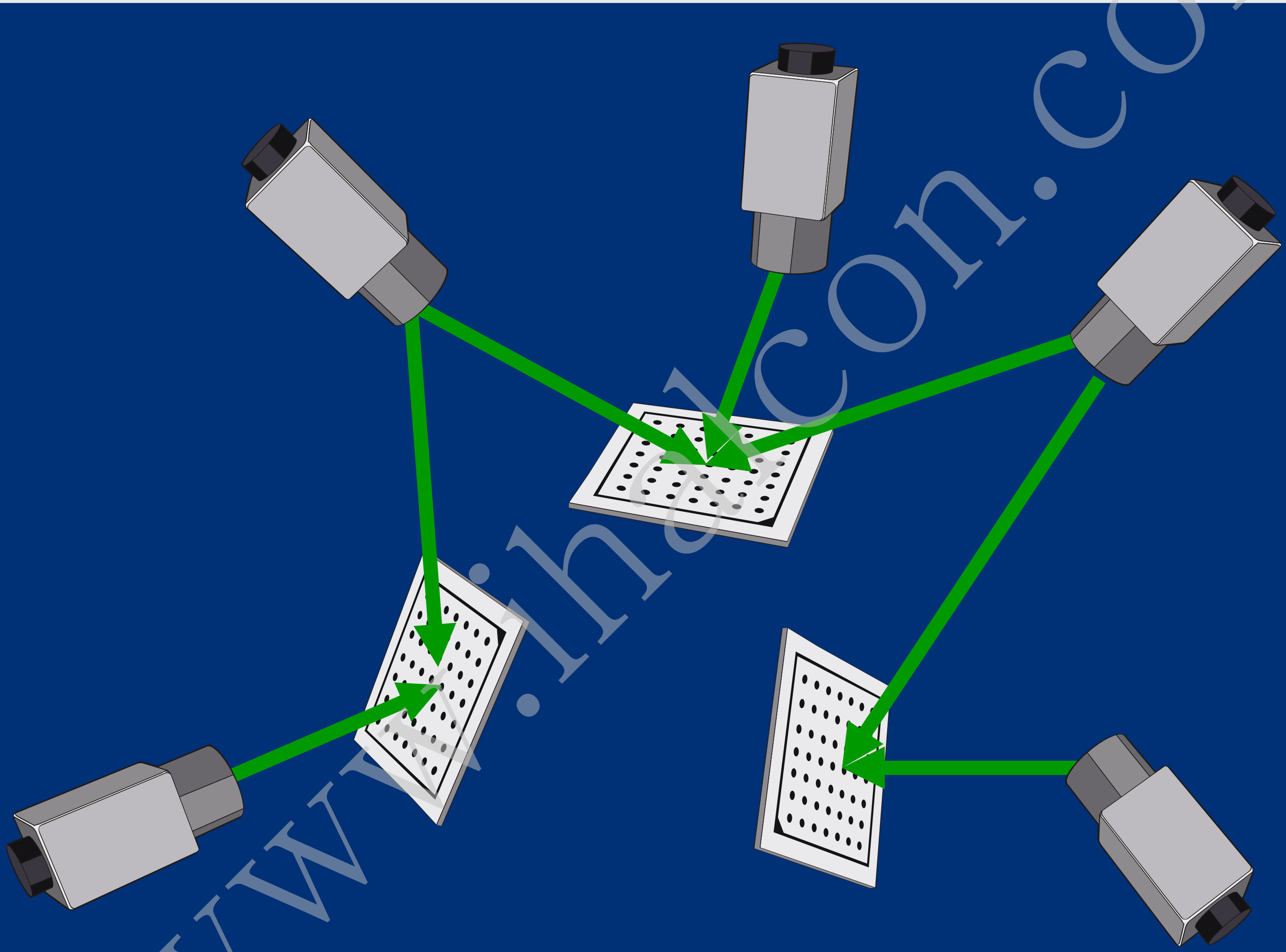


```
set_calib_data (::
▶ CalibDataID,
▶ ItemType,
▶ ItemIndex,
▶ DataName,
▶ DataValue :)
```

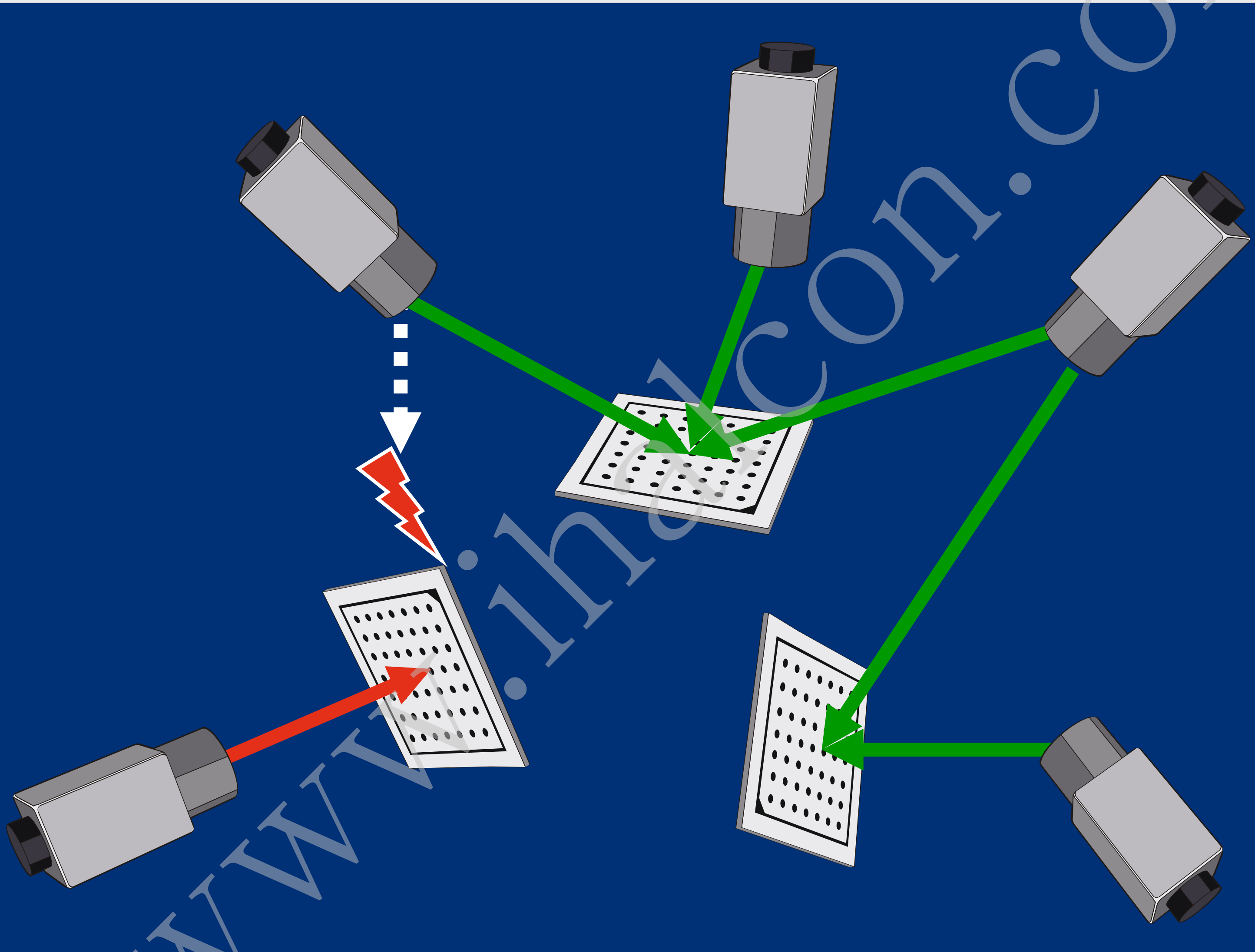
'model'
'camera'
'calib_obj_pose'

'model'	'camera'	'calib_obj_pose'
'reference_camera'	'calib_settings', 'excluded_settings'	'calib_settings', 'excluded_settings'
Index	'all' 'pose' 'params' 'focus' 'kappa' …	'all' 'pose' 'alpha' 'beta' 'gamma' …

All cameras must be connected through poses



All cameras must be connected through poses



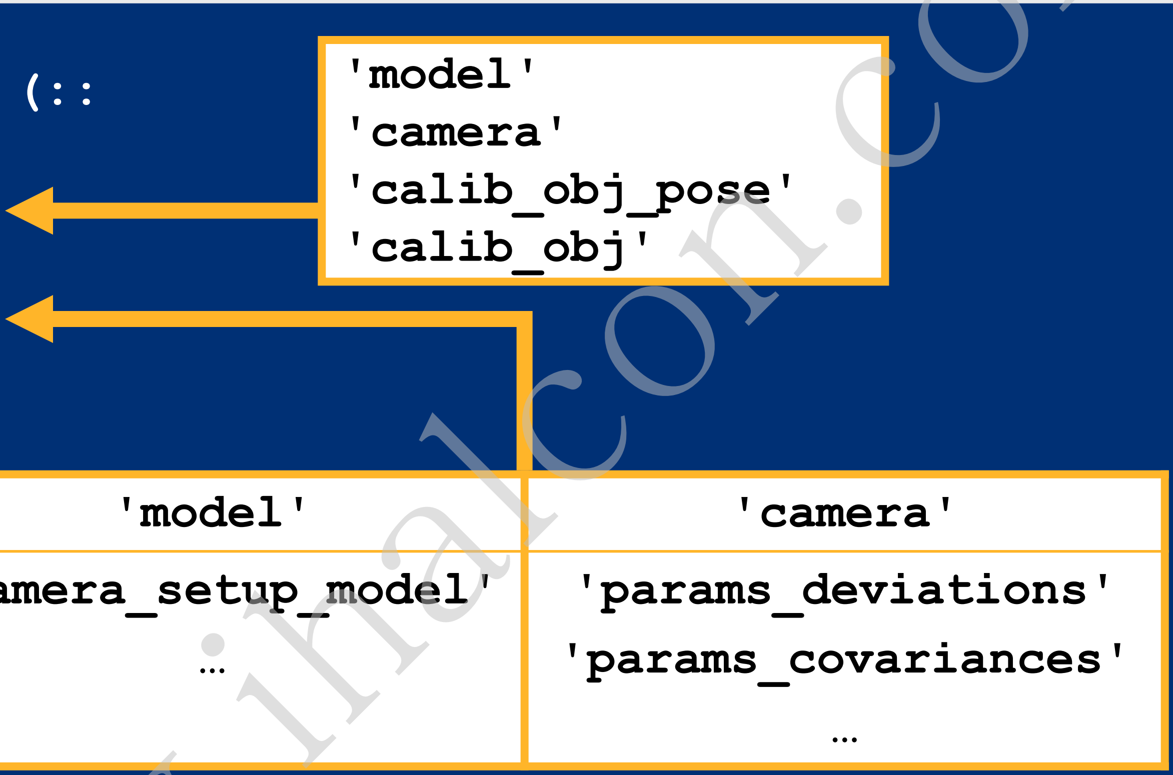
Perform calibration

```
calibrate_cameras (::
▶ CalibDataID :
◀ Error)
```

Average back-projection error in pixels

Analyze calibration results

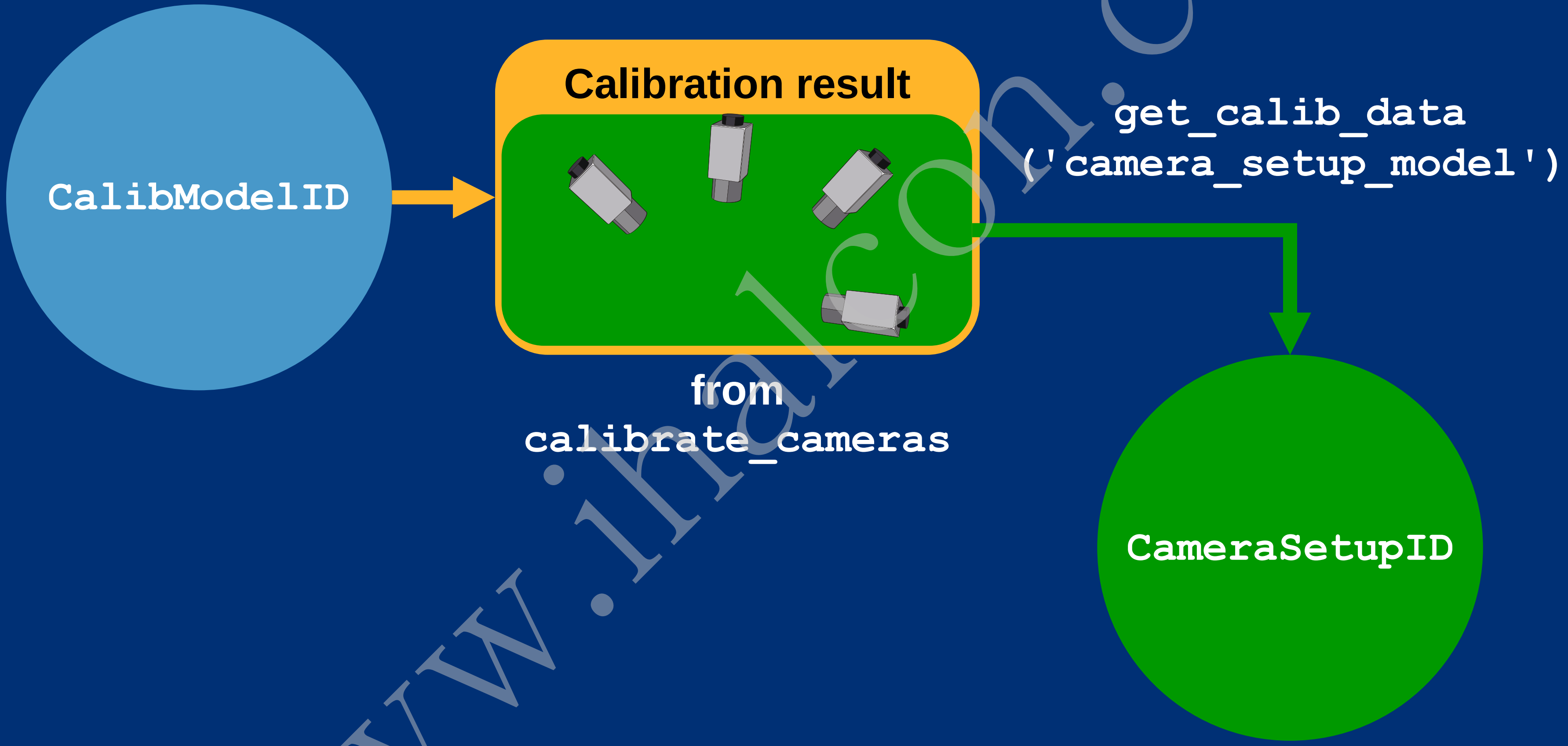
```
get_calib_data (::
▶ CalibDataID,
▶ ItemType,
▶ ItemIndex,
▶ DataName,
▶ DataValue :)
```



'model'
'camera'
'calib_obj_pose'
'calib_obj'

'model'	'camera'
'camera_setup_model' …	'params_deviations' 'params_covariances' …

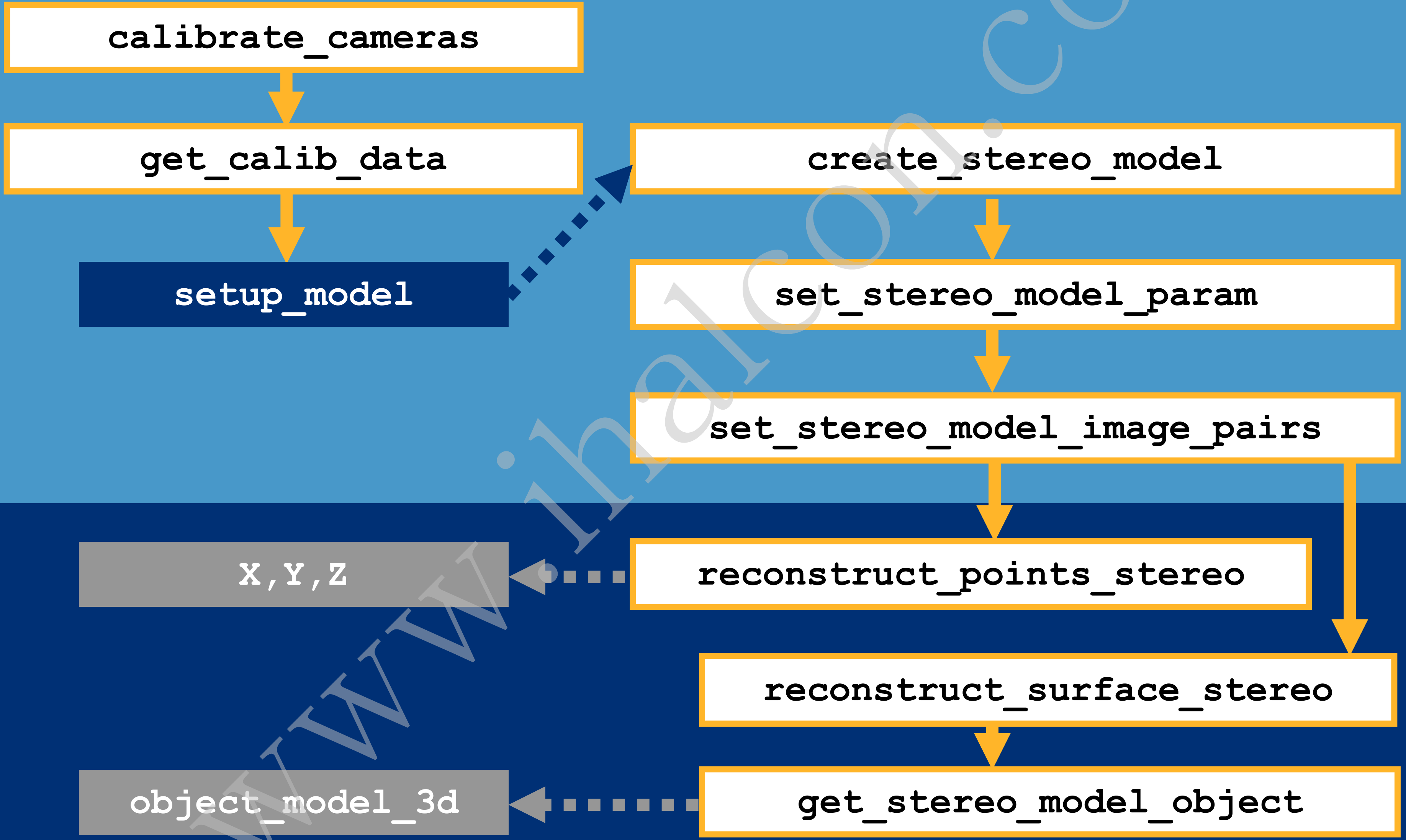
Multi-view stereo uses a camera setup model



from calibrate_cameras

Camera parameters and poses

HALCON supports multi-view stereo



calibrate_cameras

get_calib_data

setup_model

create_stereo_model

set_stereo_model_param

set_stereo_model_image_pairs

X,Y,Z

reconstruct_points_stereo

reconstruct_surface_stereo

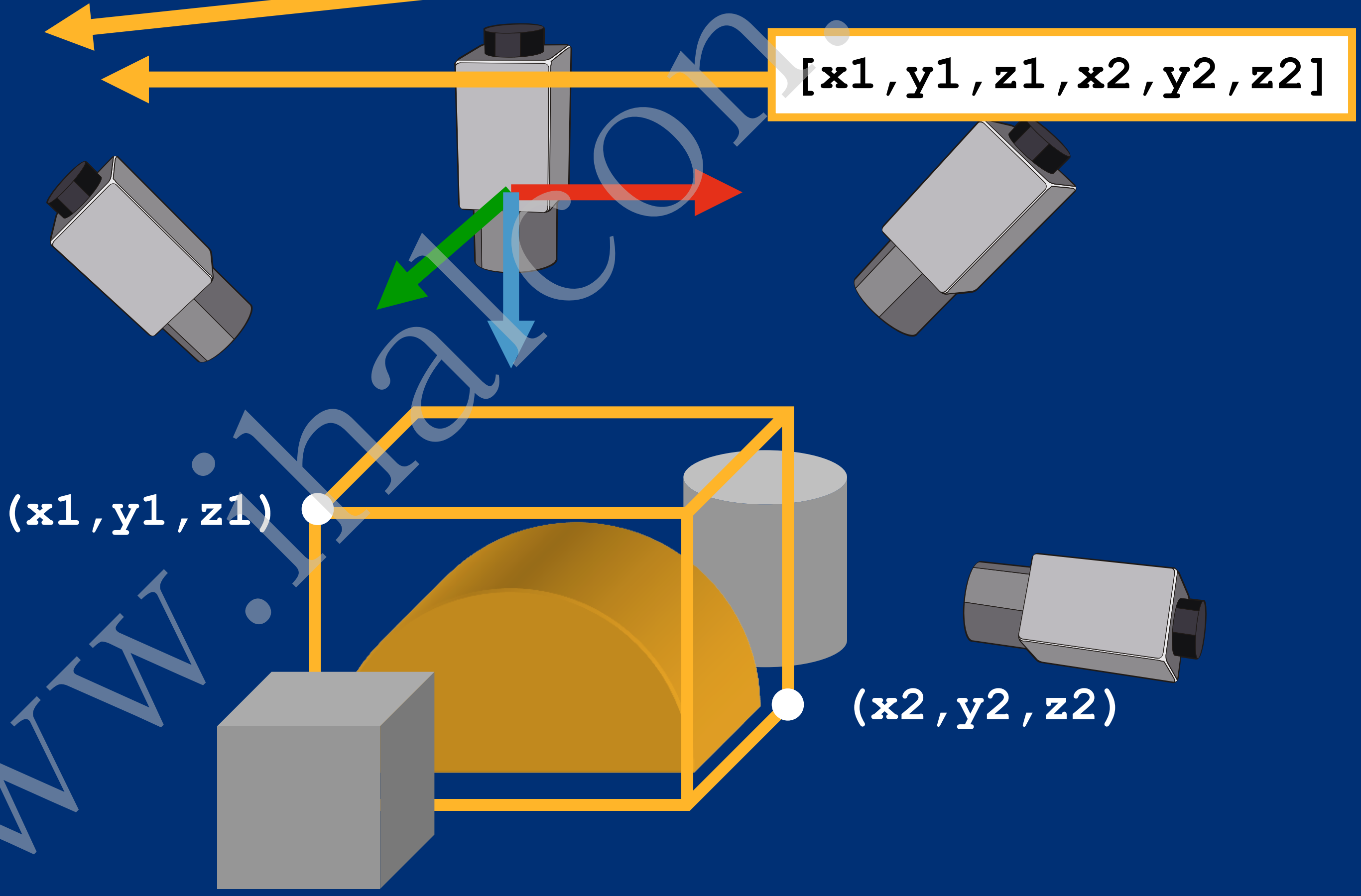
object_model_3d

get_stereo_model_object

Setting the bounding box is important

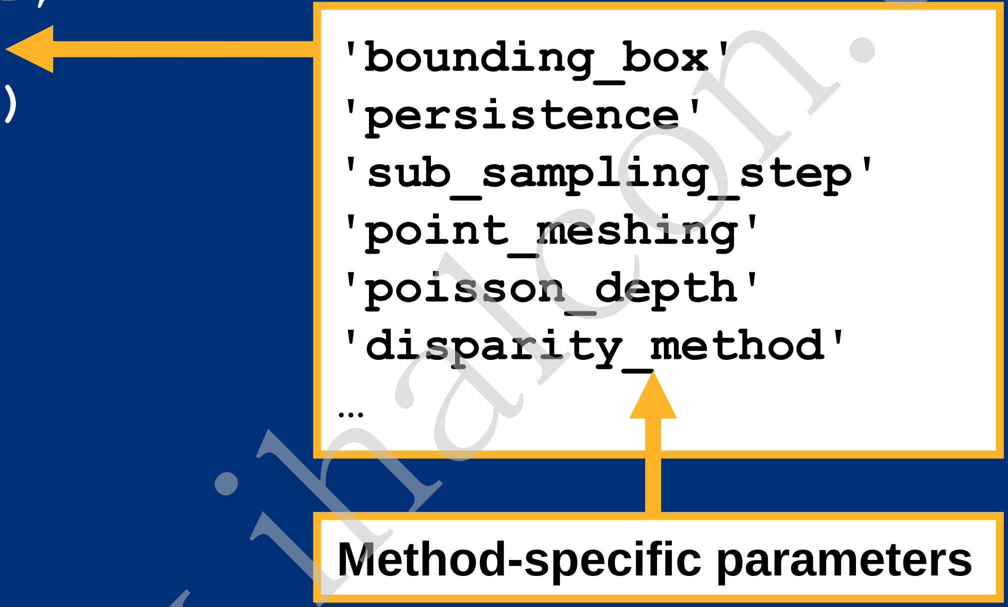
```
set_stereo_model_param (::
▶ StereoModelID,
▶ ParamName,
▶ ParamValue :)
```

'bounding_box'



Multi-view stereo is very flexible

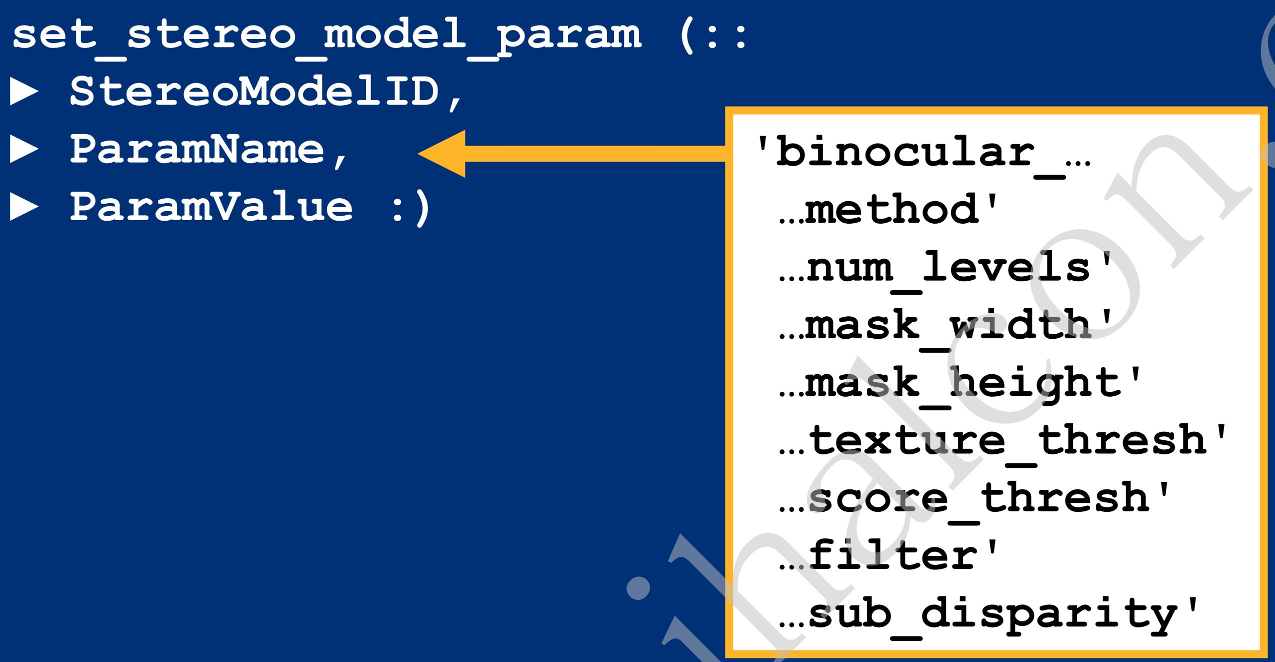
```
set_stereo_model_param (::
▶ StereoModelID,
▶ ParamName,
▶ ParamValue :)
```



```
'bounding_box'
'persistence'
'sub_sampling_step'
'point_meshing'
'poisson_depth'
'disparity_method'
...
```

Method-specific parameters

Multi-view stereo uses pairwise binocular stereo



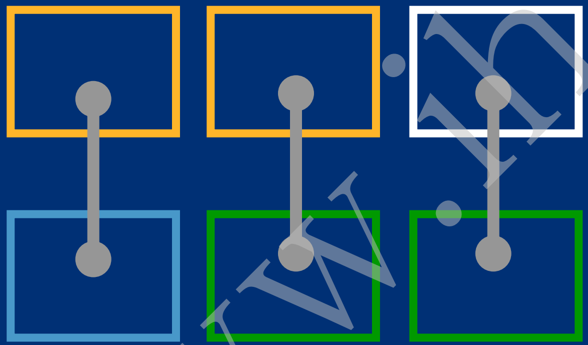
```
set_stereo_model_param (: :
▶ StereoModelID,
▶ ParamName,
▶ ParamValue :)
```

```
'binocular_…
 …method'
 …num_levels'
 …mask_width'
 …mask_height'
 …texture_thresh'
 …score_thresh'
 …filter'
 …sub_disparity'
```

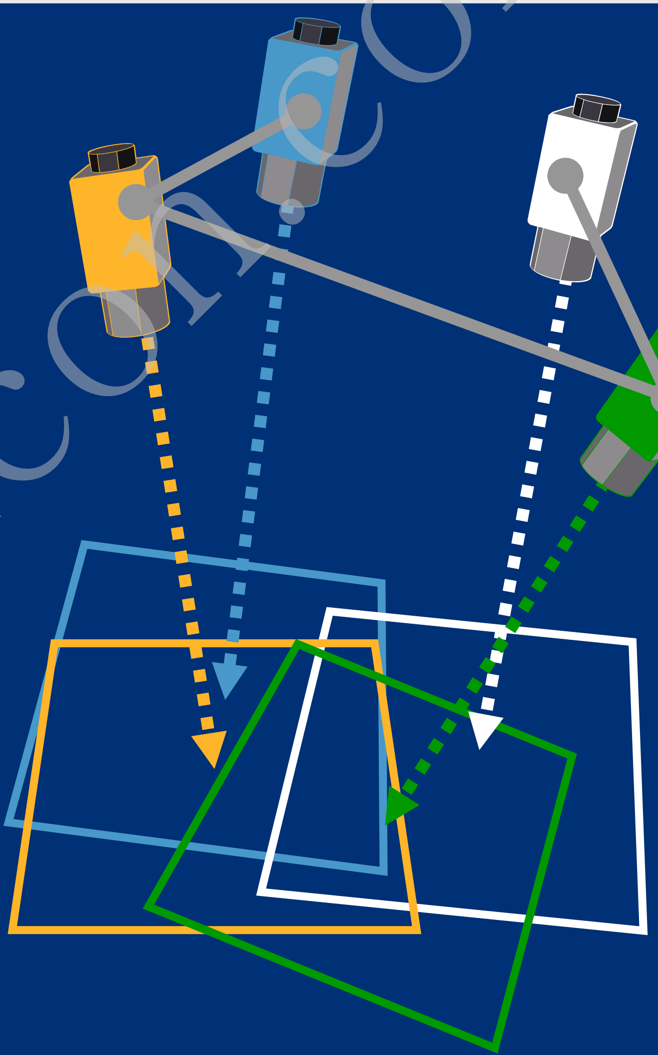
Set the stereo image pairs with
set_stereo_model_image_pairs

```
set_stereo_model_image_pairs (::
   ► StereoModelID,
   ► From,
   ► To :)
```

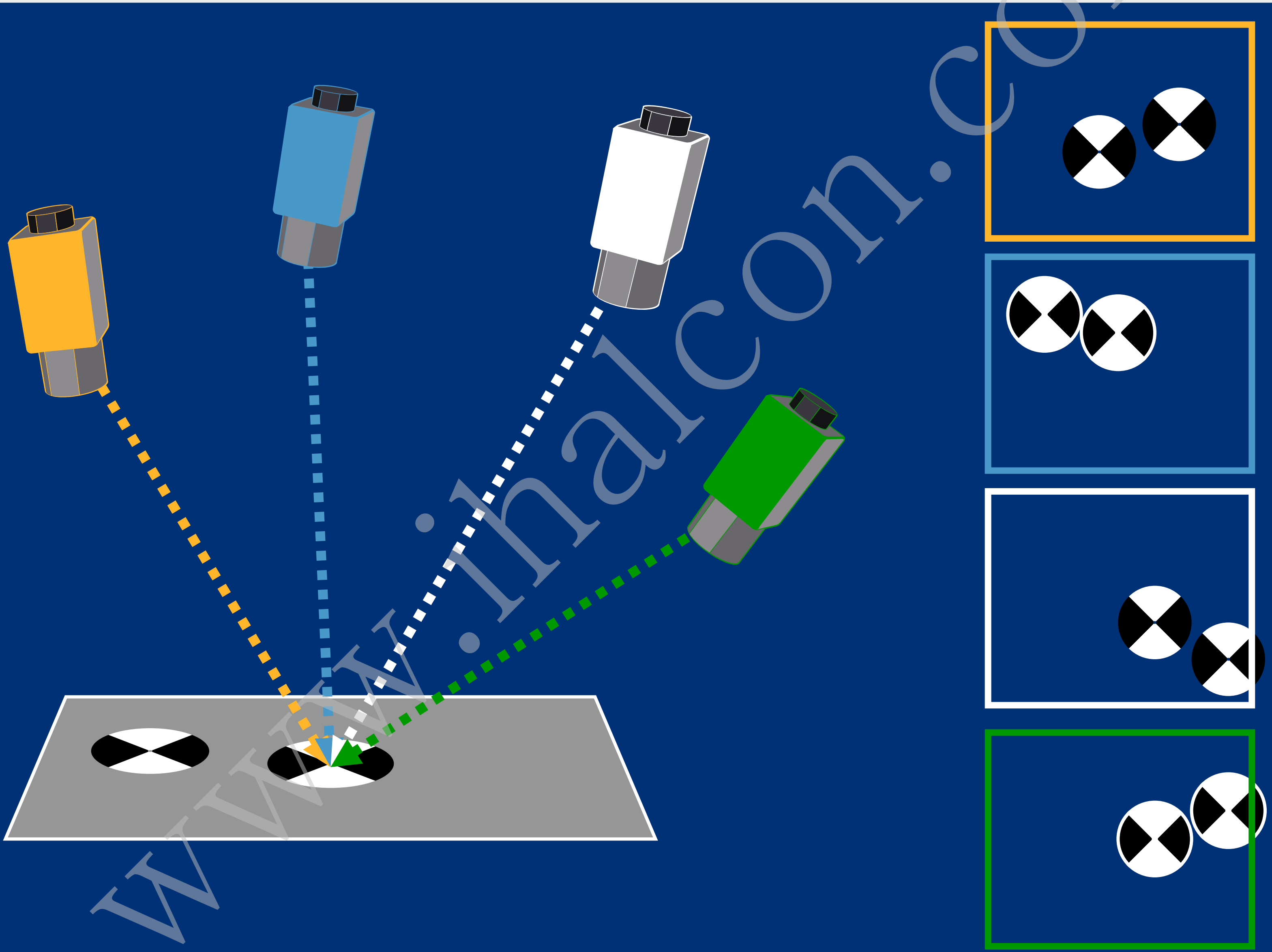
From = [0 , 0 , 2]



To = [1 , 3 , 3]

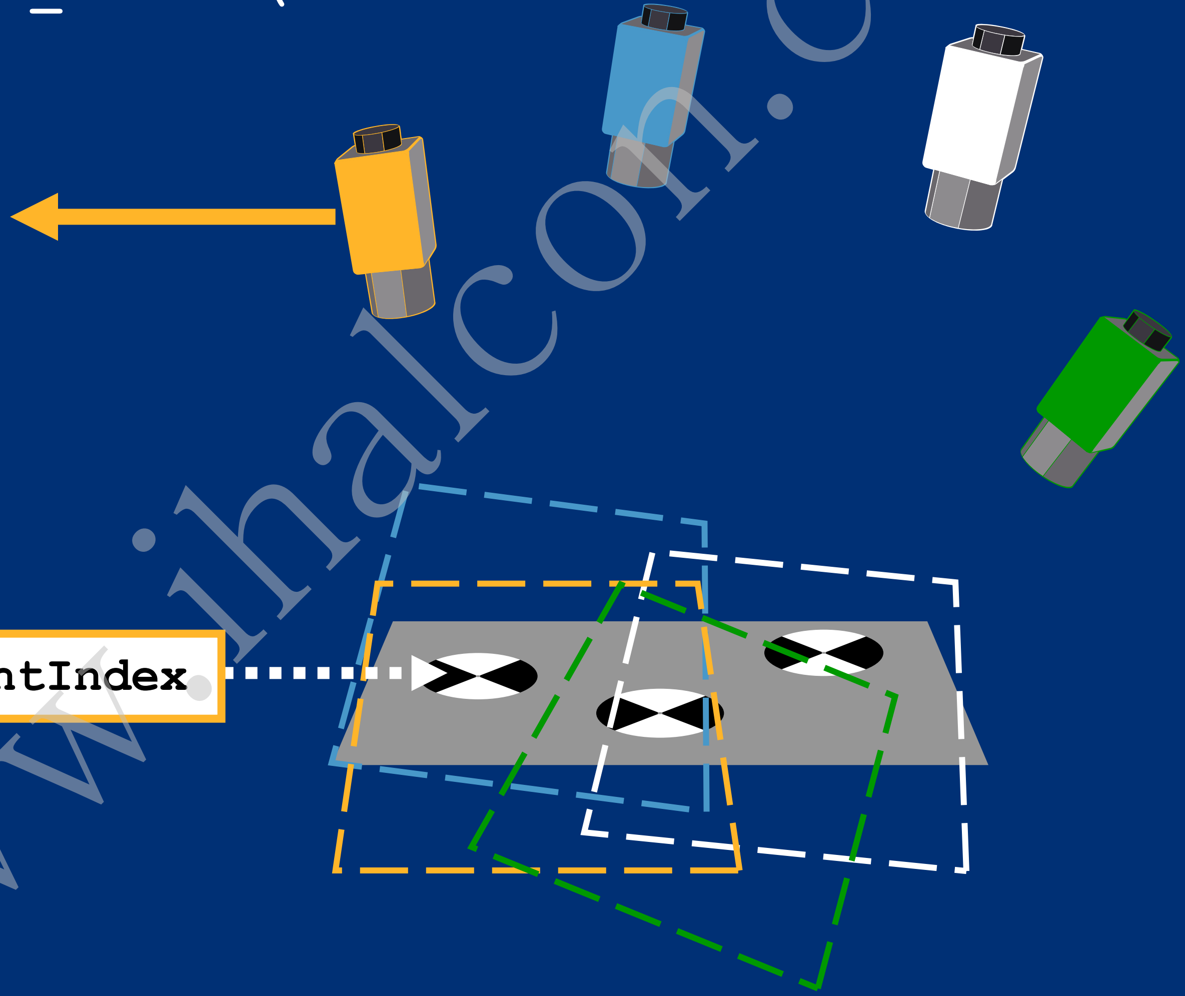


Reconstruct points with stereo



Determine 3D point coordinates with `reconstruct_points_stereo`

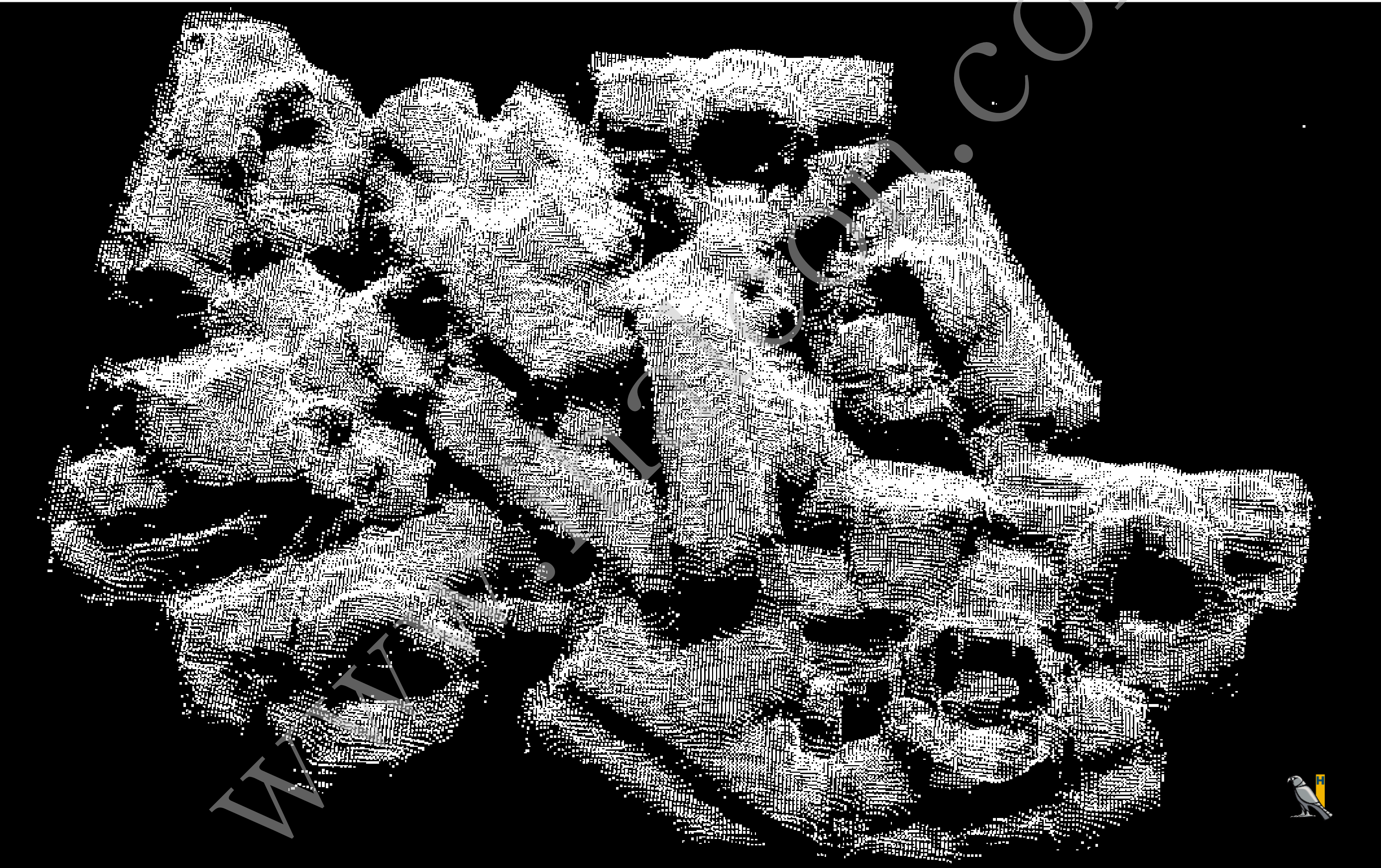
```
reconstruct_points_stereo (::
▶ StereoModelID,
▶ Row, Column,
▶ CovIP,
▶ CameraIndex,
▶ PointIndex :
◀ X, Y, Z,
◀ CovWP,
◀ PointIndexOut)
```



Row, Column, PointIndex

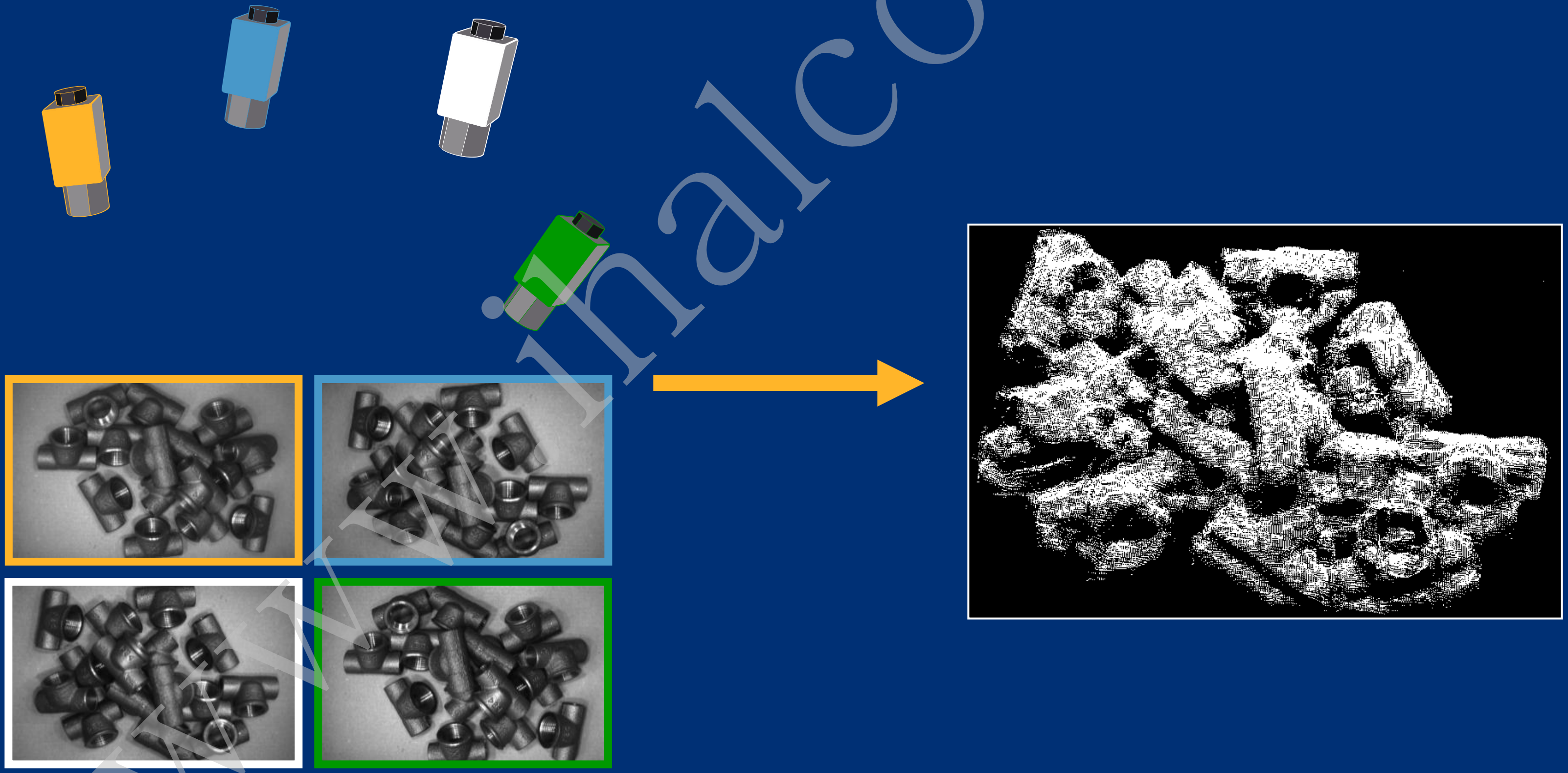
Reconstruct 3D scenes from multiple images

Reconstruct 3D scenes from multiple images



Reconstruct 3D scenes from multiple images with reconstruct_surface_stereo

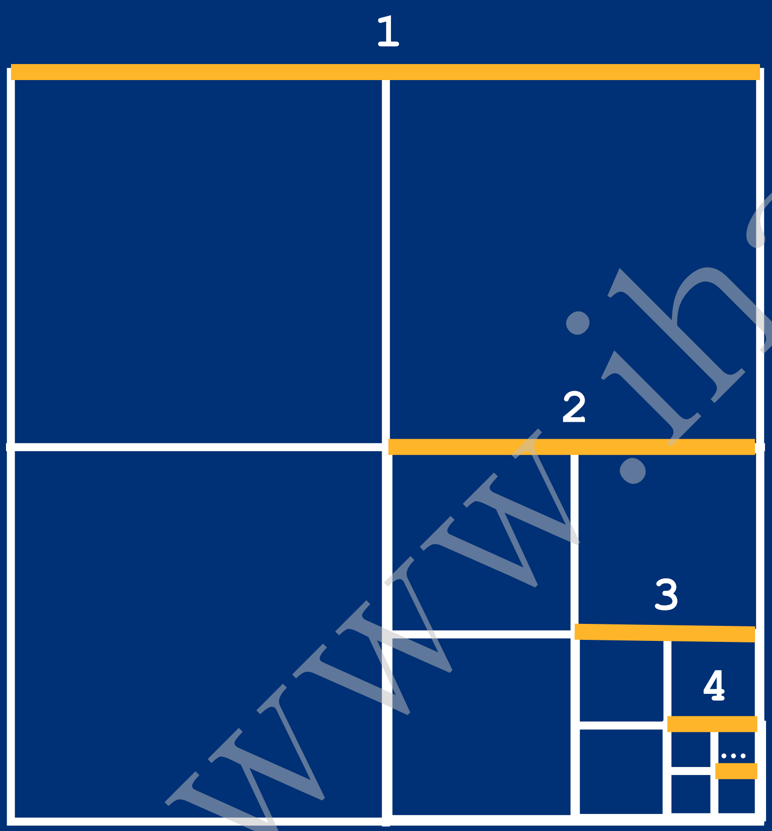
```
reconstruct_surface_stereo (
▶ Images ::
▶ StereoModelID :
◀ ObjectModel3D)
```

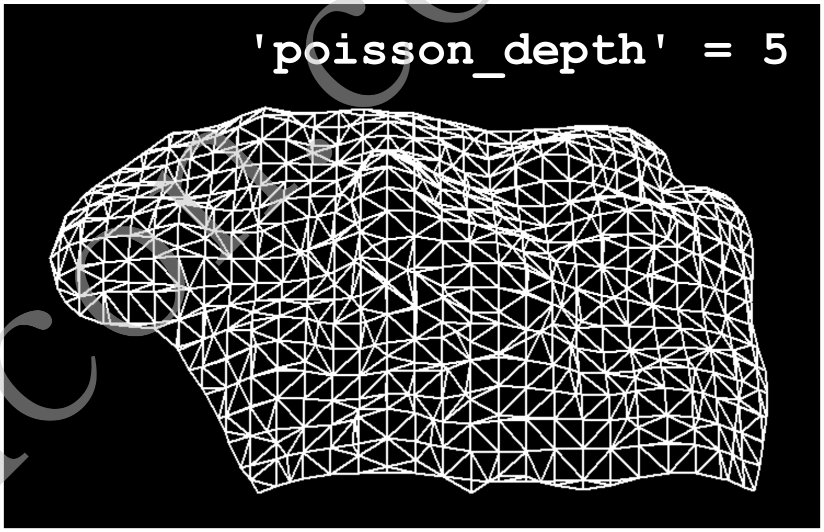


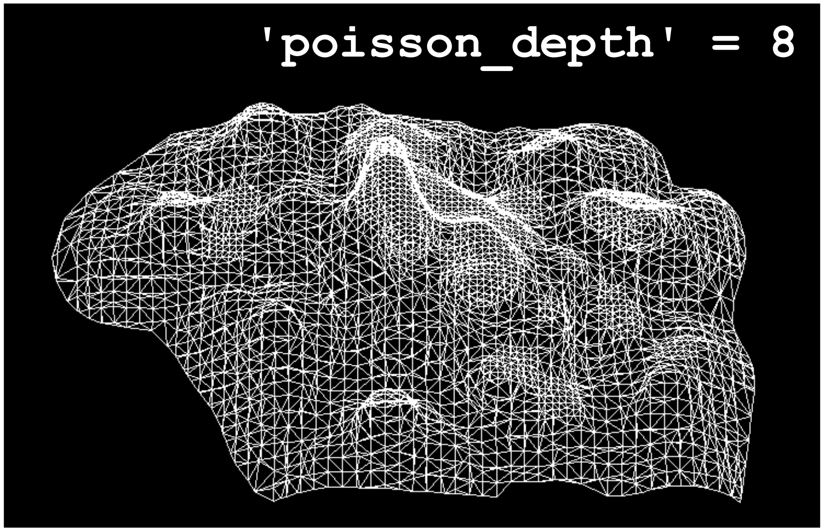
reconstruct_surface_stereo optionally returns a meshed surface

'point_meshing' = 'poisson'

'poisson_depth'

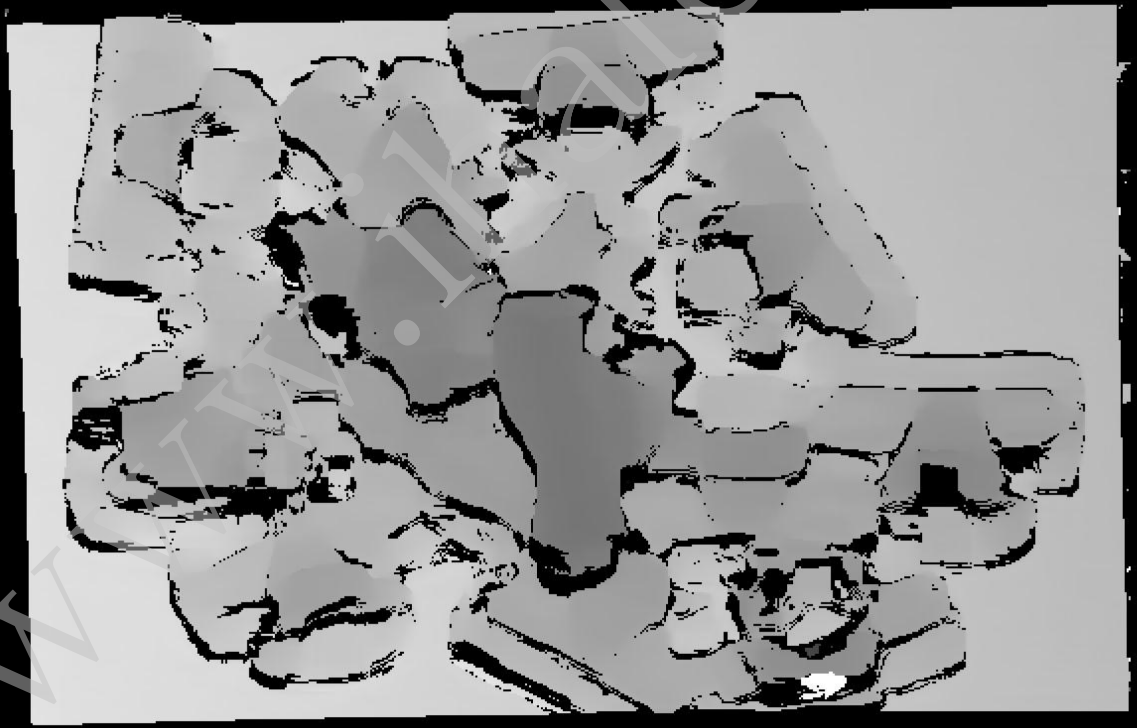






Additional data is stored in the stereo model if switched to persistence mode

```
set_stereo_model_param (StereoModelID, 'persistence', 1)
reconstruct_surface_stereo ( … )
get_stereo_model_object ( …, 'disparity_image')
                             'score_image'
                             '*_image_rect'
```

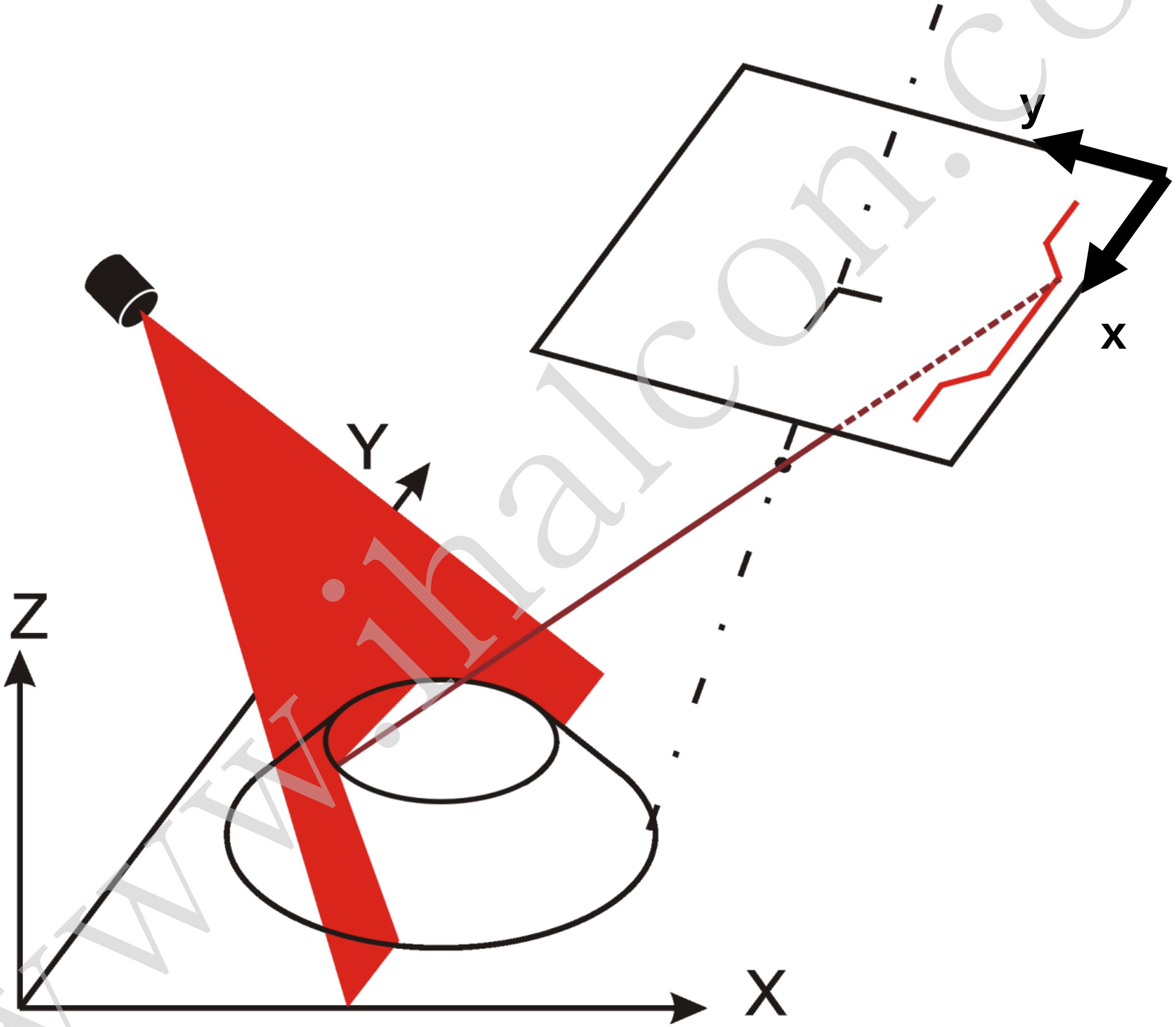


HALCON

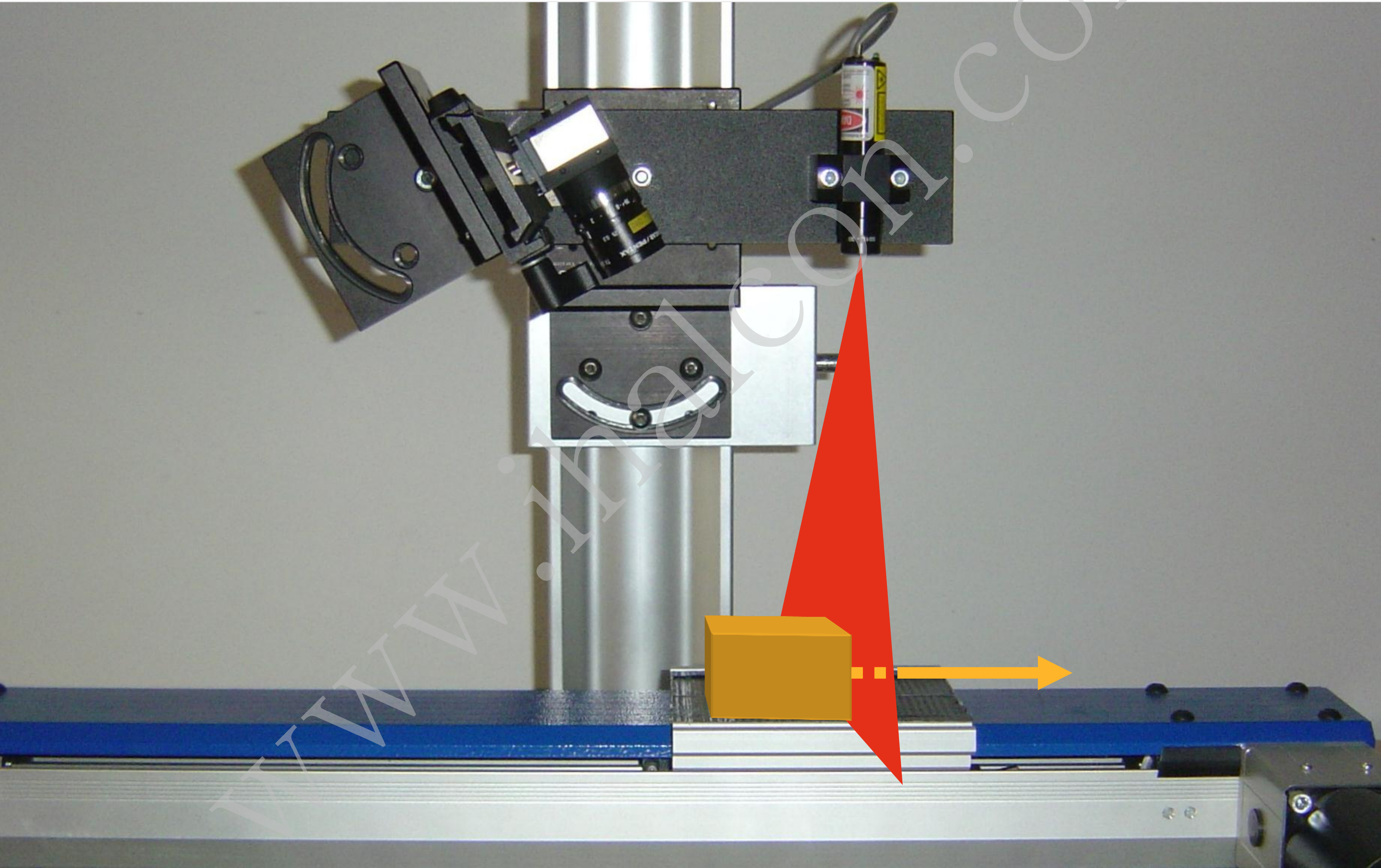
the Power of Machine Vision

Sheet-of-Light 3D Measurement

HALCON supports sheet-of-light measuring



Typical Setup



Alternative Setups

Stereo camera with laser projector

- The Laser line can be used to solve the correspondence problem (e.g. when the sample is lacking texture)
- In order to enhance the completeness of the measurement (e.g. reduce occluded areas)

Cameras with multiple line projector

- Parallelization (simultaneous acquisition of multiple profiles)
- Possible mismatch of the laser lines! (depending on the object geometry)

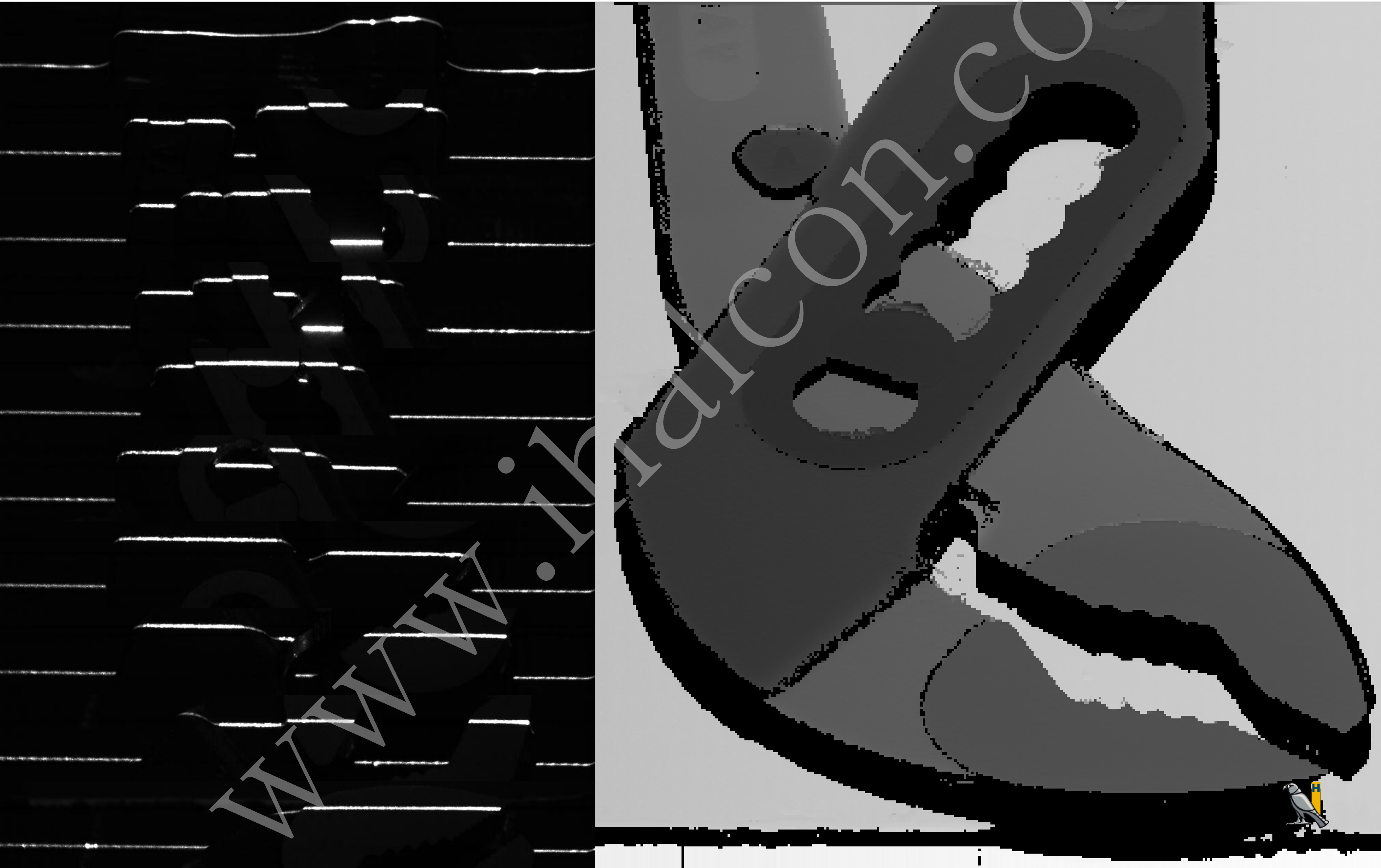
Fixed sensor, moving sample

- Online measurement or quality assessment (assembly line)

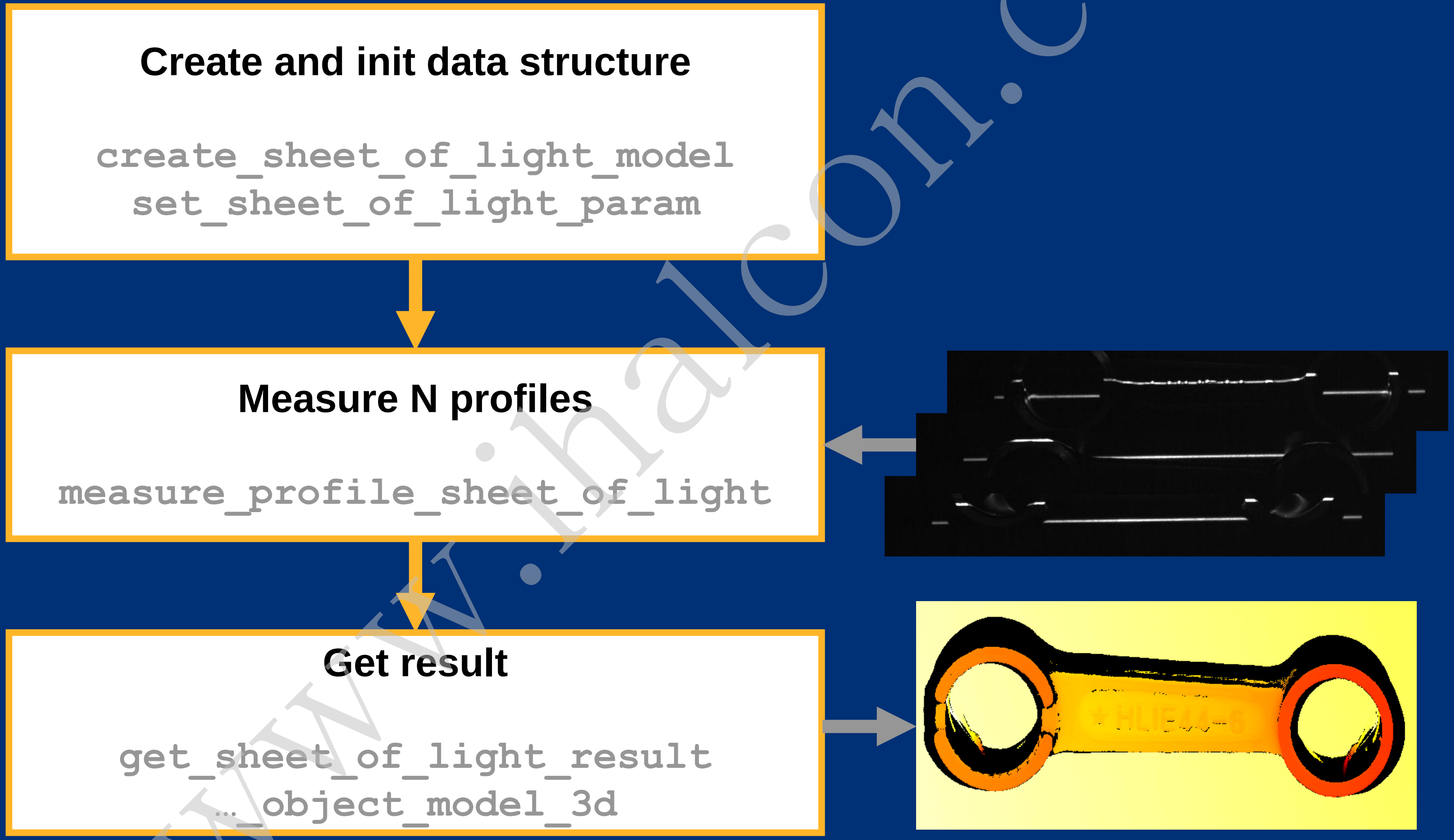
Fixed sample, moving sensor

- Hand held sensors (and optical tracking of the sensor)
- Robot-guided sensor, CMM-guided sensor or sensor mounted on a measuring arm

The surface measurement is performed by recording successive profiles



Perform a measurement in three steps



Create and init data structure

```
create_sheet_of_light_model
set_sheet_of_light_param
```

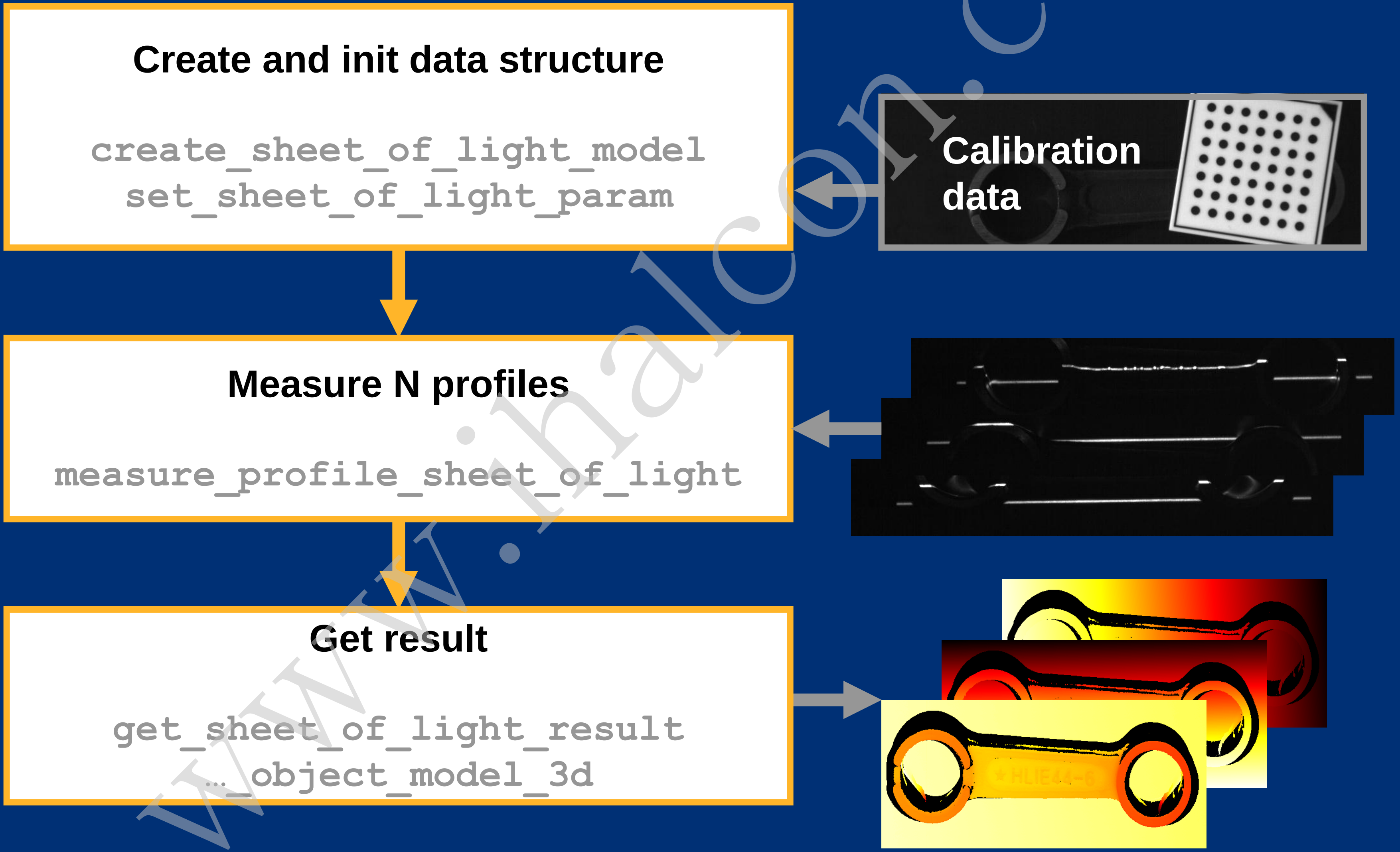
Measure N profiles

```
measure_profile_sheet_of_light
```

Get result

```
get_sheet_of_light_result
..._object_model_3d
```

Perform a measurement in three steps



Create and init data structure

```
create_sheet_of_light_model
set_sheet_of_light_param
```

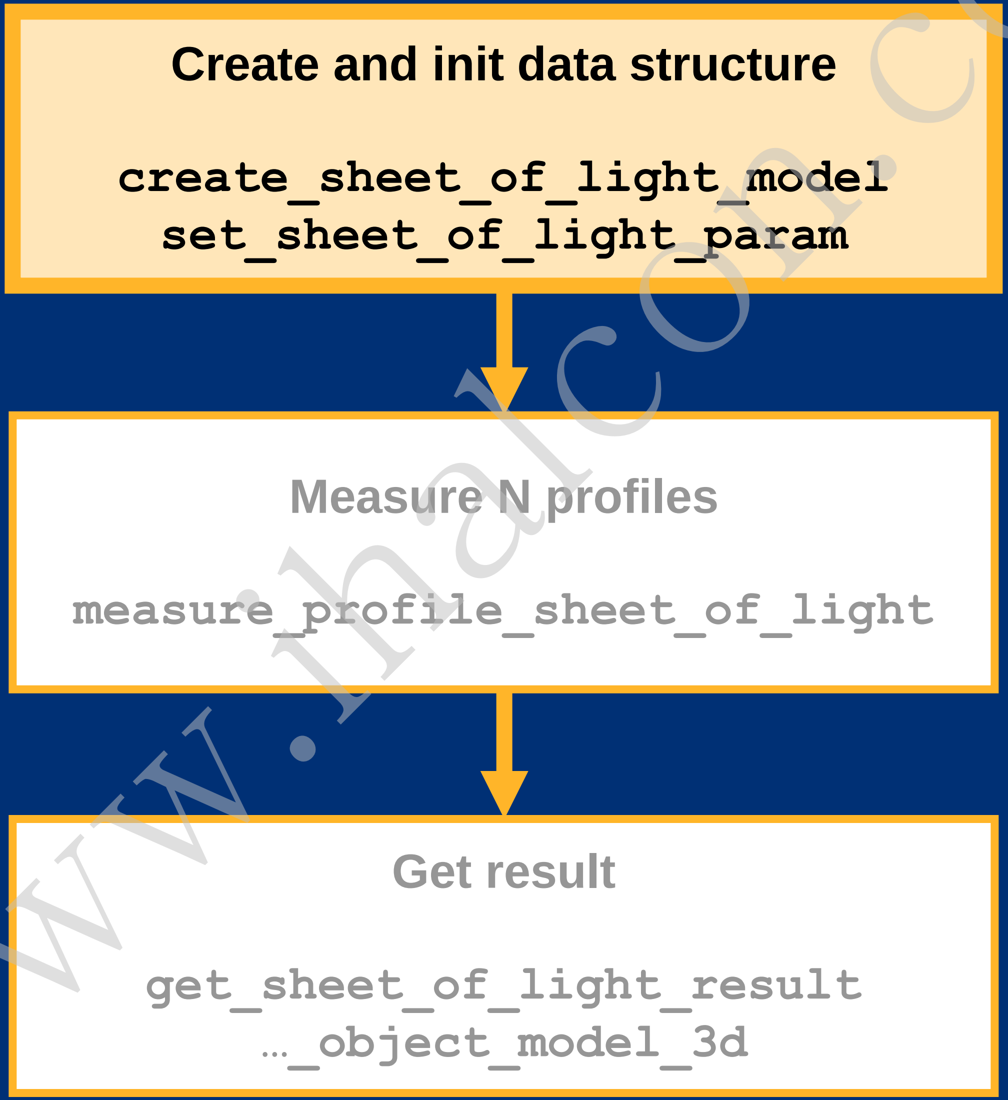
Measure N profiles

```
measure_profile_sheet_of_light
```

Get result

```
get_sheet_of_light_result
..._object_model_3d
```

First, create a sheet-of-light model



Create and init data structure

```
create_sheet_of_light_model
set_sheet_of_light_param
```

Measure N profiles

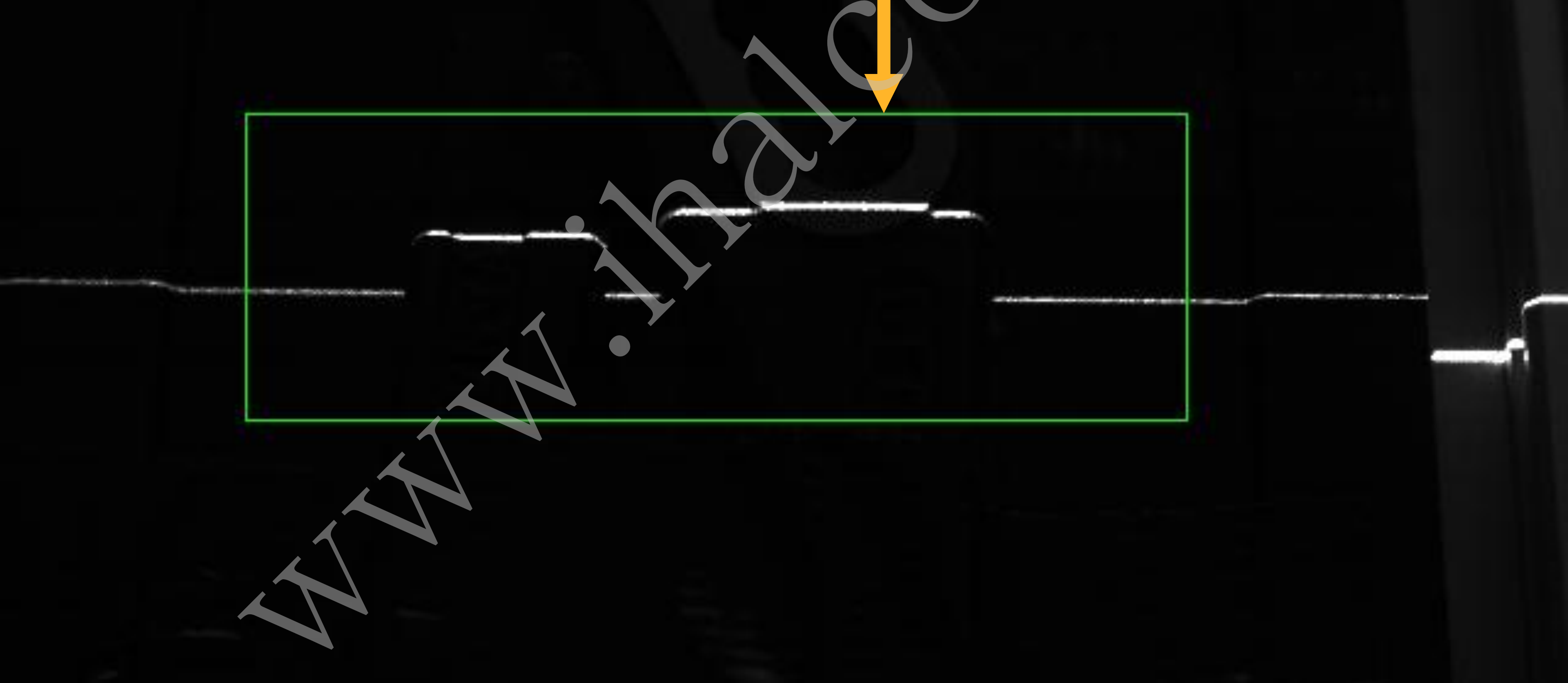
```
measure_profile_sheet_of_light
```

Get result

```
get_sheet_of_light_result
…_object_model_3d
```

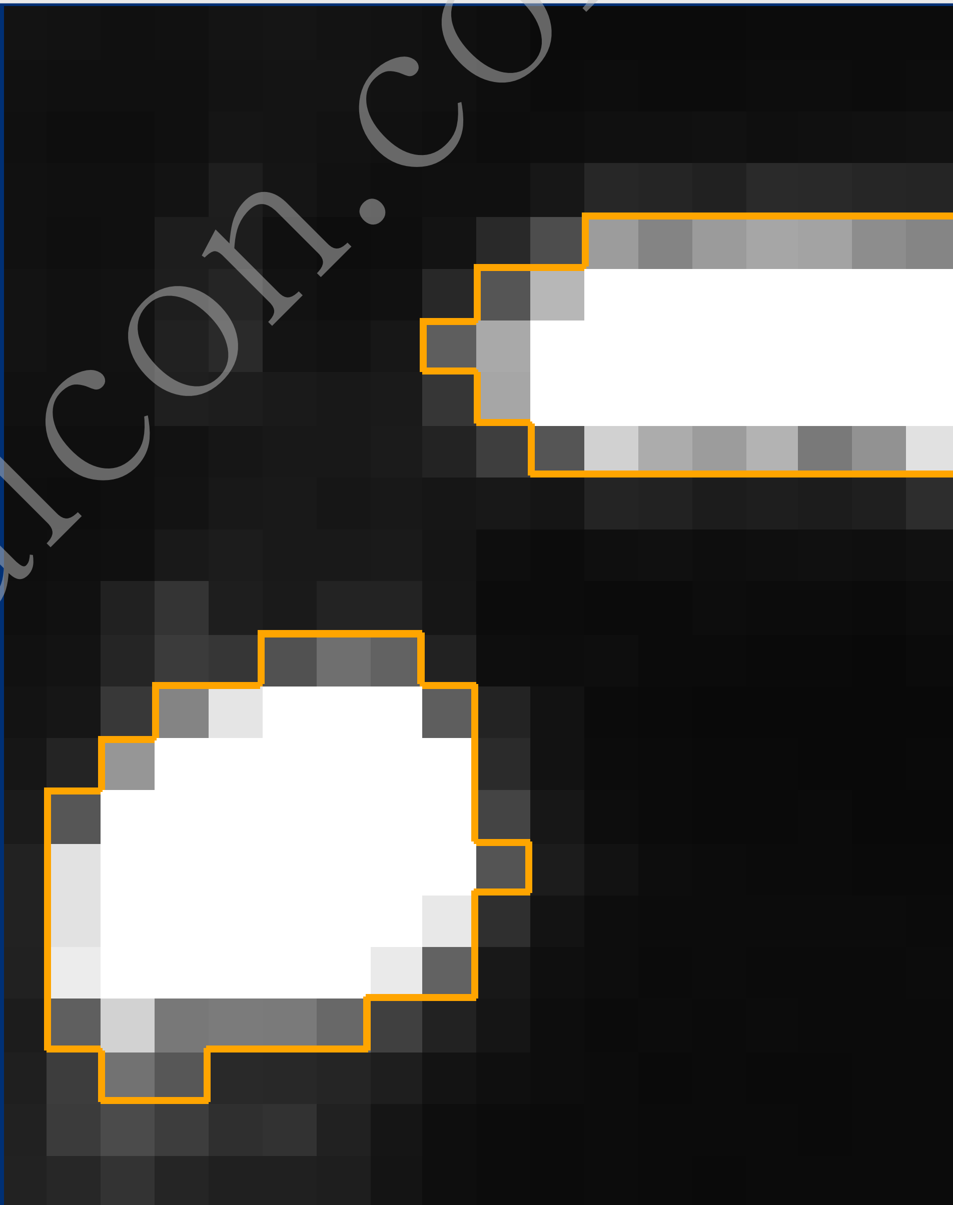
Create the data structure

```
create_sheet_of_light_model (
   ► ProfileRegion::
   ► GenParamNames,
   ► GenParamValues:
   ◄ SheetOfLightModelID)
```



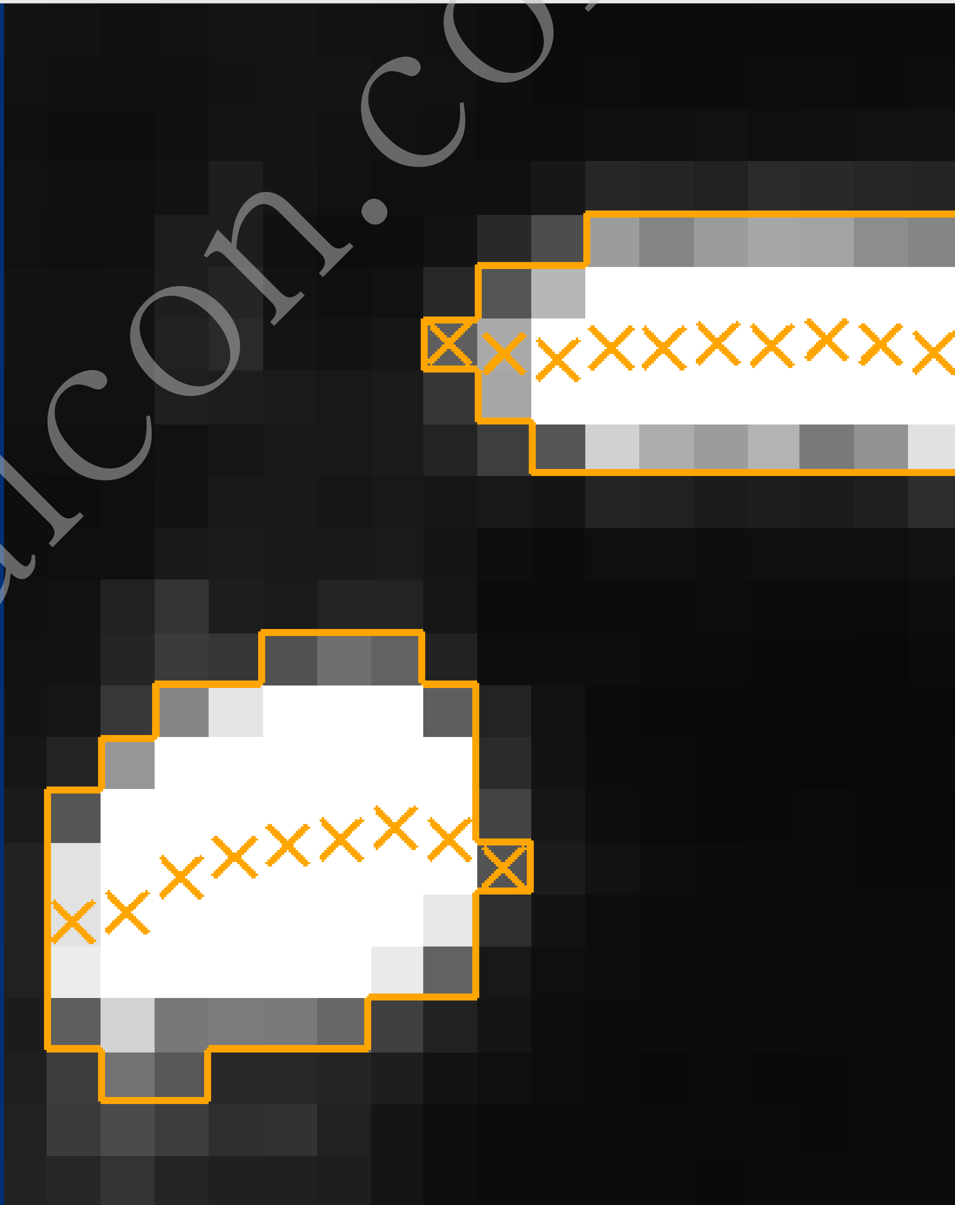
Specify the threshold for line detection

```
'min_gray' = 100
```



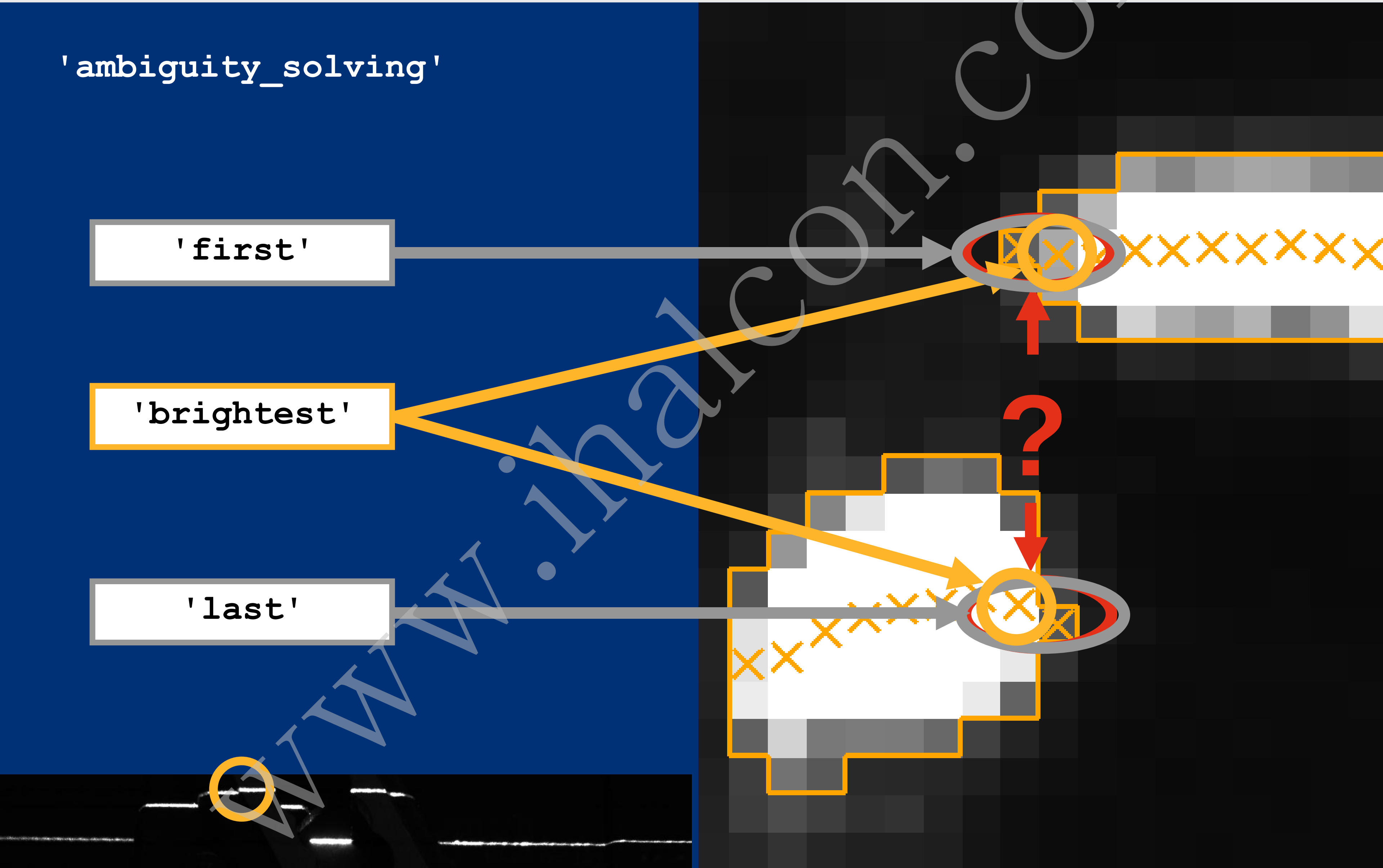
Specify how to determine the position of the profile

```
'method' = 'center_of_gravity'
```



Specify how ambiguities are handled

'ambiguity_solving'



'first'

'brightest'

'last'

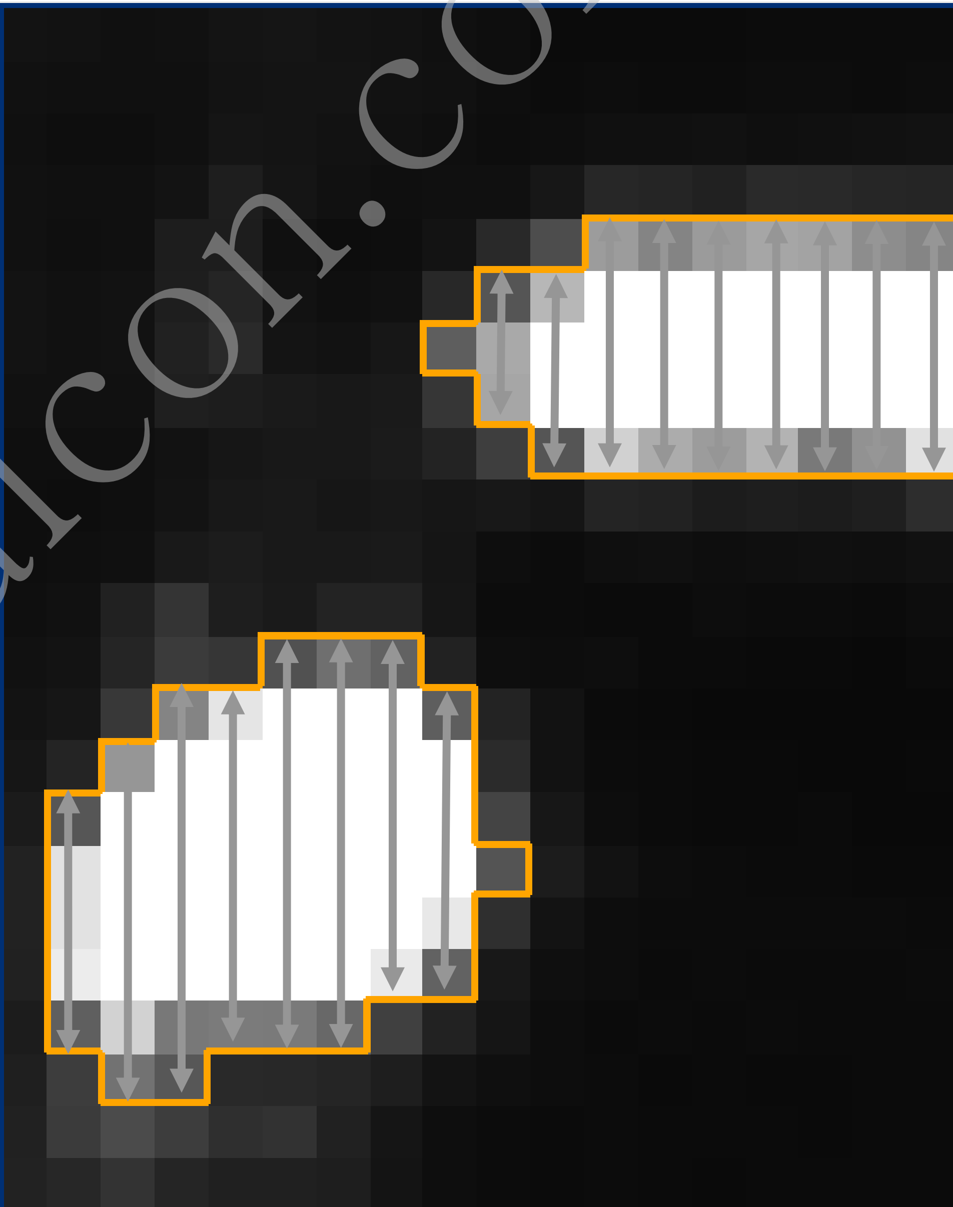
Specify a quality measure for the measurement

'score_type'

'none'

'width'

'intensity'



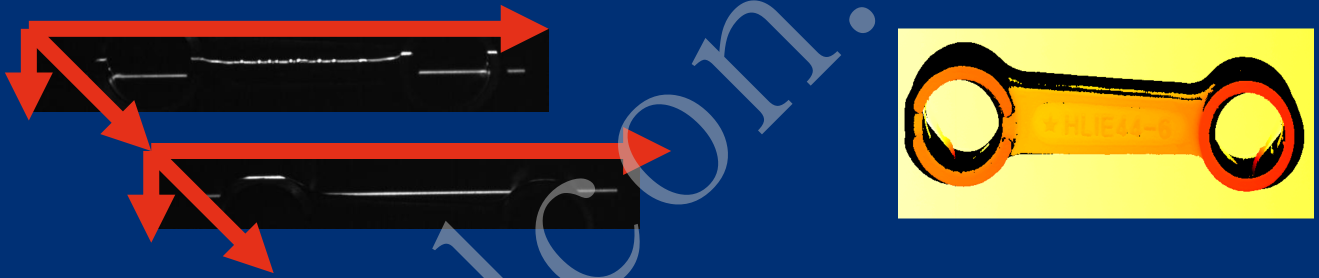
Specify the number of profiles

```
'num_profiles' = 512
```

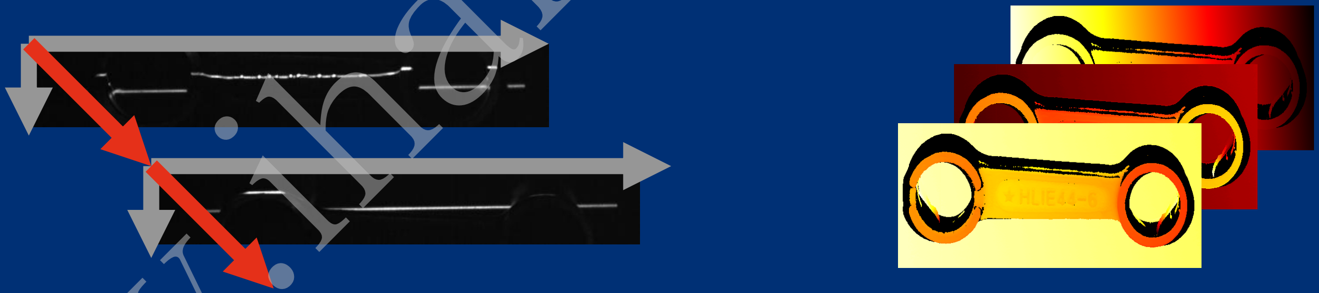
Specify the calibration type

'calibration'

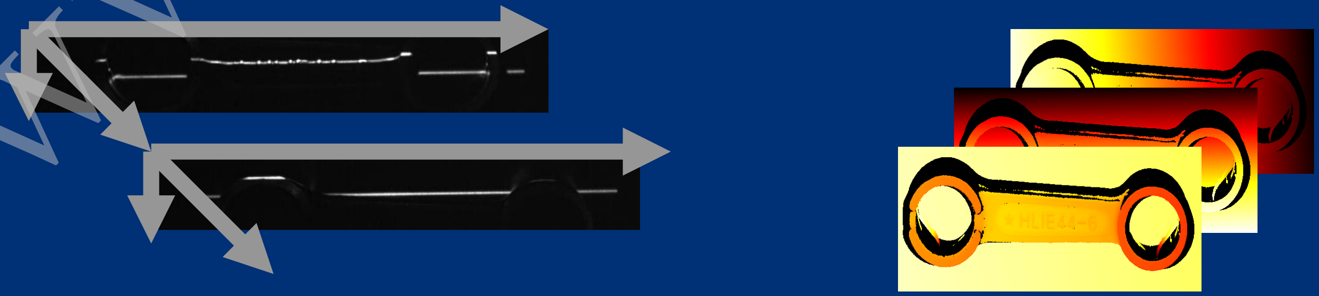
'none'



'xz'



'xyz'



All parameters can be set with set_sheet_of_light_param

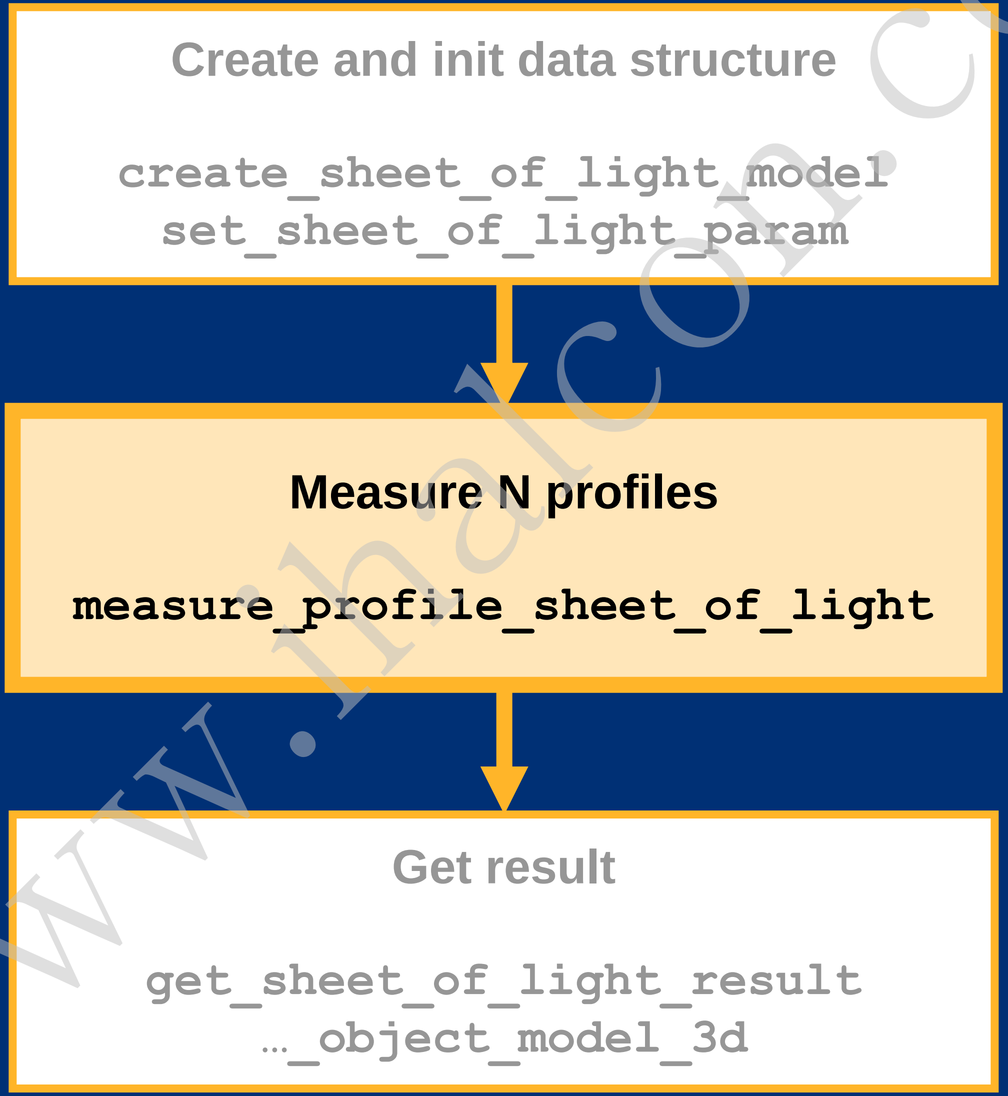
```
set_sheet_of_light_param (::
   ▶ SheetOfLightModelID,
   ▶ GenParamNames,
   ▶ GenParamValues:)
```

```
'method'
'ambiguity_solving'
'score_type'
'num_profiles'
'min_gray'

'calibration'
'scale'

'camera_parameter'
'camera_pose'
'lightplane_pose'
'movement_pose'
```

Measure profiles



Create and init data structure

`create_sheet_of_light_model`
`set_sheet_of_light_param`

Measure N profiles

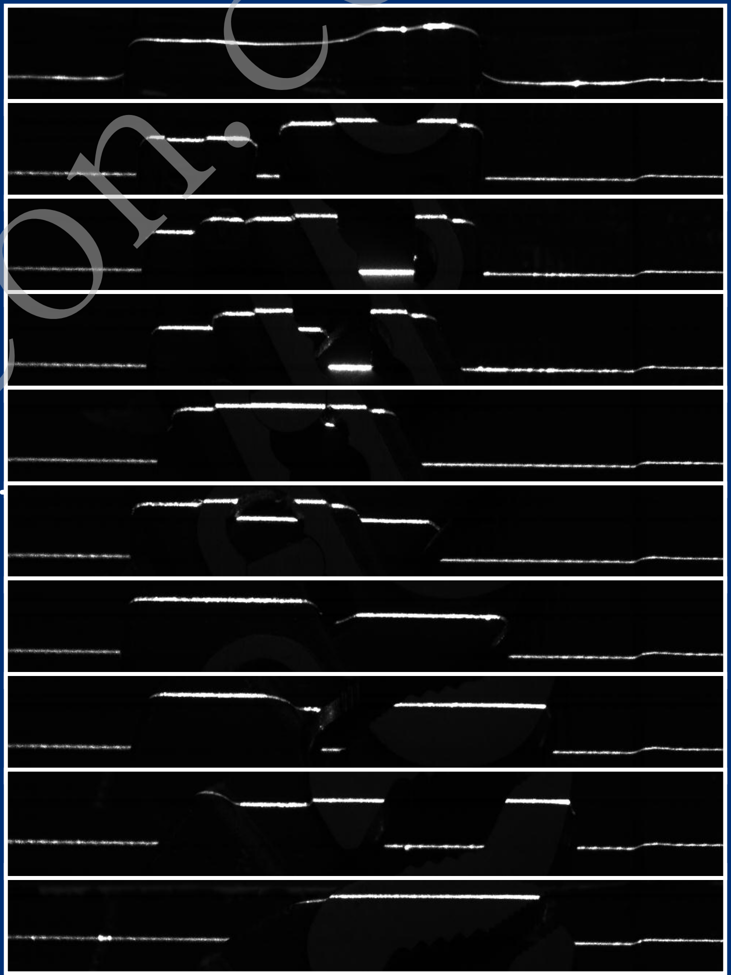
`measure_profile_sheet_of_light`

Get result

`get_sheet_of_light_result`
`…_object_model_3d`

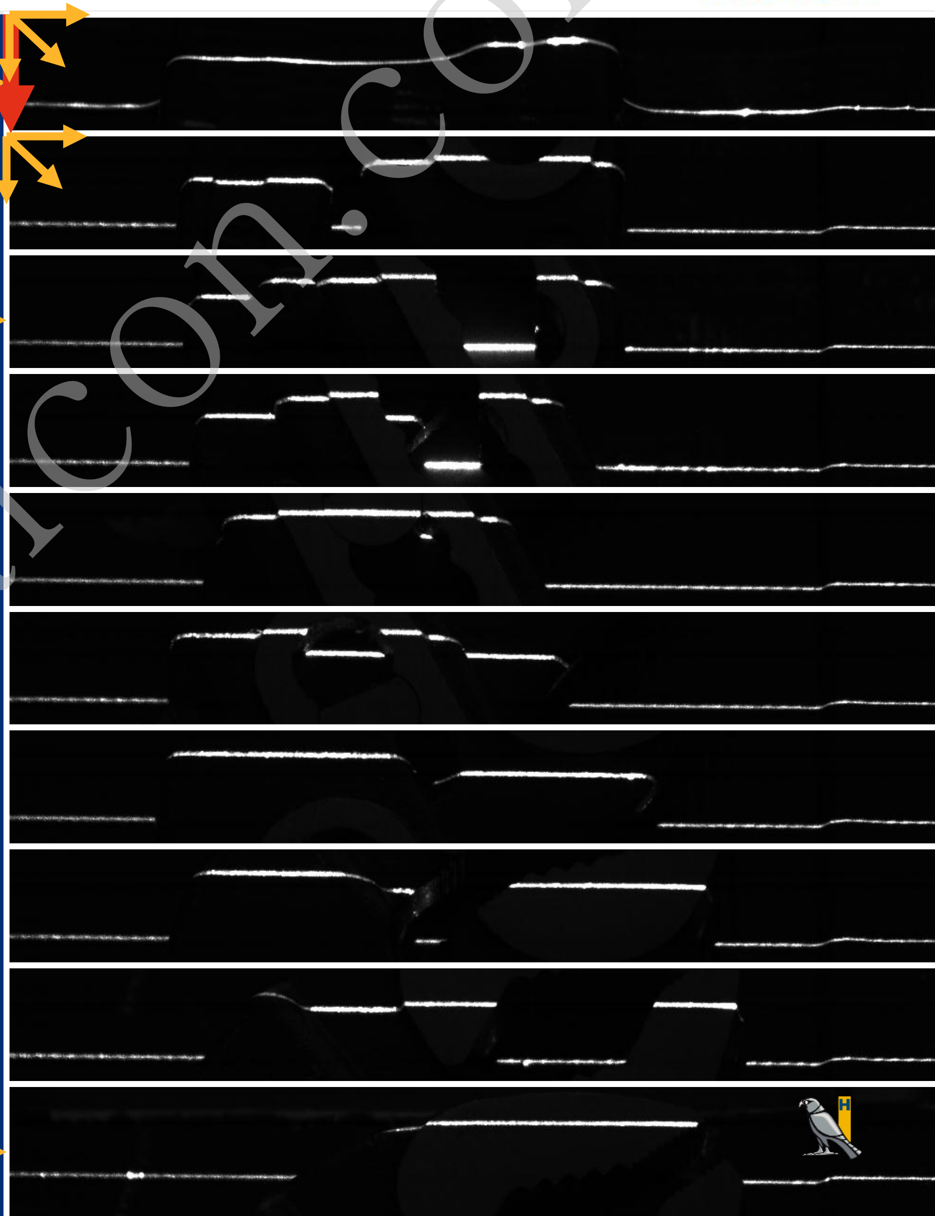
Measure as many profiles as you require

```
for 1 to n
        grab_image
        measure_profile_sheet_of_light
endfor
```

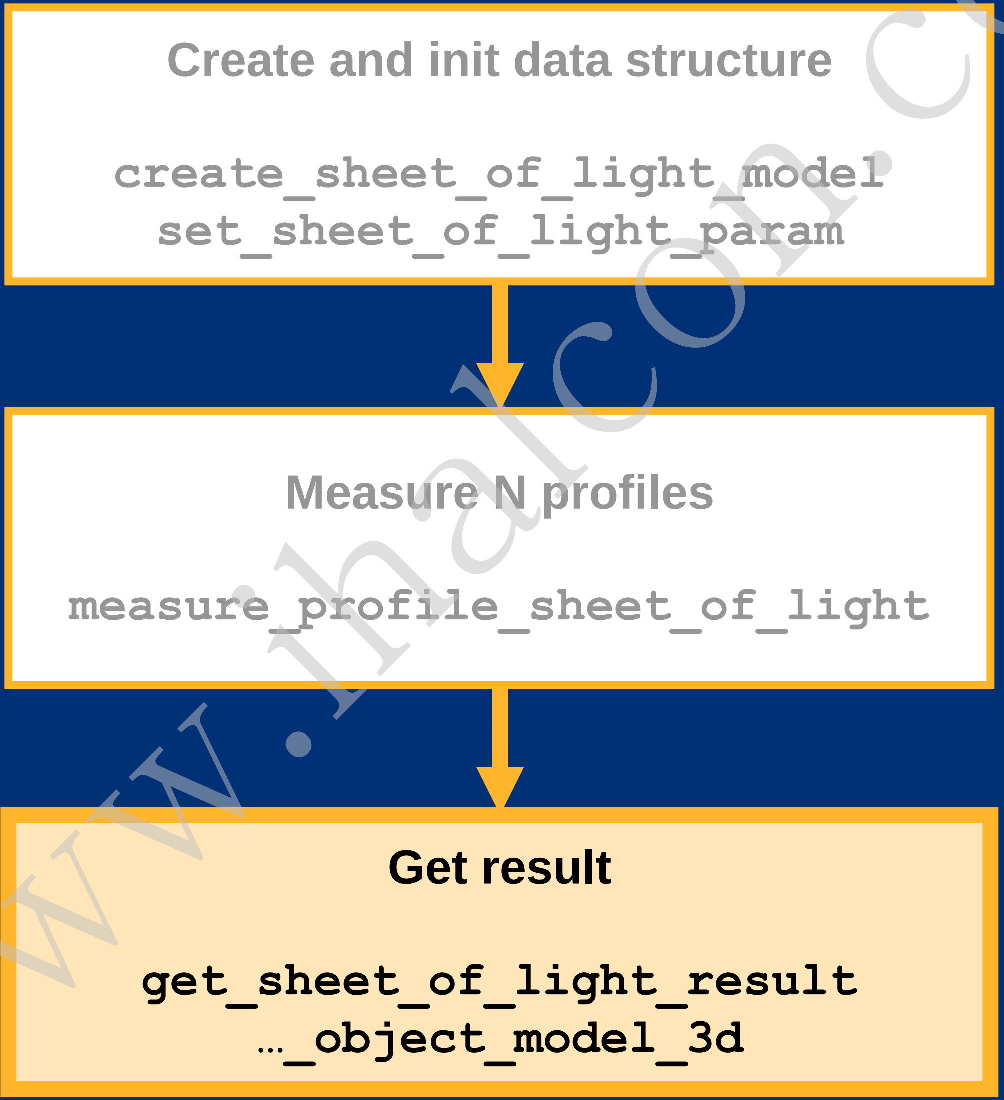


Measure as many profiles as you require

```
measure_profile_sheet_of_light(
    ▶ ProfileImage::
    ▶ SheetOfLightModelID,
    ▶ MovementPose:)
```



Get the results



Create and init data structure

```
create_sheet_of_light_model
set_sheet_of_light_param
```

Measure N profiles

```
measure_profile_sheet_of_light
```

Get result

```
get_sheet_of_light_result
…_object_model_3d
```

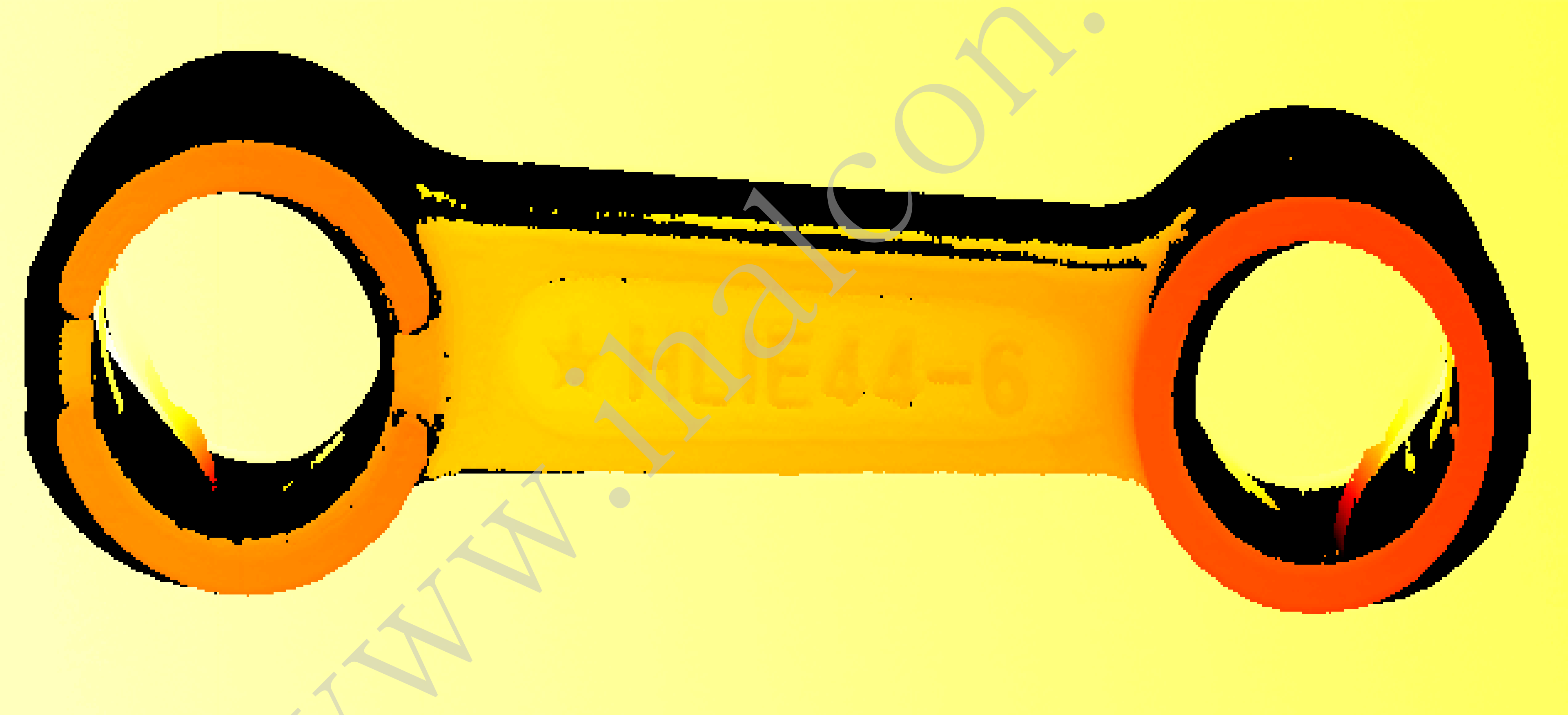
Get the results with `get_sheet_of_light_results`

```
get_sheet_of_light_results (:
   ◀ ResultValue:
   ▶ SheetOfLightModelID,
   ▶ ResultName:)
```

```
'disparity'
'x'
'y'
'z'
'score'
```

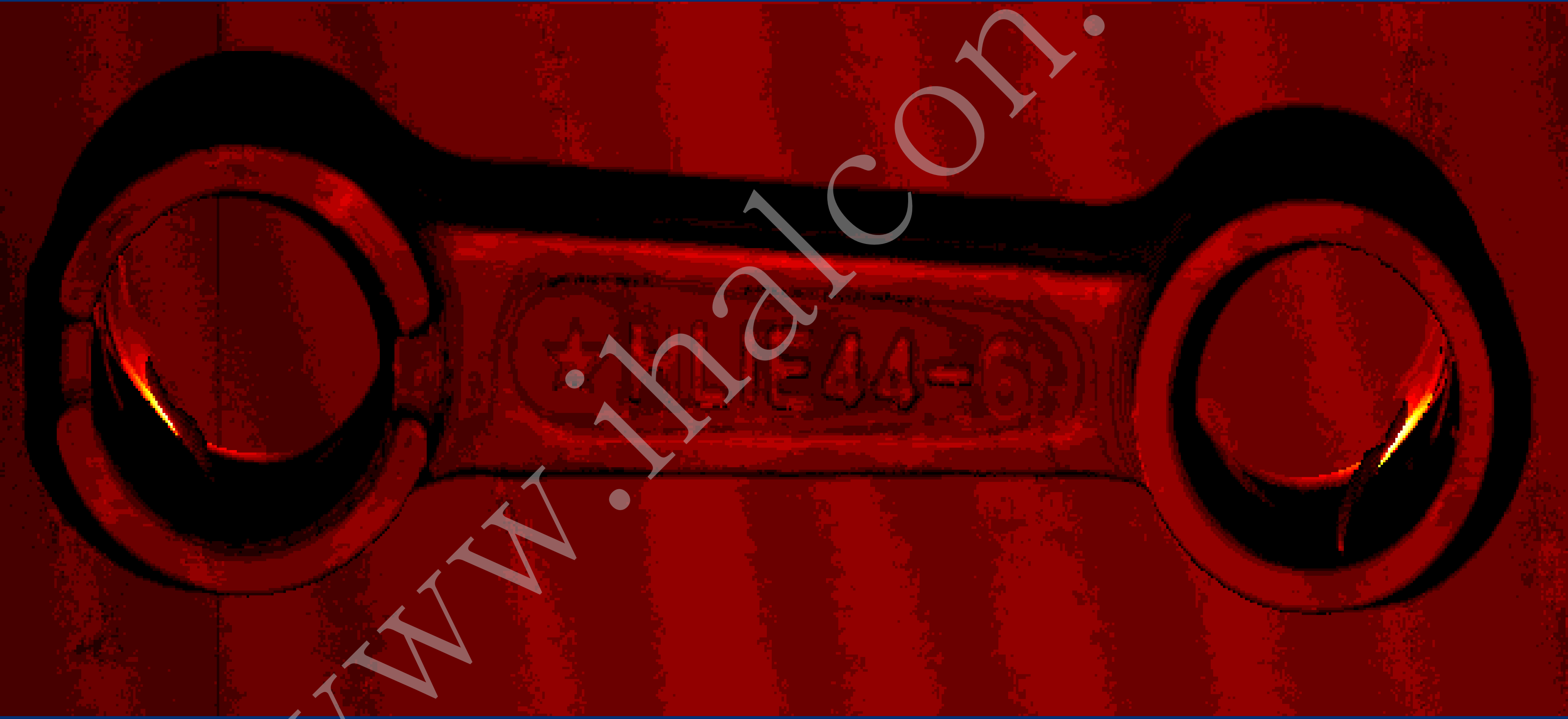
Get the results with `get_sheet_of_light_results`

`'disparity'`



Get the results with `get_sheet_of_light_results`

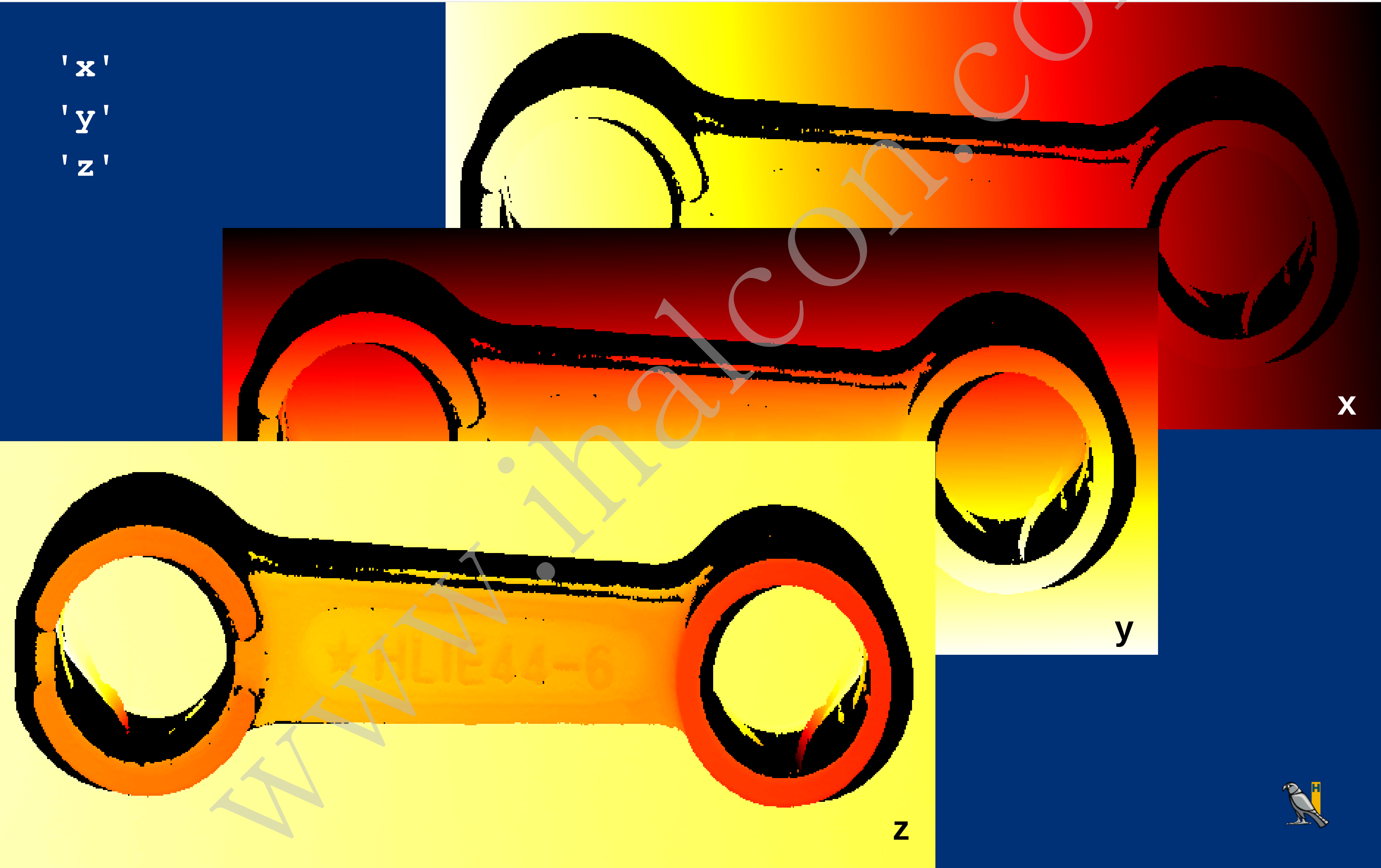
`'score'`



`'score_type' = 'width'`

Get the results with `get_sheet_of_light_results`

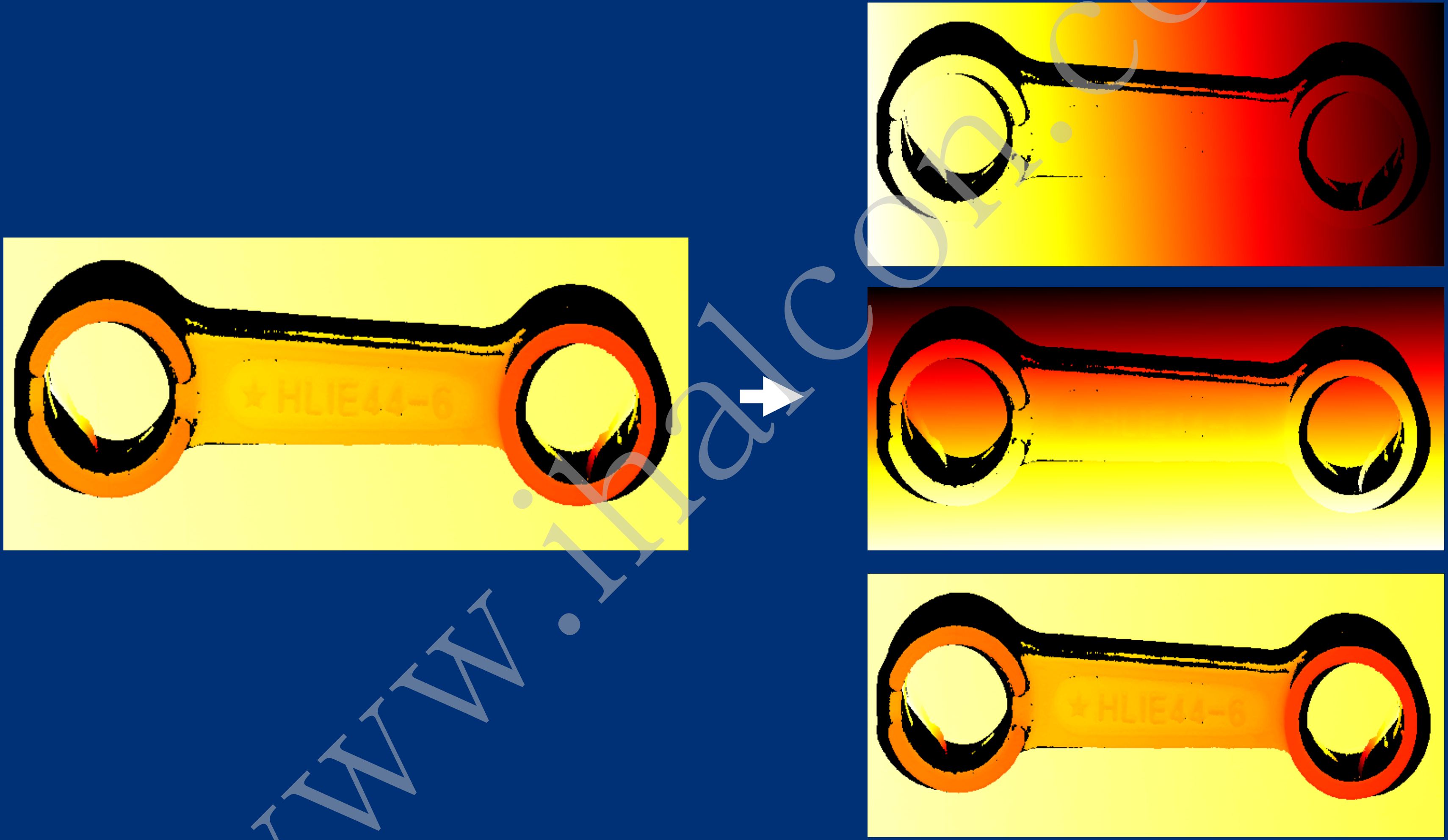
- `'x'`
- `'y'`
- `'z'`



Get the results with
get_sheet_of_light_results_object_model_3d

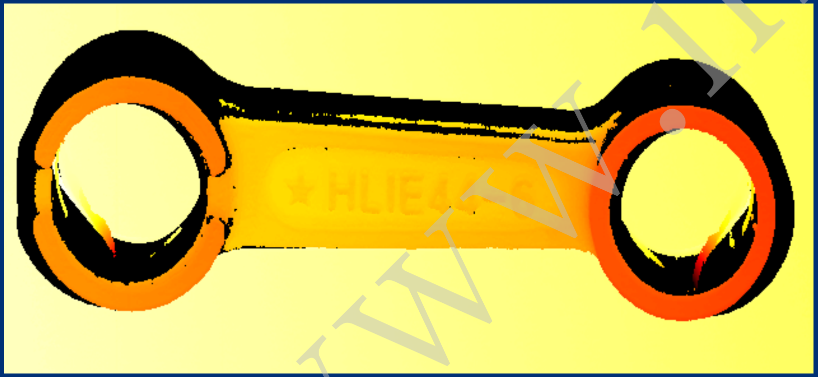
```
get_sheet_of_light_results_object_model_3d (: :
   ► SheetOfLightModelID :
   ◄ ObjectModel3DID:
   )
```

The calibration can also be performed after the measurement

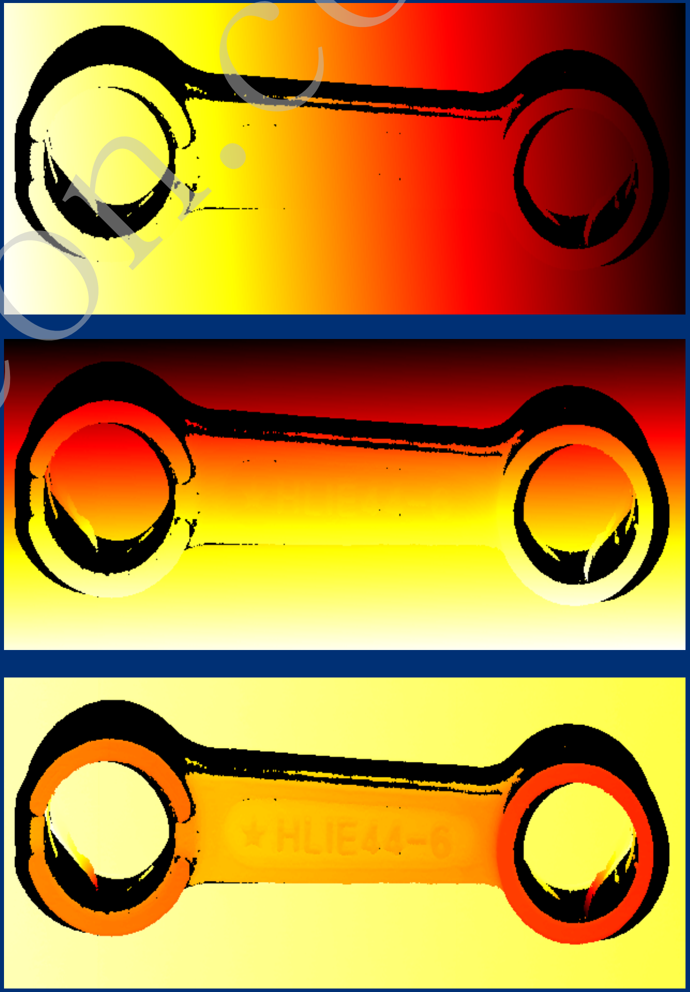


get_sheet_of_light_results apply_sheet_of_light_calibration

In addition, measurements performed with an alternative system can be calibrated

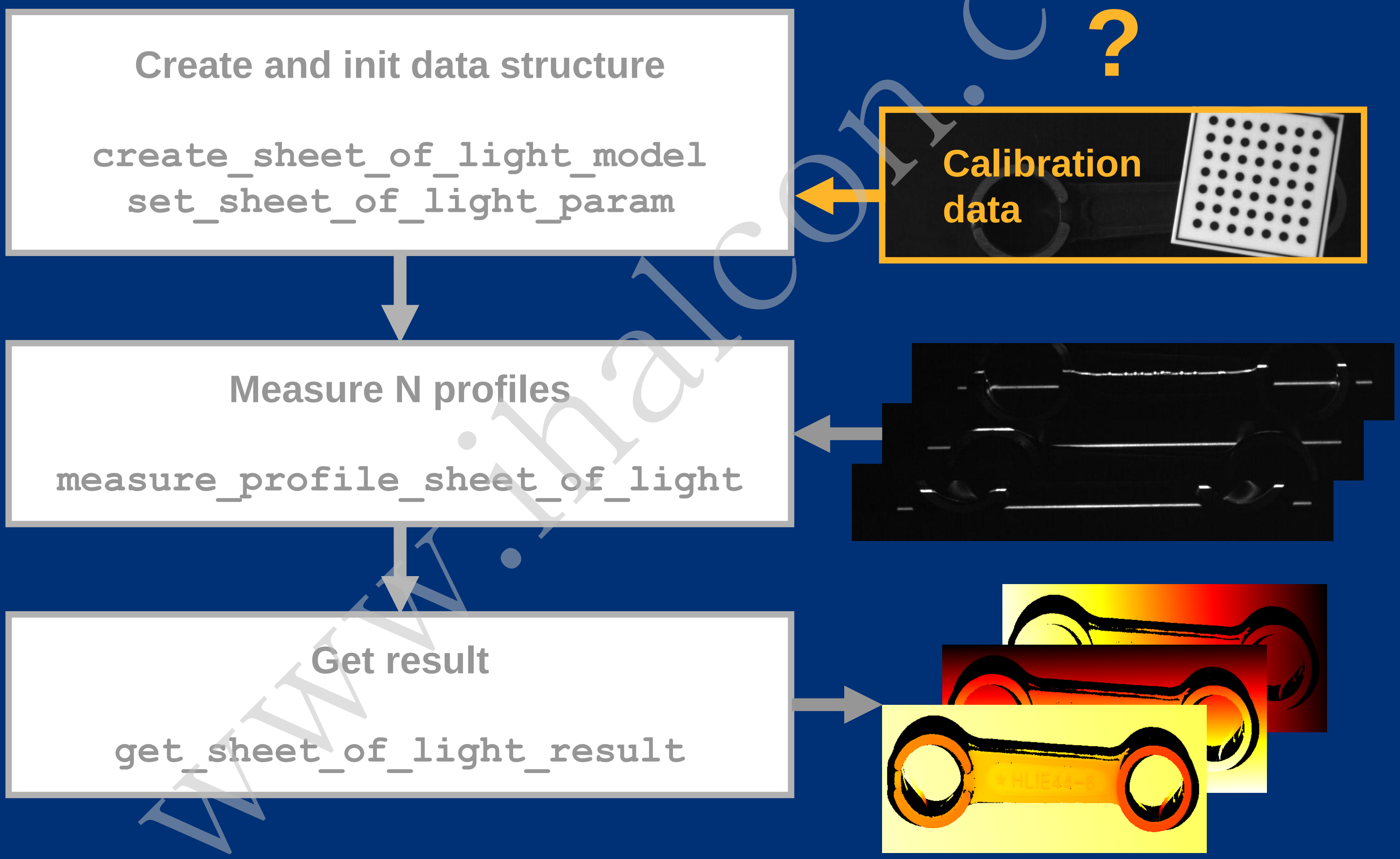


grab_image



apply_sheet_of_light_calibration

How to obtain the calibration parameters?



Create and init data structure

```
create_sheet_of_light_model
set_sheet_of_light_param
```

?

Calibration data

Measure N profiles

```
measure_profile_sheet_of_light
```

Get result

```
get_sheet_of_light_result
```

Which parameters must be provided by the calibration?

```
set_sheet_of_light_param (::
   ▶ SheetOfLightModelID,
   ▶ GenParamNames,
   ▶ GenParamValues:)
```

```
'method'
'ambiguity_solving'
'score_type'
'num_profiles`
'min_gray'

'calibration'
'scale'

'camera_parameter'
'camera_pose'
'lightplane_pose'
'movement_pose'
```

Depend on the requirements of the application

Provided by the camera calibration

Must be determined !

Must be determined !

Use simplified sheet-of-light model parameters in an “orthogonal” setup

```
create_sheet_of_light_model(…'calibration','offset_scale',…)
```

```
set_sheet_of_light_param (ModelID, 'scale_x', ScaleX)
set_sheet_of_light_param (ModelID, 'scale_y', ScaleY)
set_sheet_of_light_param (ModelID, 'scale_z', ScaleZ)
```

Geometrical description of the measurement system

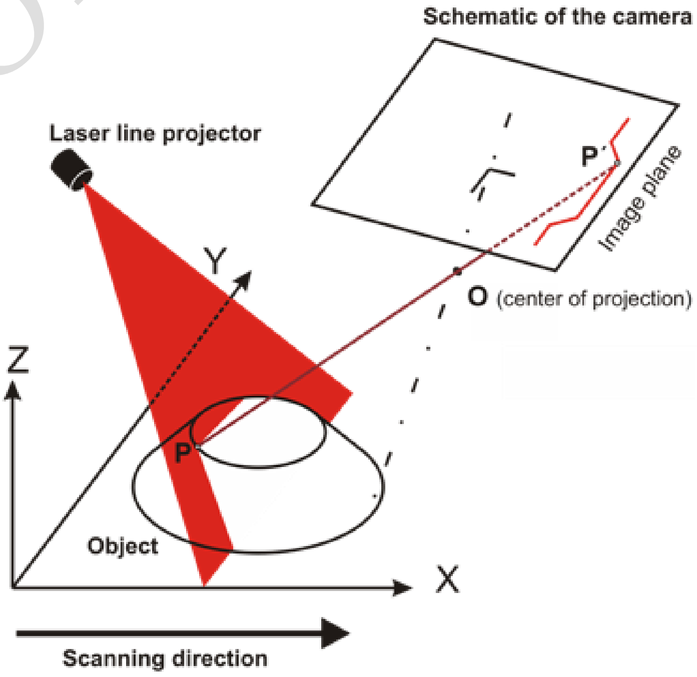
Unknown parameters

1. Internal parameters of the camera
2. Pose of the camera in the WCS
3. Orientation of the light plane in the WCS
4. Orientation of the scan vector in the WCS
5. Coordinates of P in the WCS

Known parameters

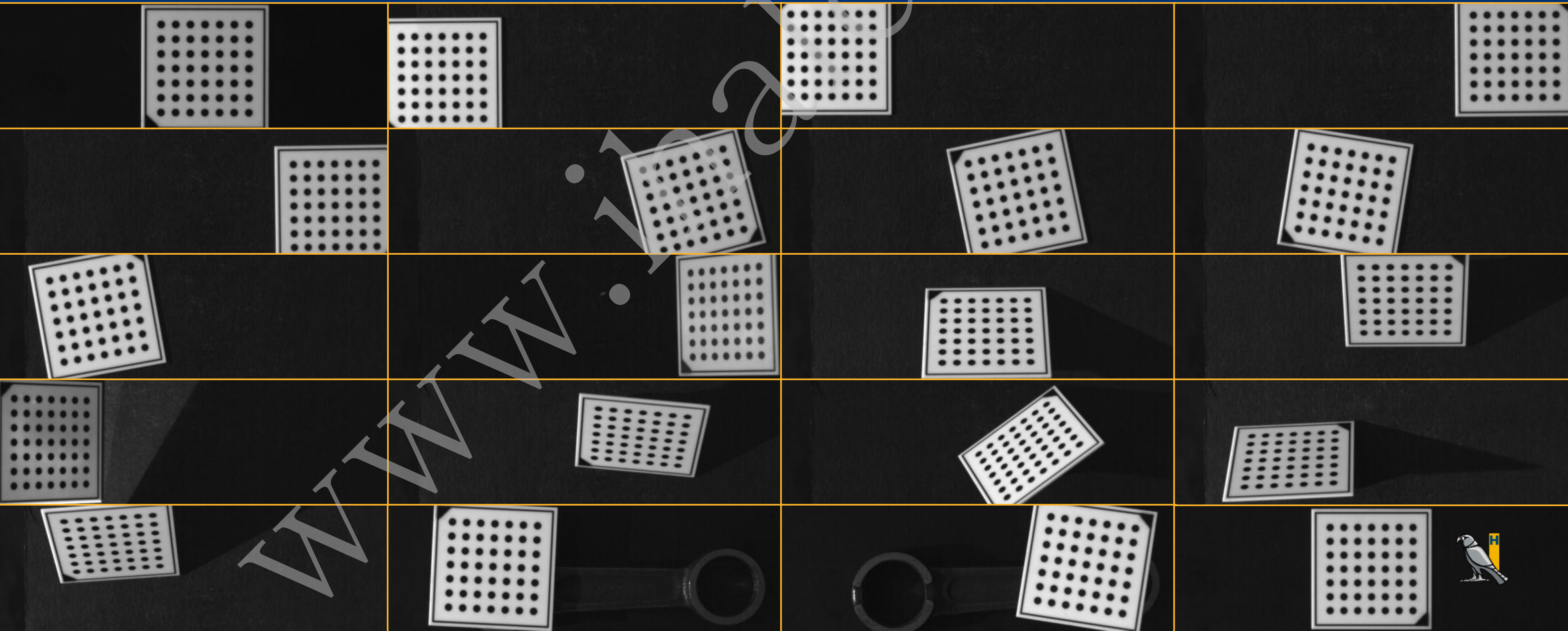
- Coordinates of the line points in the ICS

Standard approach: parameter sets 1 to 4 are system dependent and must be determined (calibrated)

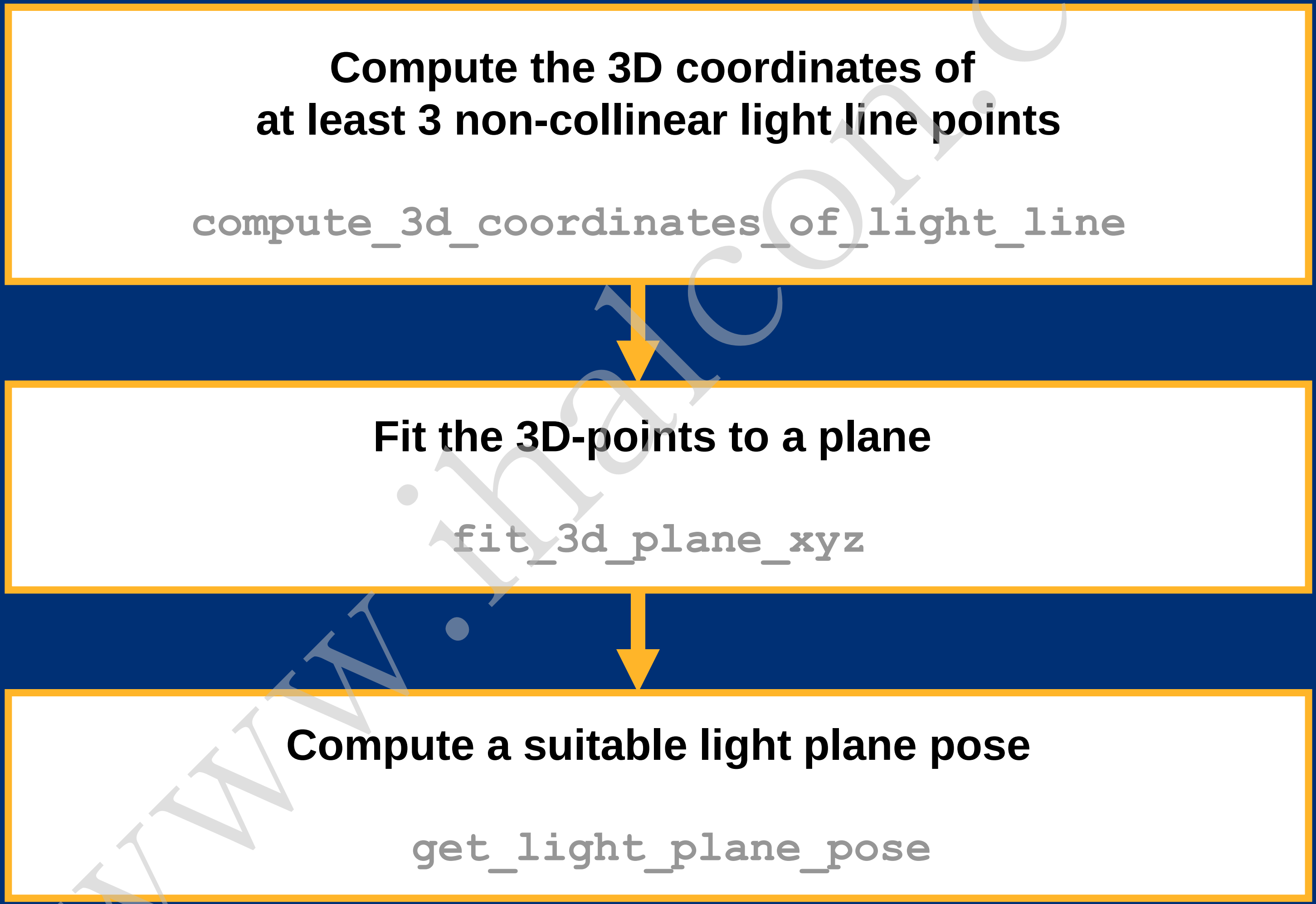


Perform the calibration of the camera as usual

- Perform a standard camera calibration
- Use a standard HALCON calibration plate)
- Acquire and use more than 8 images of the calibration plate
- The calibration of the camera provides the values of the parameters `'camera_parameter'` and `'camera_pose'`



The light plane pose is determined in three steps



Compute the 3D coordinates of
at least 3 non-collinear light line points

```
compute_3d_coordinates_of_light_line
```

Fit the 3D-points to a plane

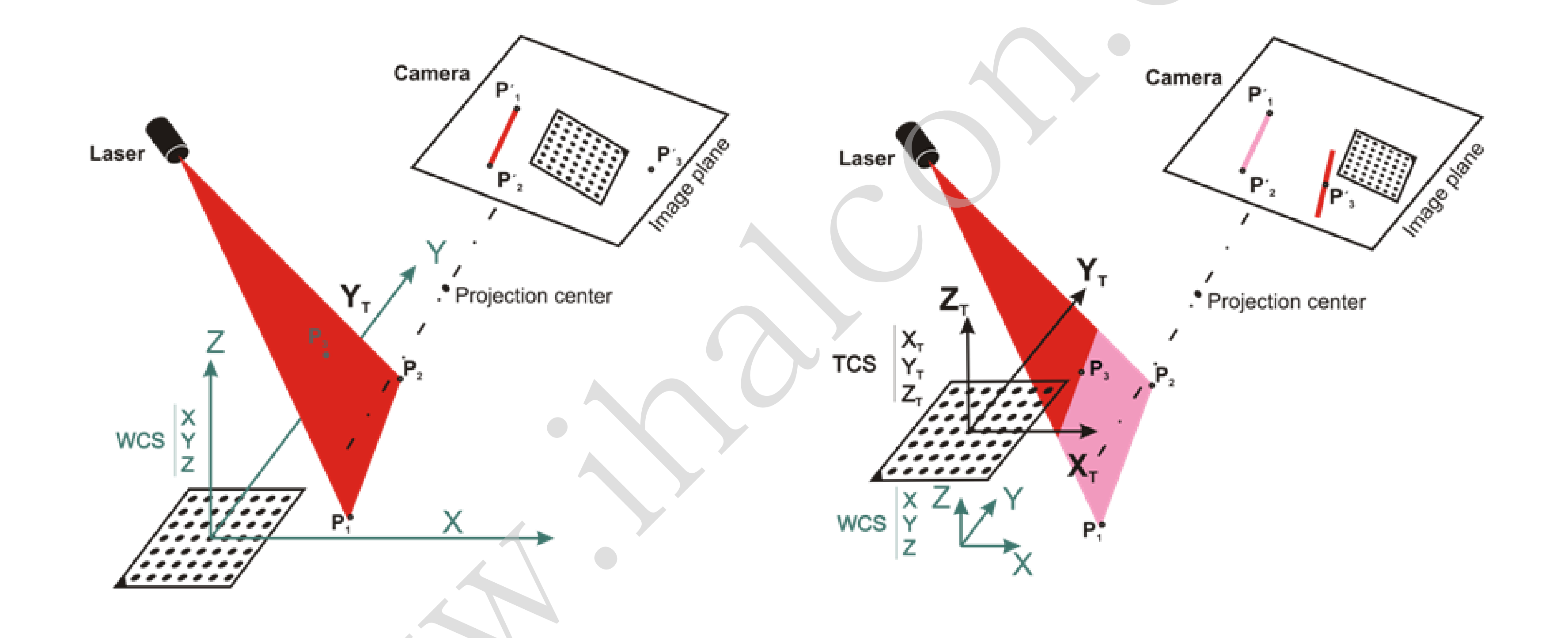
```
fit_3d_plane_xyz
```

Compute a suitable light plane pose

```
get_light_plane_pose
```

Determine the orientation of the light plane

A plane in space is defined by at least 3 non collinear points in space

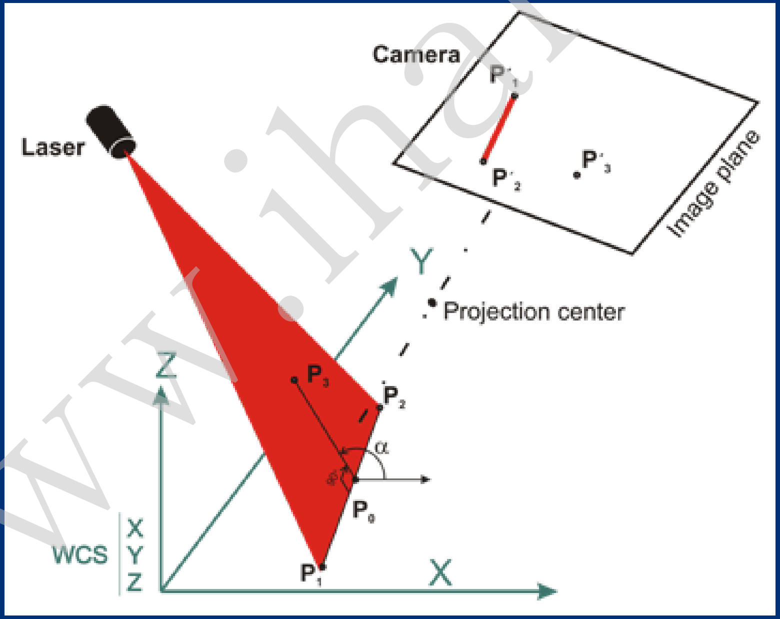


- Record the laser line at two different heights in the measurement volume
- Use of more than 3 points brings better accuracy through redundancy!

Light plane coordinate system

It is convenient to define a “light plane coordinate system” (LPCS)

- Z^{LPCS} = 0 correspond to the light plane,
- The origin of the LPCS can be chosen arbitrary besides the precedent condition
- How: rotate the WCS around the axis P_1P_2 by the angle α !

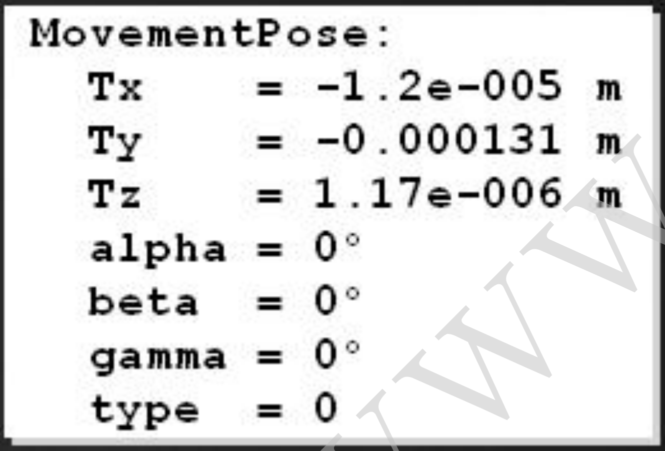


How to determine the movement pose?

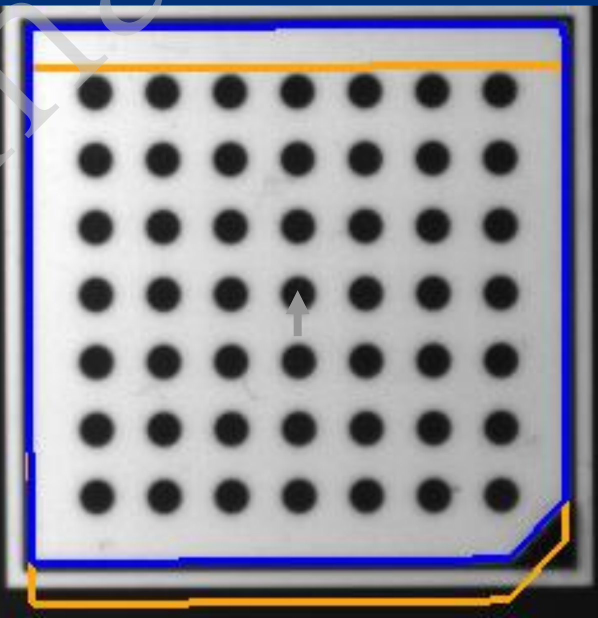
Measure the displacement of the scanning system by using a known object, like e.g. a calibration plate

- Acquire an image of the known object at the position 1
- Measure the coordinates of relevant points (x_1^{WCS}, y_1^{WCS}, z_1^{WCS})
- Move the object using the scanning system
- Acquire an image of the known object at the position 2
- Measure the coordinates of relevant points (x_2^{WCS}, y_2^{WCS}, z_2^{WCS})
- The translation vector of the scanning system is given by:

$$t^{WCS} = \begin{pmatrix} x_2^{WCS} - x_1^{WCS} \\ y_2^{WCS} - y_1^{WCS} \\ z_2^{WCS} - z_1^{WCS} \end{pmatrix}$$



```
MovementPose:
  Tx    = -1.2e-005 m
  Ty    = -0.000131 m
  Tz    = 1.17e-006 m
  alpha = 0°
  beta  = 0°
  gamma = 0°
  type  = 0
```



Sheet-of-light Smart 3D Sensor

HALCON

the Power of Machine Vision

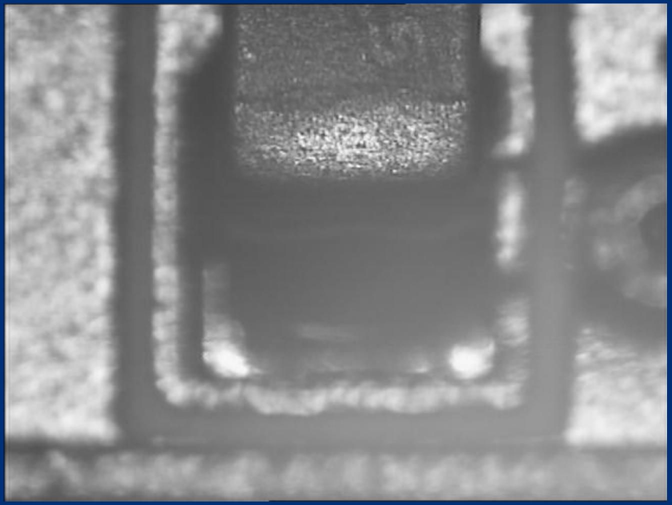
MVTec

Depth from Focus

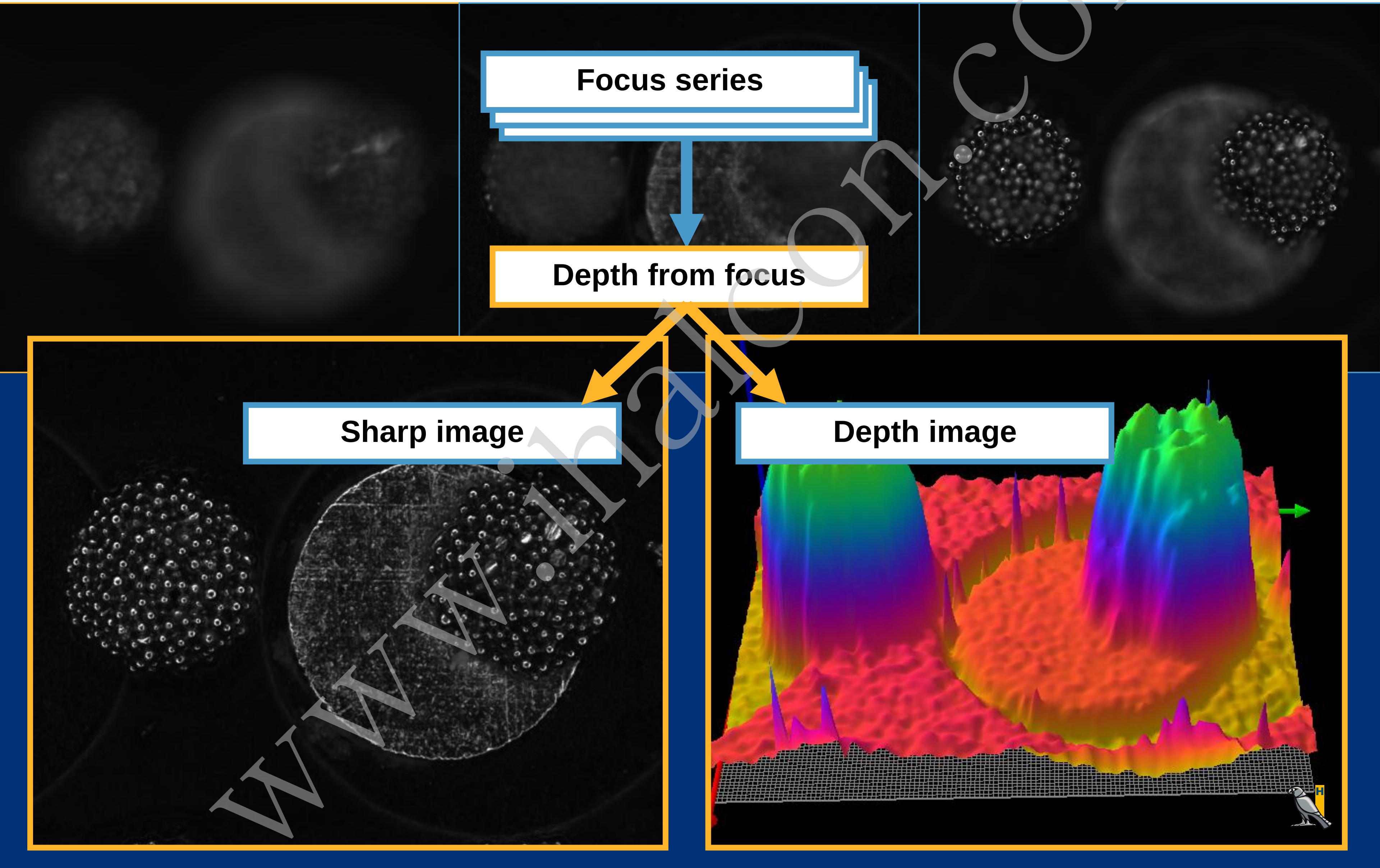
Depth From Focus: Idea

- Within a scene object points have different distances to the camera
- The camera has a limited depth of field
- Depending on the distance and the focus different object points are displayed sharply in the image
- Taking images with various object distances each object point can be displayed sharply in at least one image

- By determining in which image an object point is projected sharply the distance can be determined



Depth from focus: Basic workflow



Focus series

Depth from focus

Sharp image

Depth image

Operator: `depth_from_focus`

- Apply the depth from focus method
- Determine
  - Depth map, i.e., the distance for each pixel
  - Confidence of the distance value
- Usage
  - Use the focus series as input
  - Select the appropriate filter method
  - Get the depth information as an index for each pixel
- Parameters
  - `MultiFocusImage`: Focus series as multi channel image
  - `Depth`: Index image where the value for each pixel corresponds to the index of the image with the maximum sharpness
  - `Confidence`: Reliability of the depth measurement
  - `Filter`: Filter type to determine the sharpness
  - `Selection`: Handling of points with low confidence

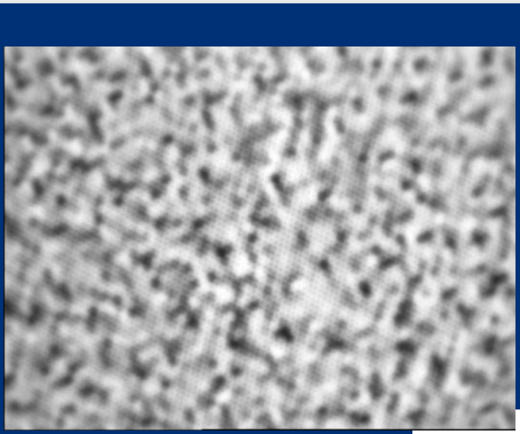
select_grayvalues_from_channels

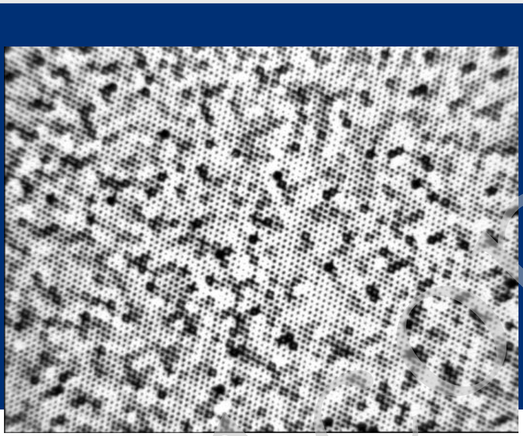
- Reconstruction of a sharp image
- Determine
  - Select the sharpest gray value for each coordinate
  - Use the depth image as index table
- Usage
  - Use the focus series and the depth image as input
  - Reconstruct the focused image
- Parameters
  - `MultiChannelImage`: Focus series as multi channel image
  - `IndexImage`: Depth image used as index table
  - `Selected`: Reconstructed sharp image

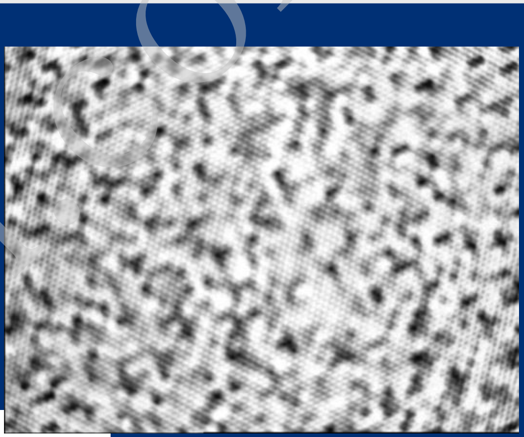
Rules to Take Images

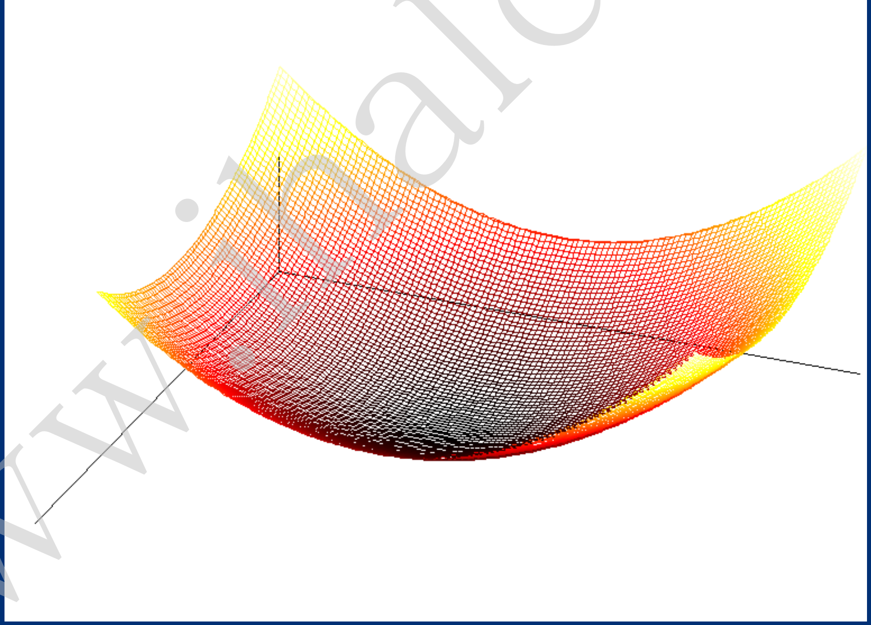
- Avoid overexposure
- Use camera with a wide dynamic range
- Use a camera with low noise
- Use diffusive illumination
- Avoid reflections
- Cover the whole distance range
- Use at least 10 images
- The maximum number of images is limited by the camera noise
- Do not move the camera or the object
- The distance between the light source and the object should remain constant

Example: Aberration

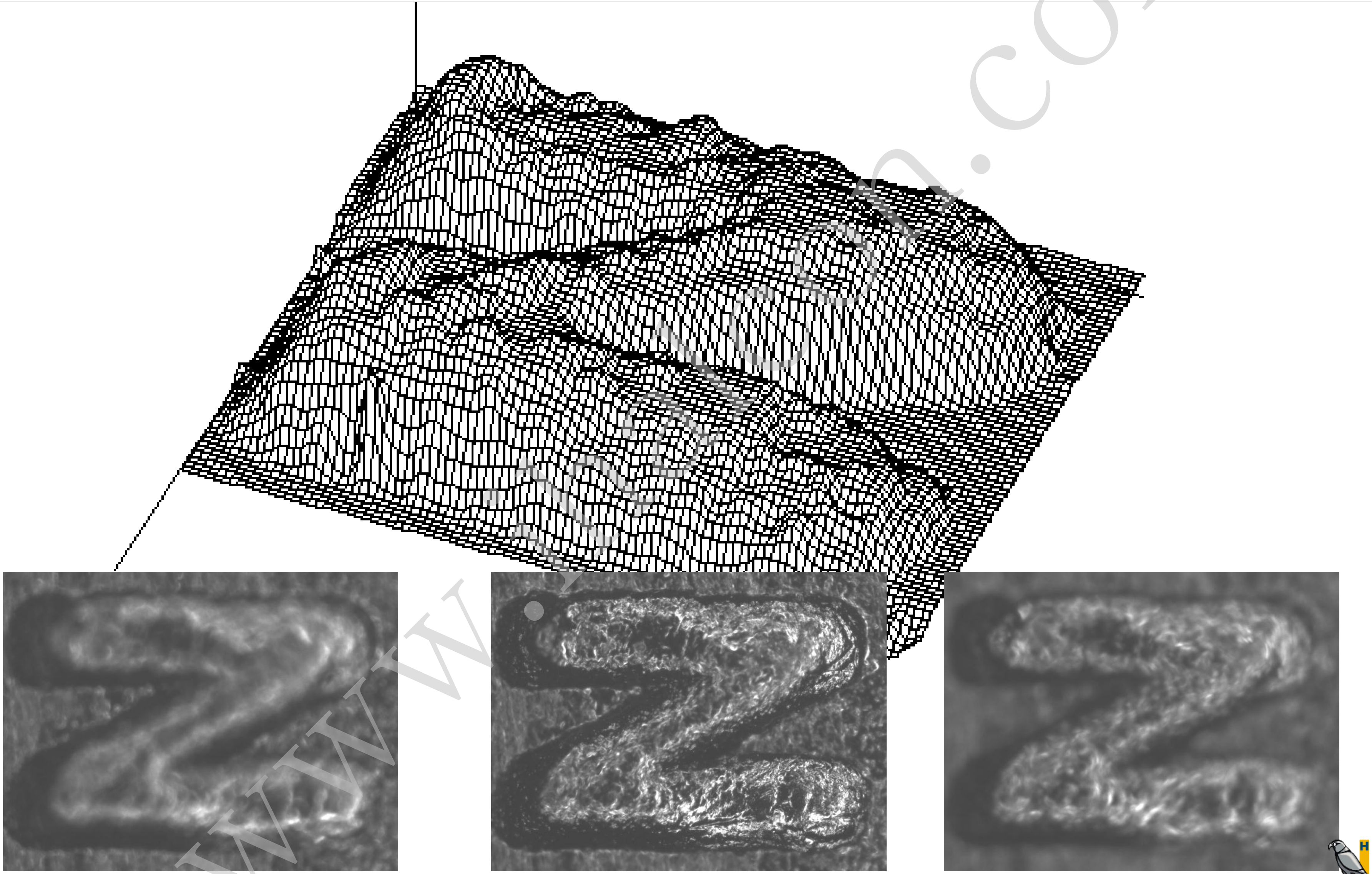




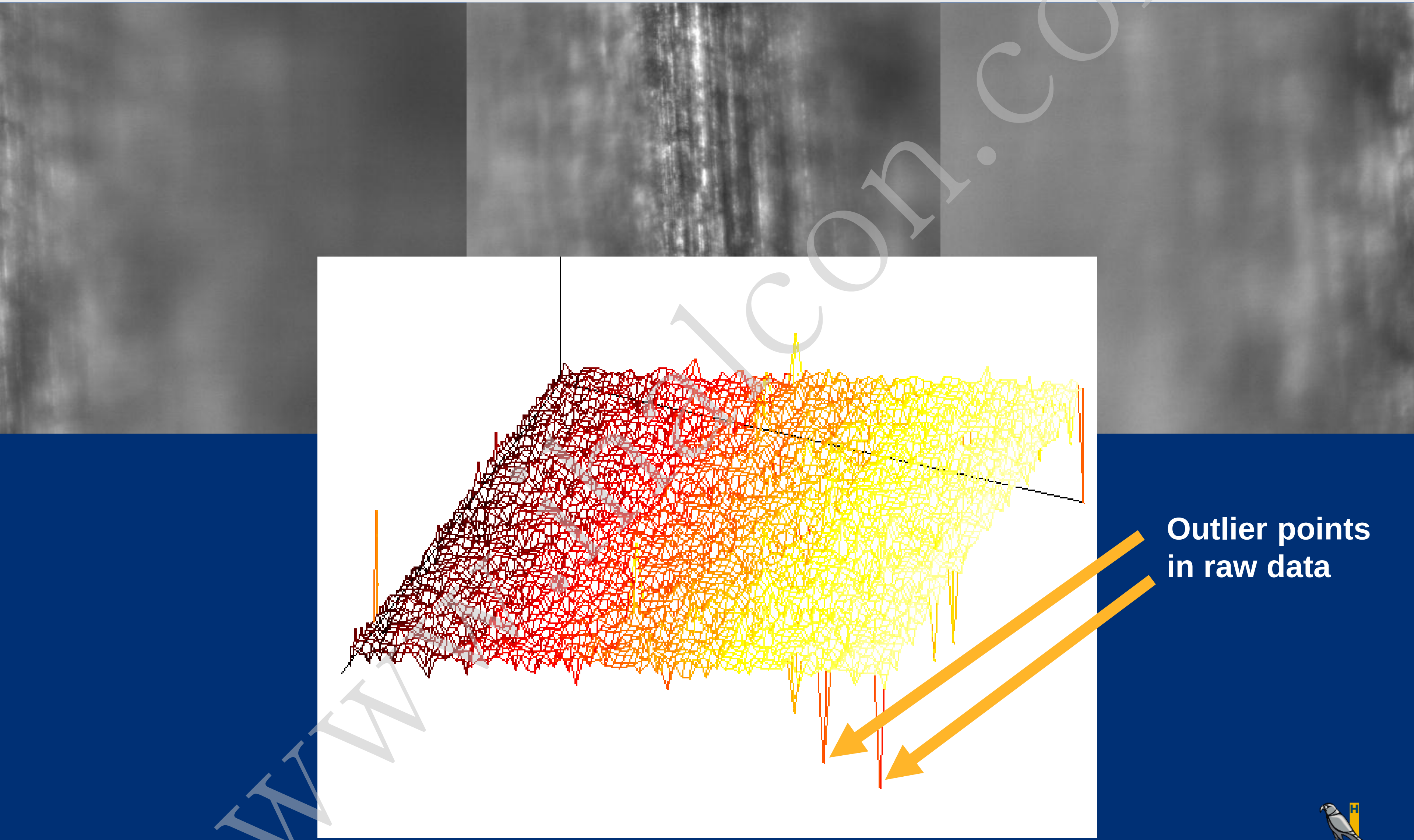




Volume Measurement

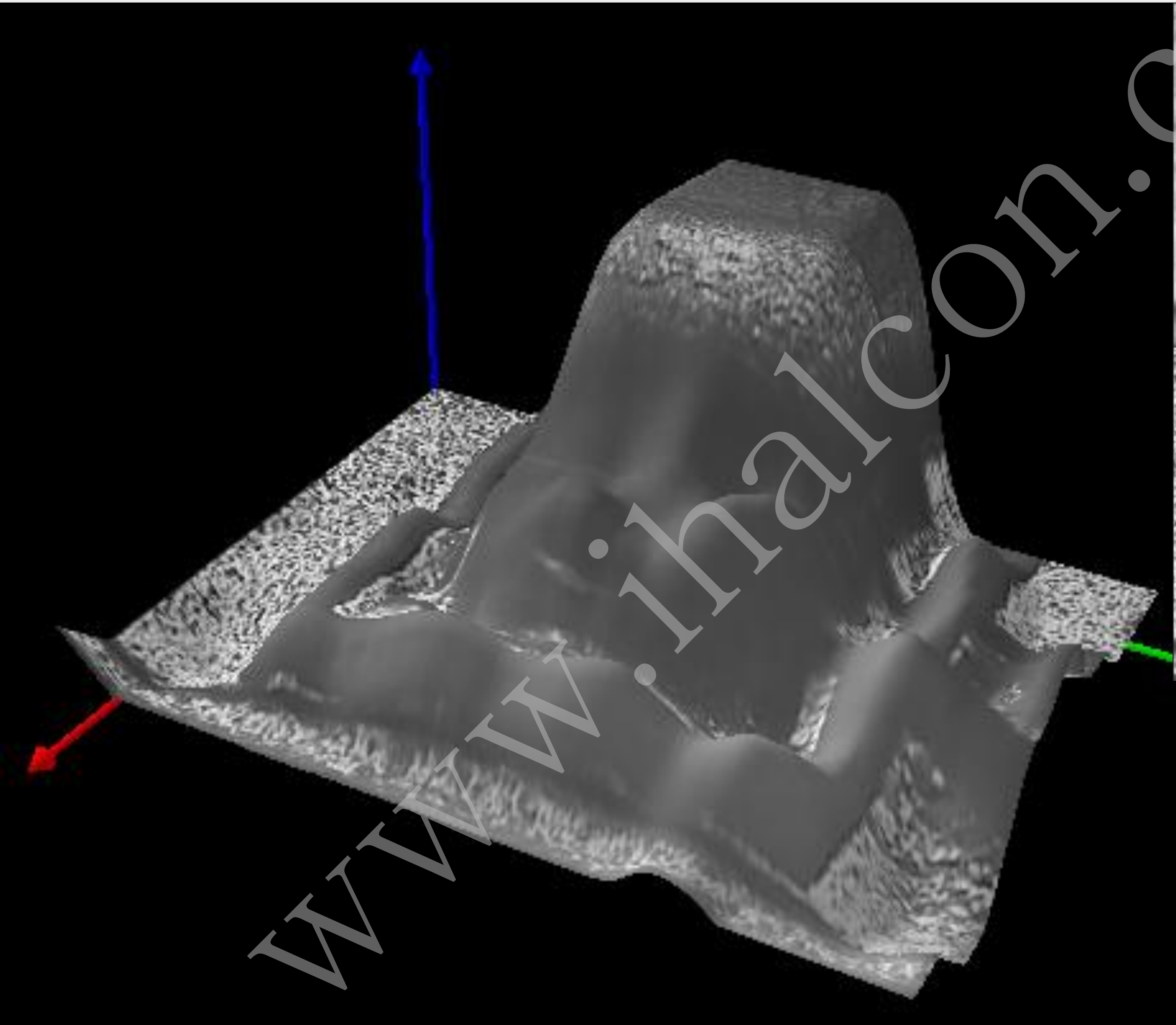


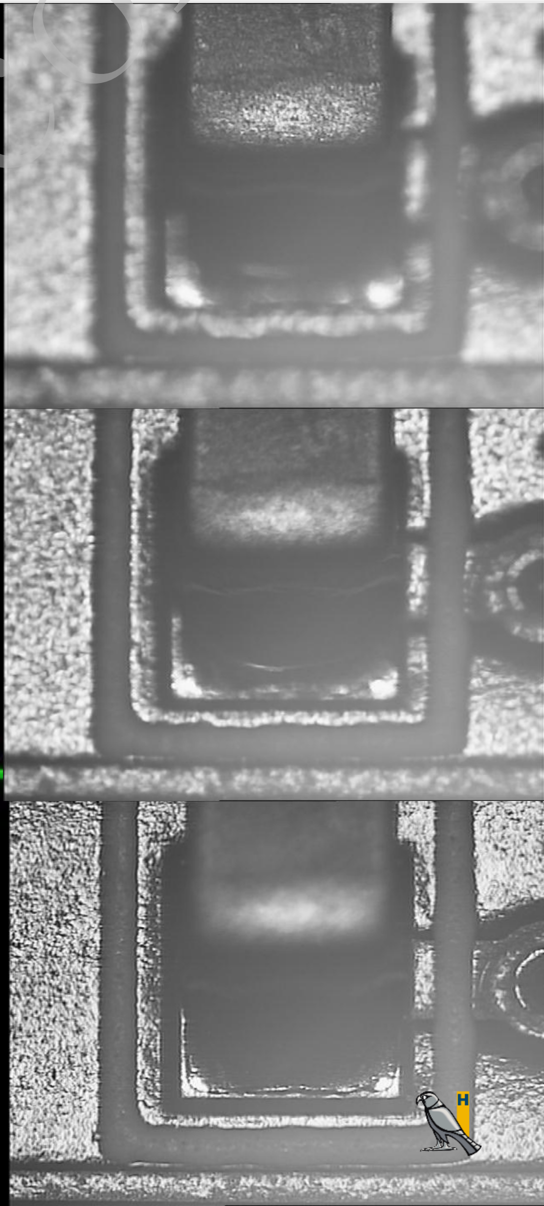
Example: Tilted Metal Surface



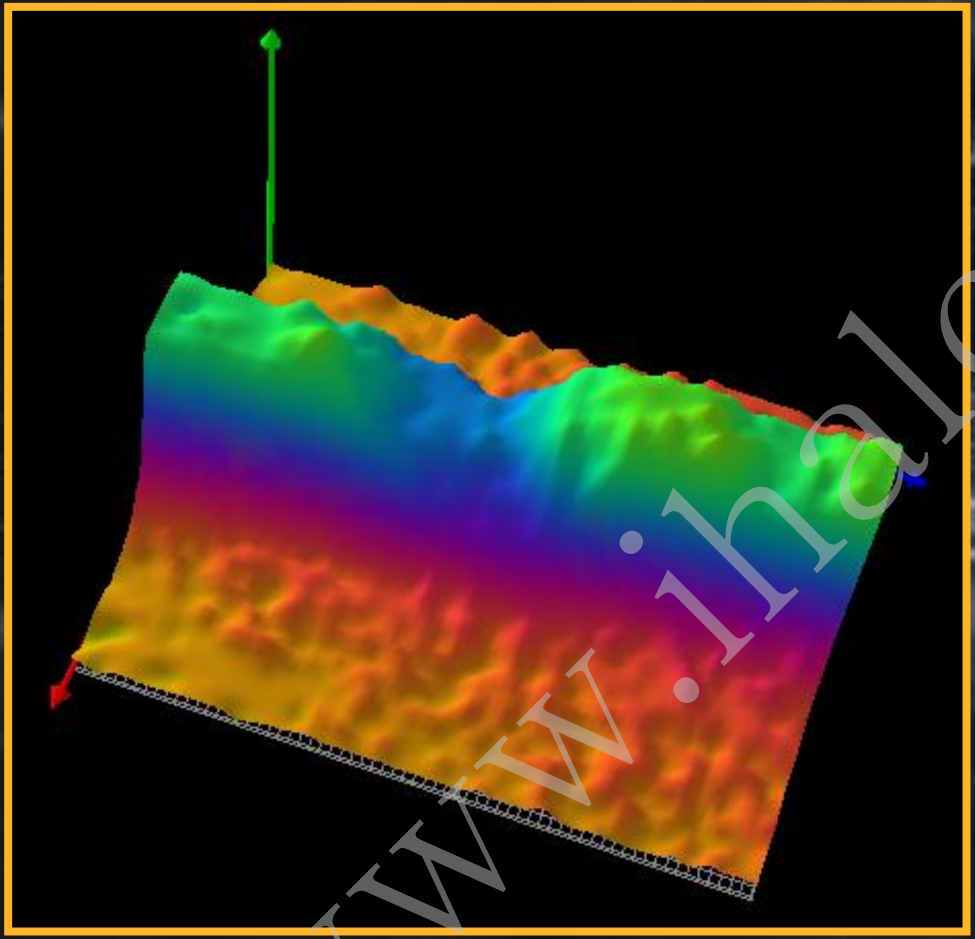
Outlier points in raw data

Example: SMD Component

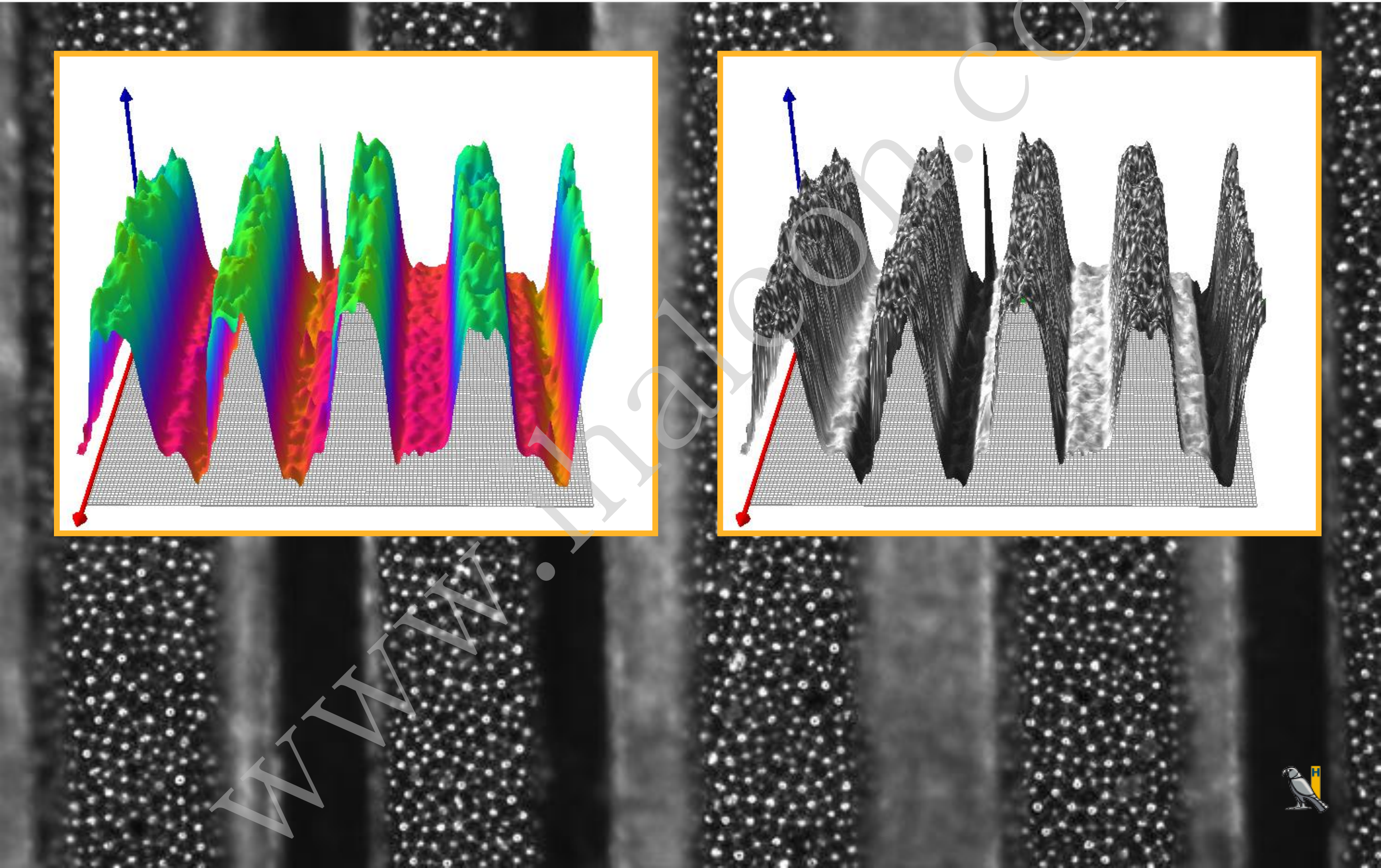




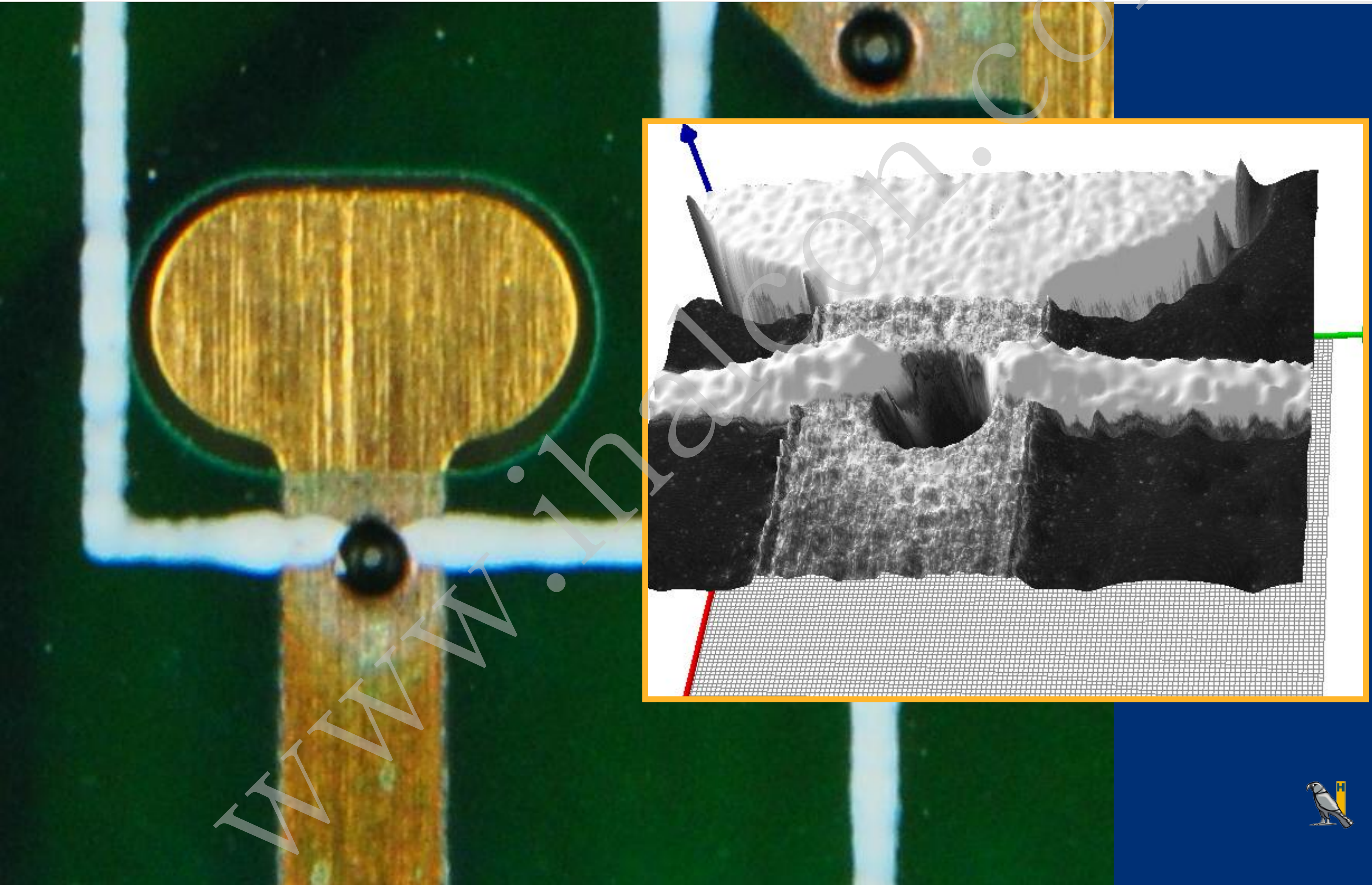
HALCON's accuracy is proven in many applications



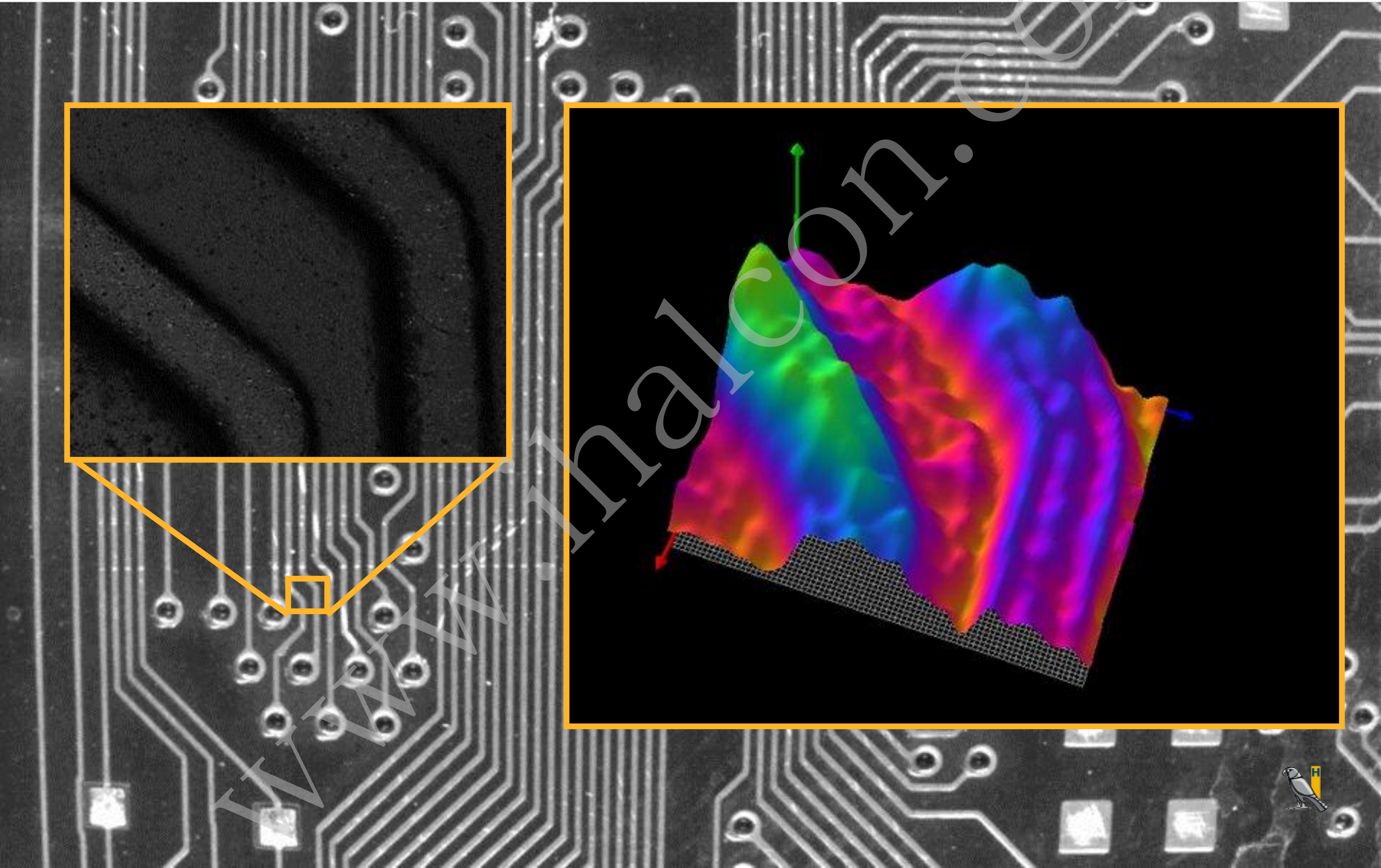
Traces of Solder Paste



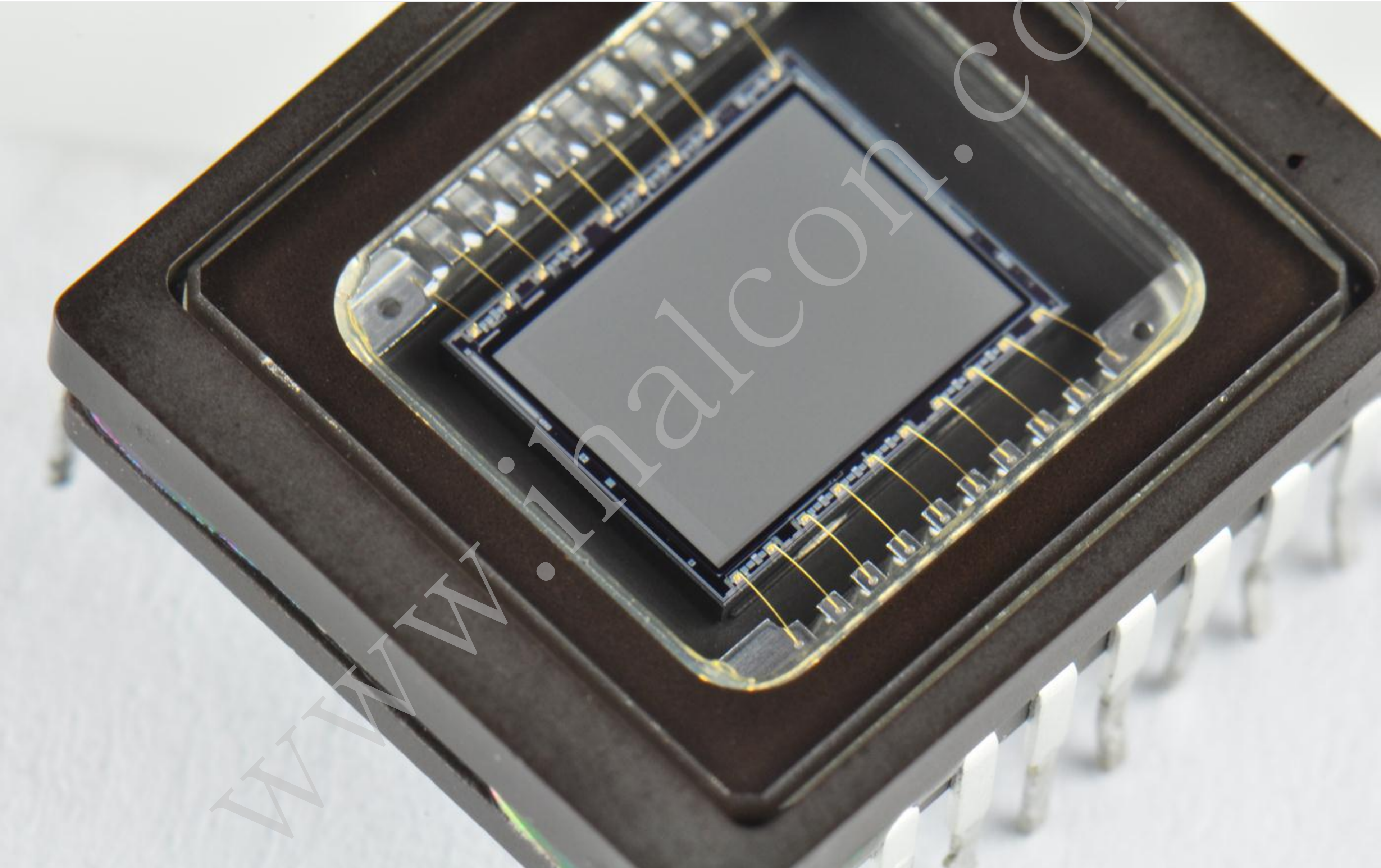
Inspection of VIAs on the PCB



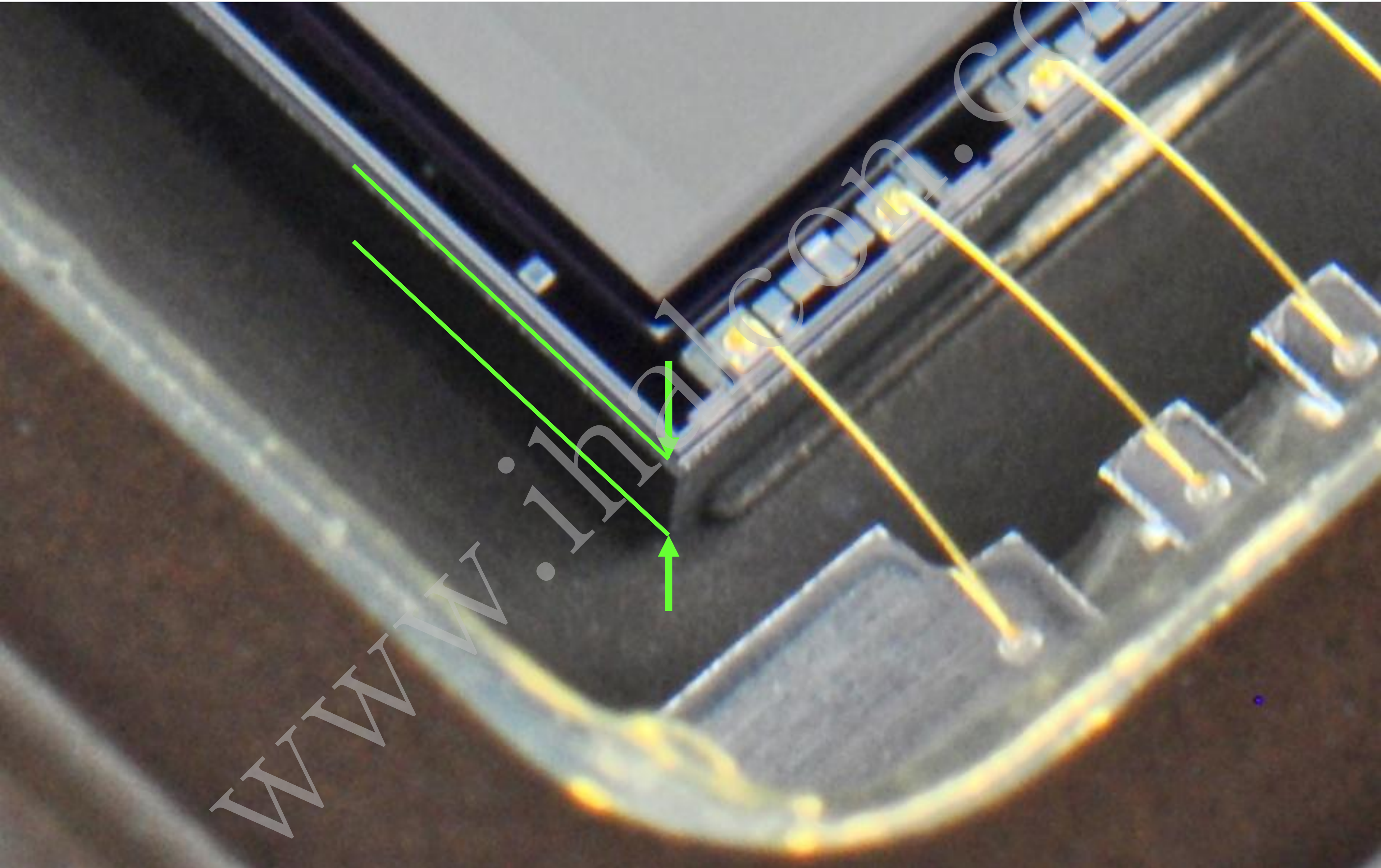
3D information of traces on PCBs can be extracted with depth from focus



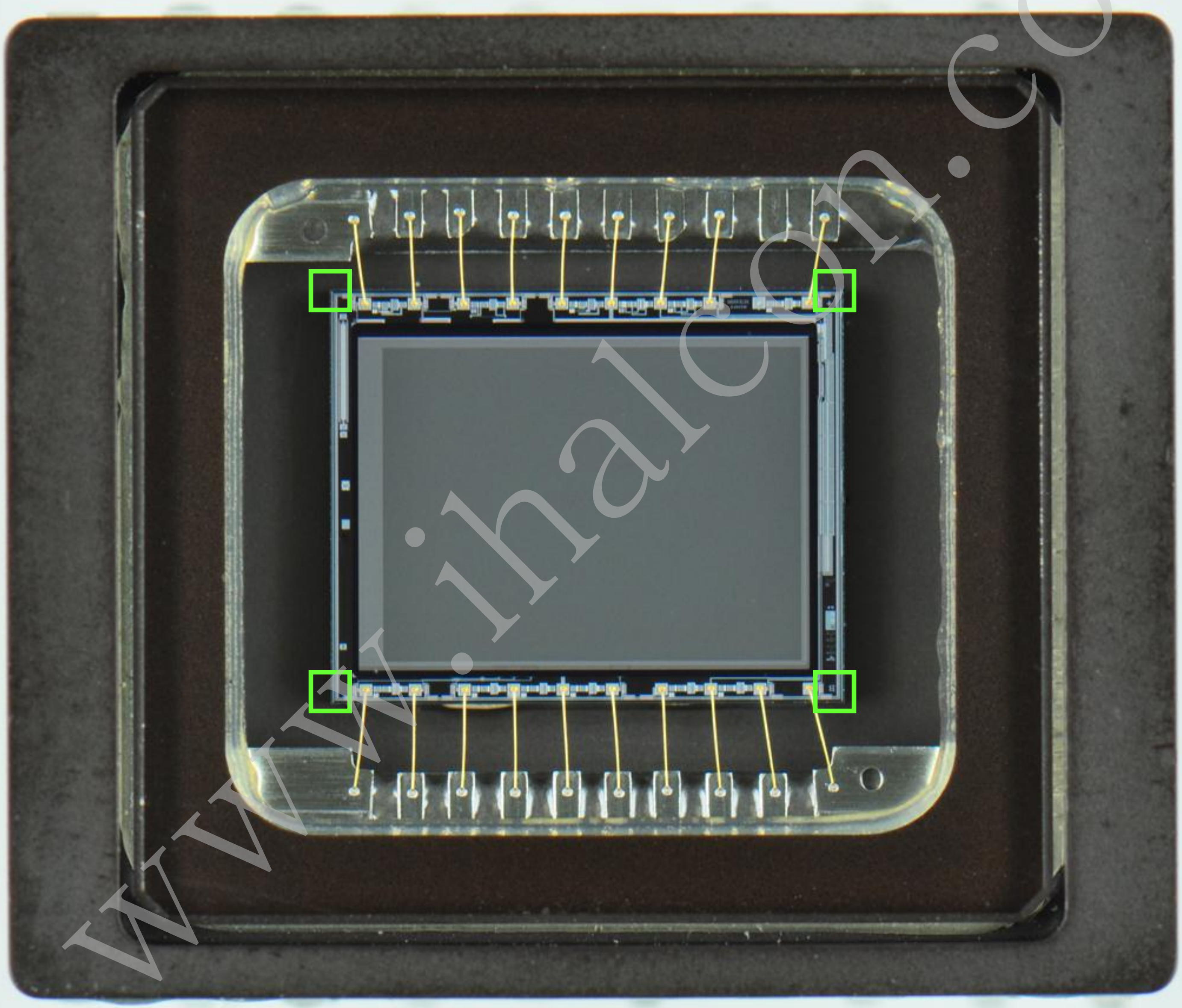
Inspection of the CMOS sensor tilt



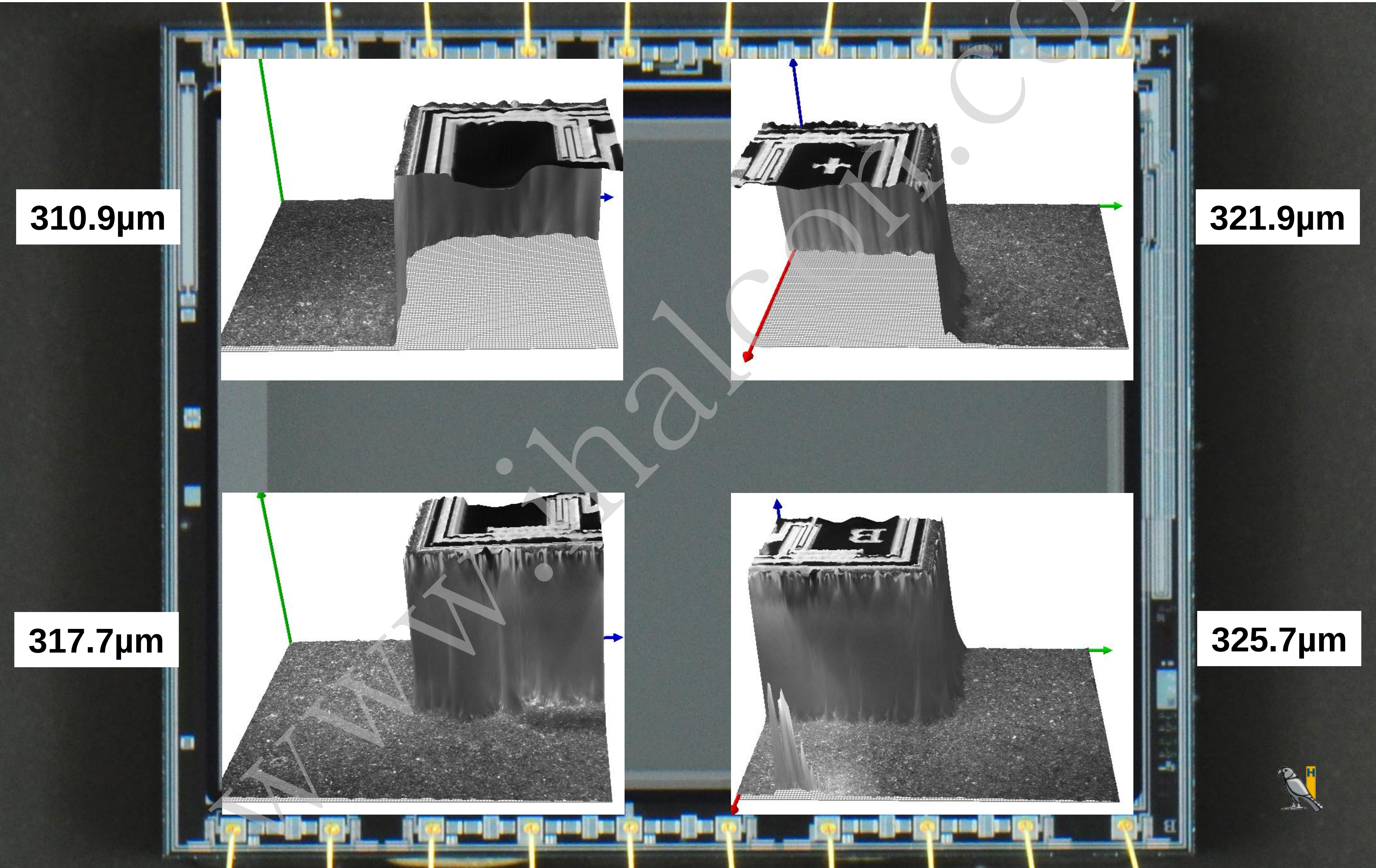
Determination of the die height relative to the housing



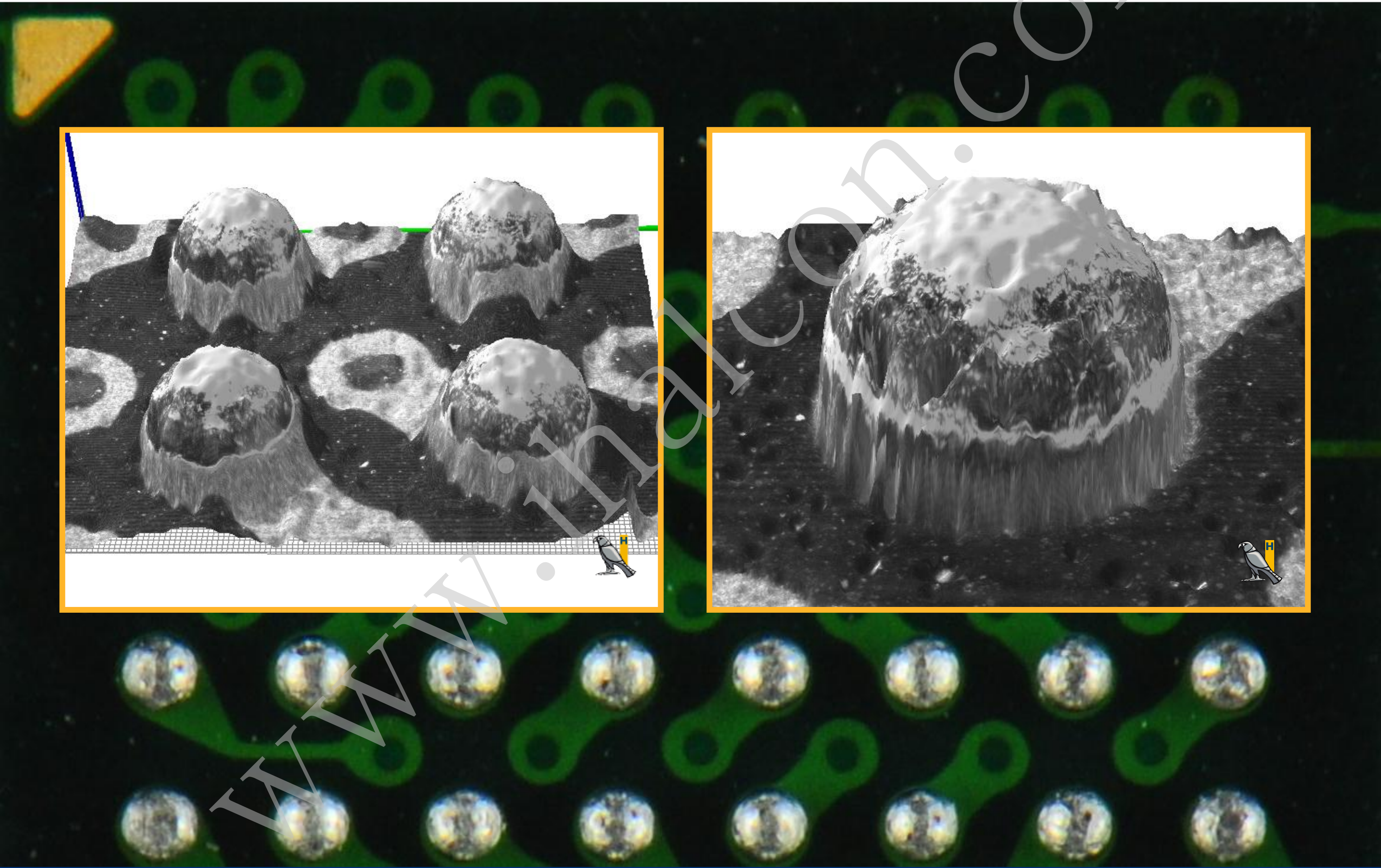
Perform one measurement for each corner



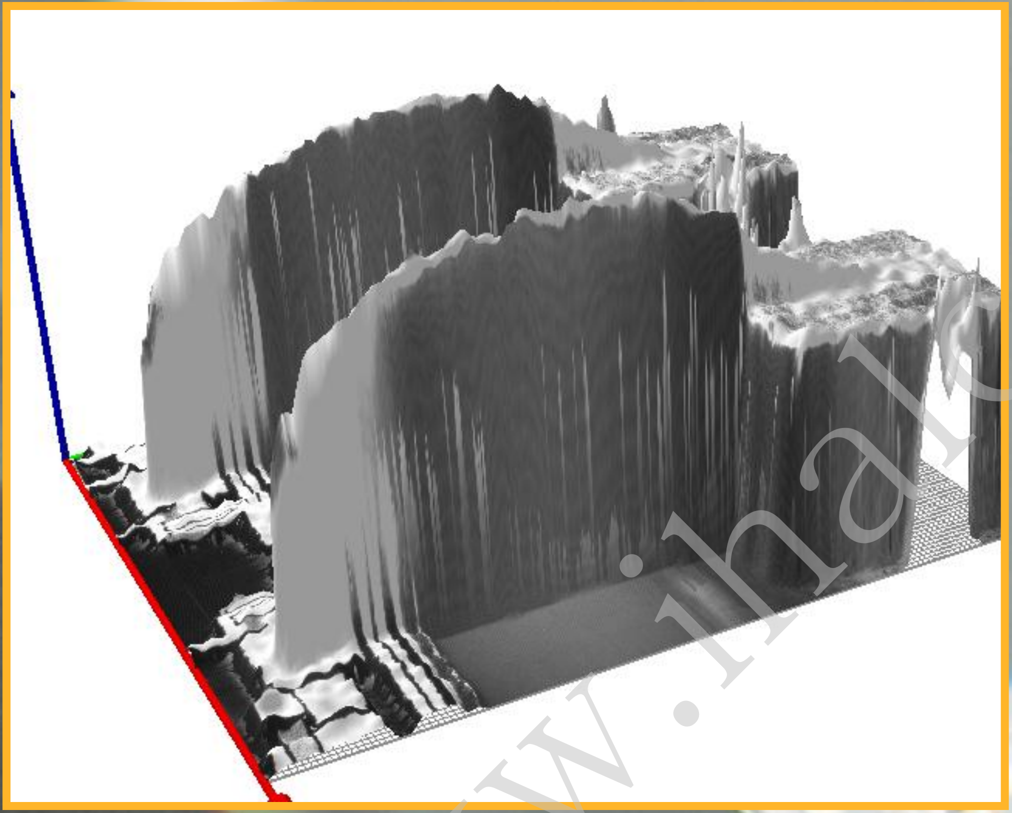
Perform one measurement for each corner



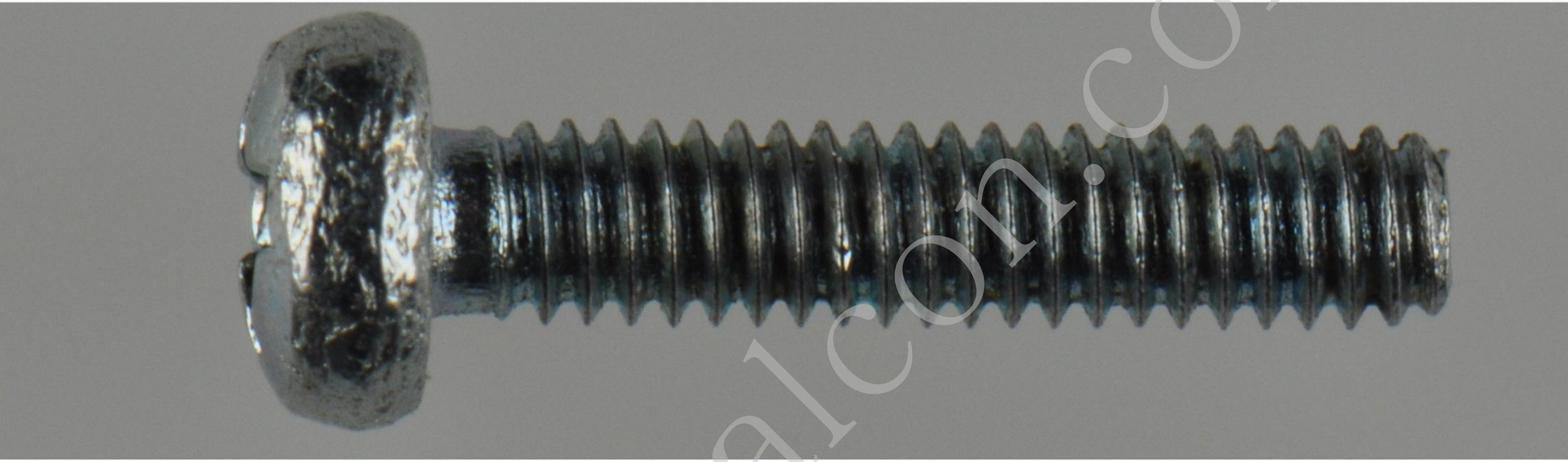
DFF for accurate 3D BGA inspection

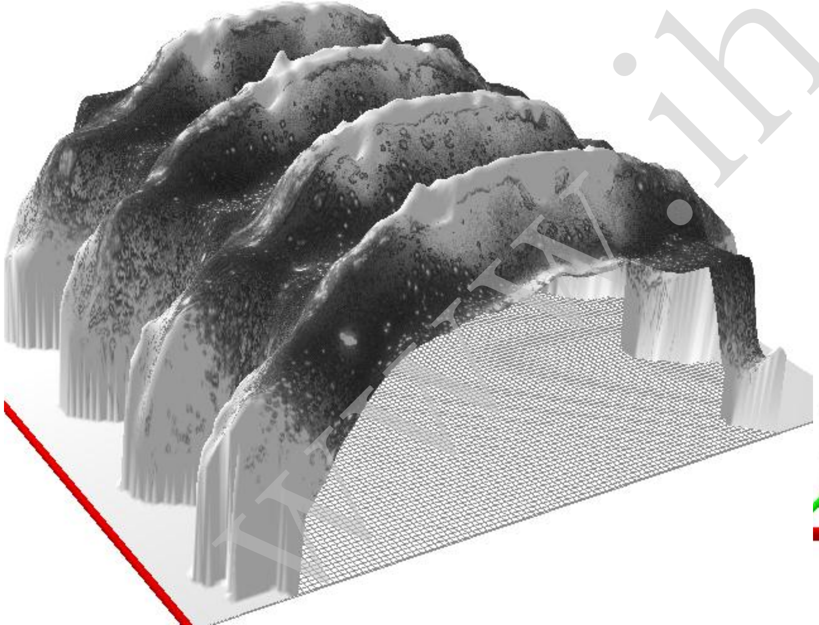


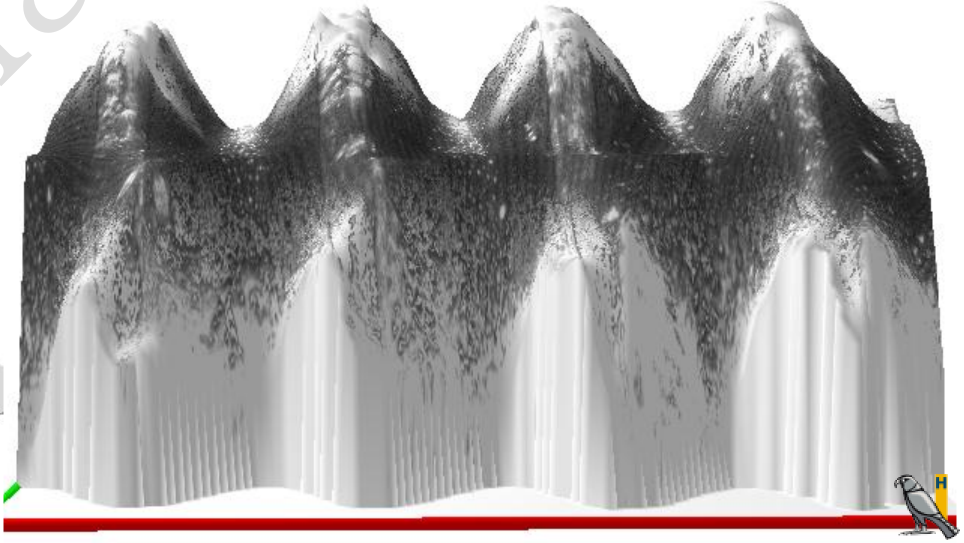
Inspection of 3D shape of wires



Inspection of Screw Thread







HALCON

the Power of Machine Vision

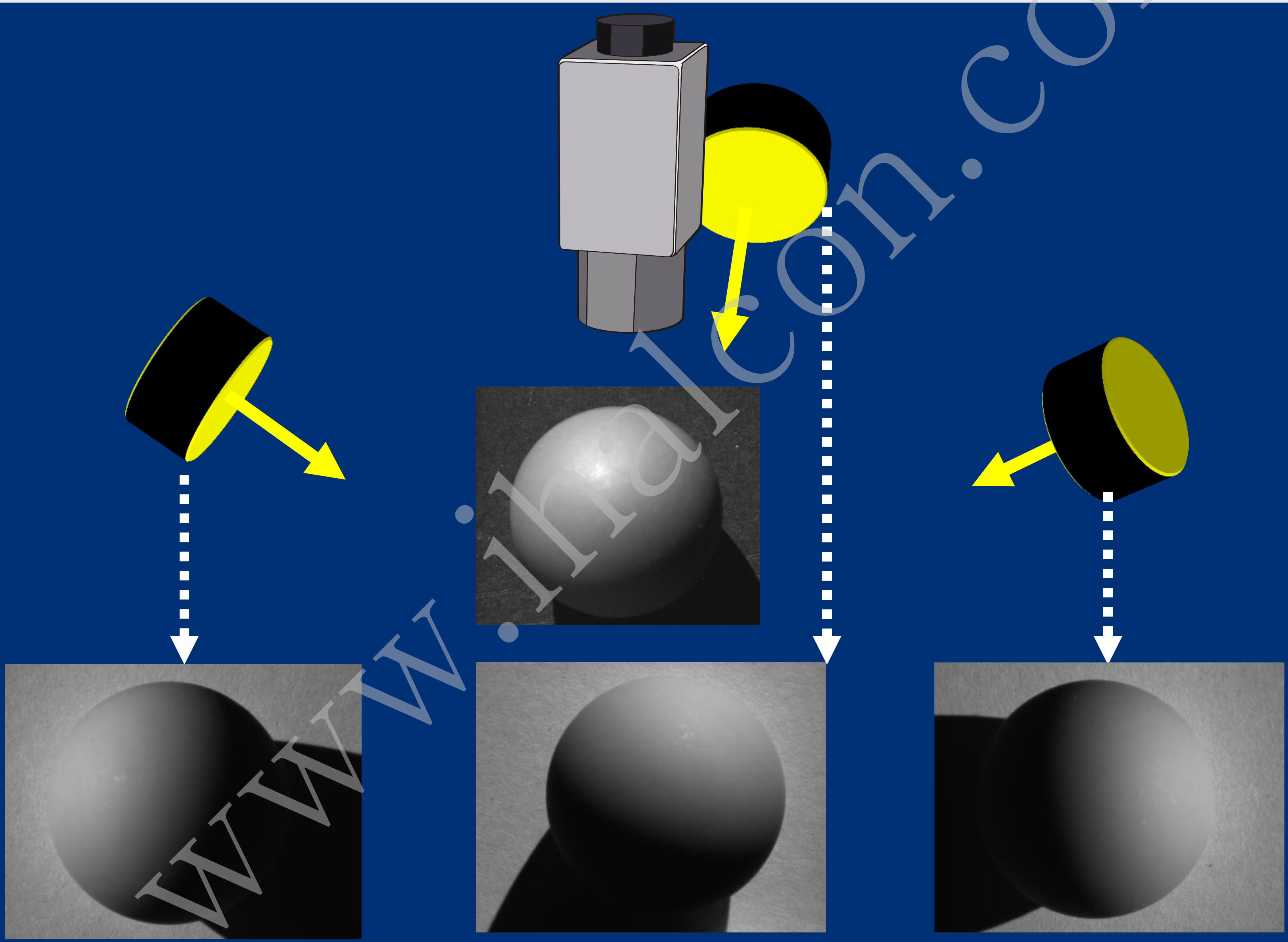
MVTEC

Photometric stereo

Photometric stereo

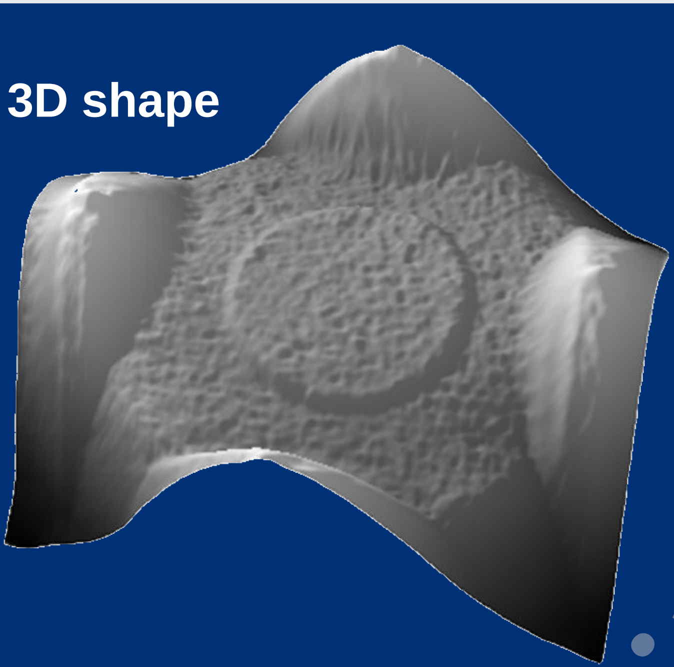
- **Principle**
- **Handling**
- **Examples**
- **Variations**

Photometric stereo extracts height information from differently illuminated images

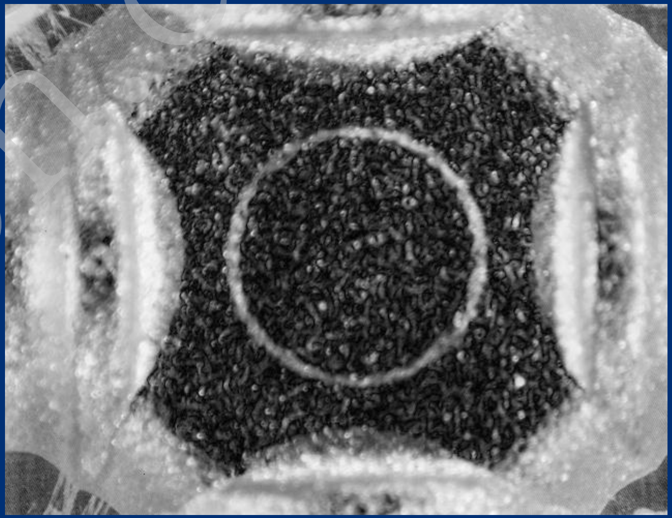


What can be derived with photometric stereo?

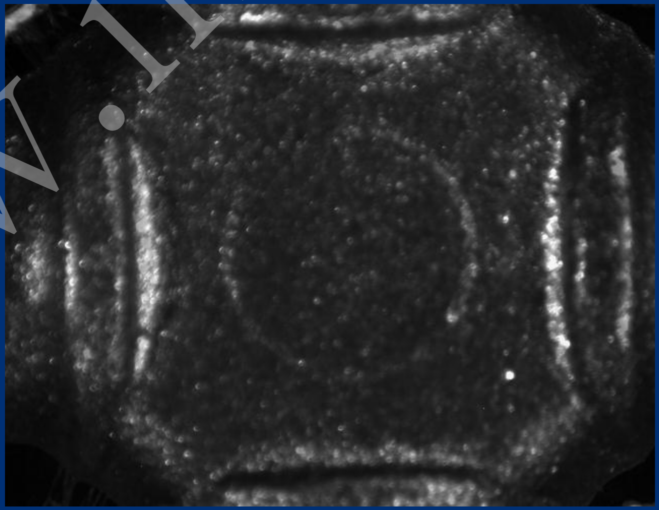
3D shape



Gradients



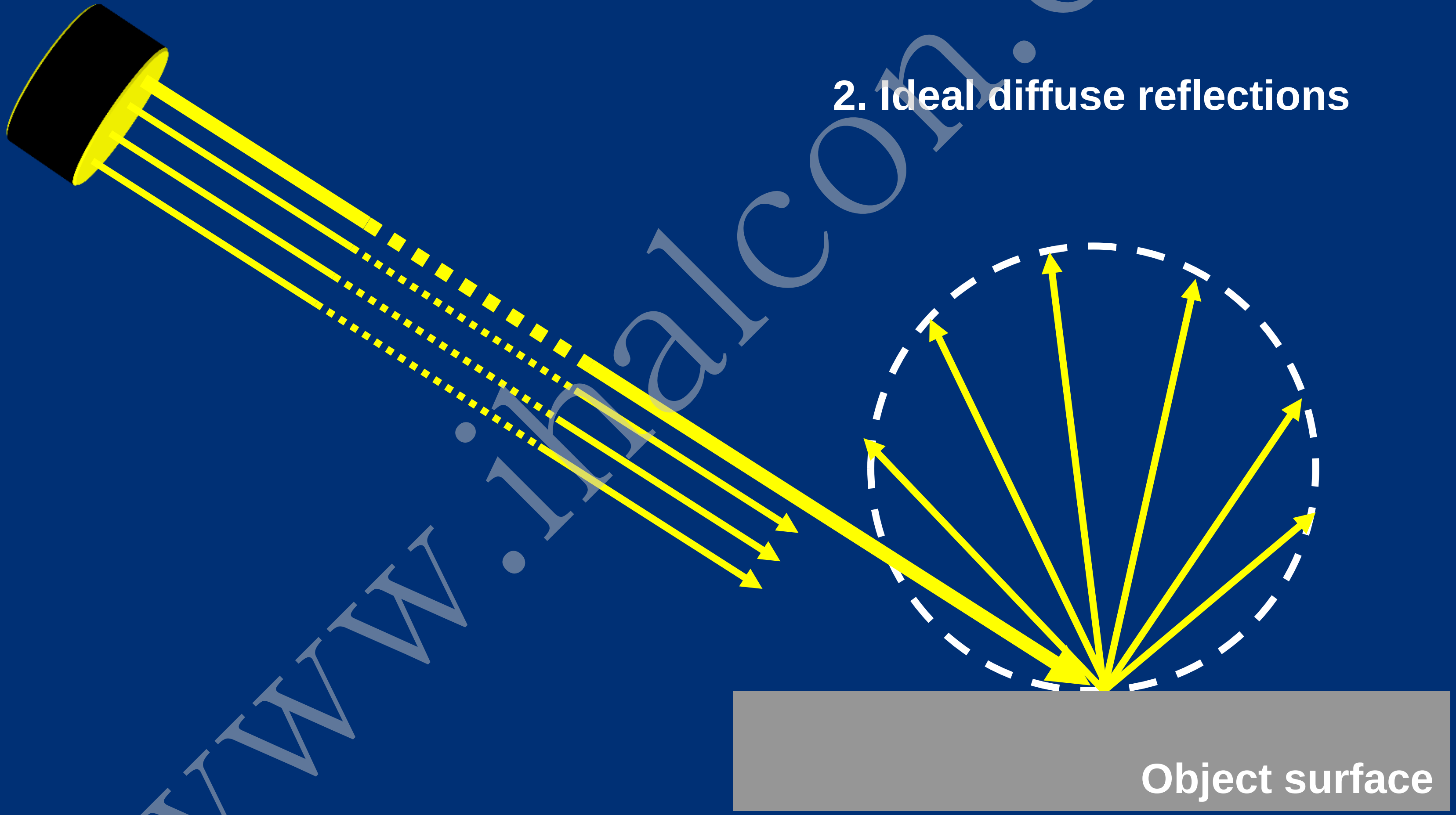
Albedo



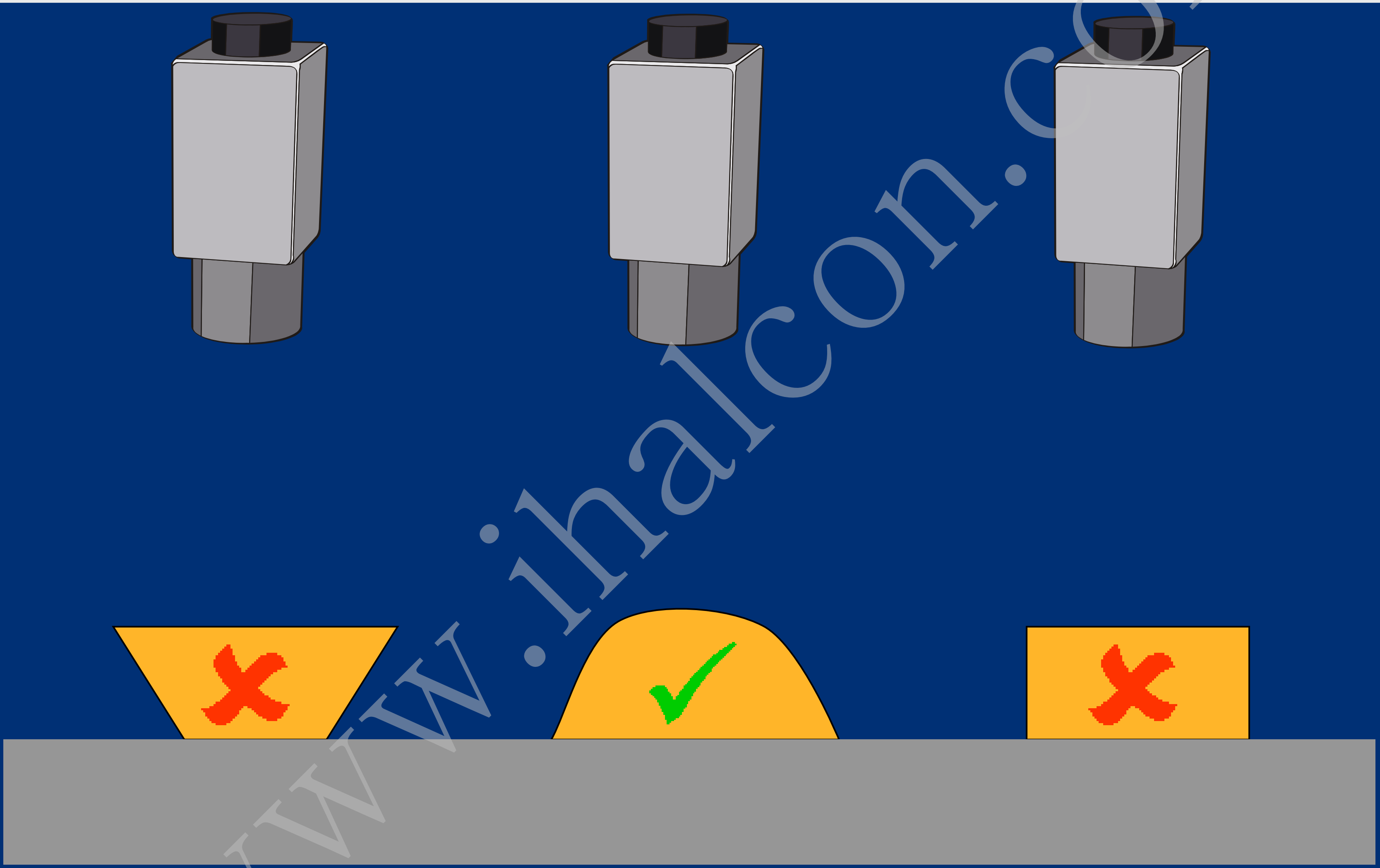
The algorithm has two assumptions:

1. The light beams are parallel

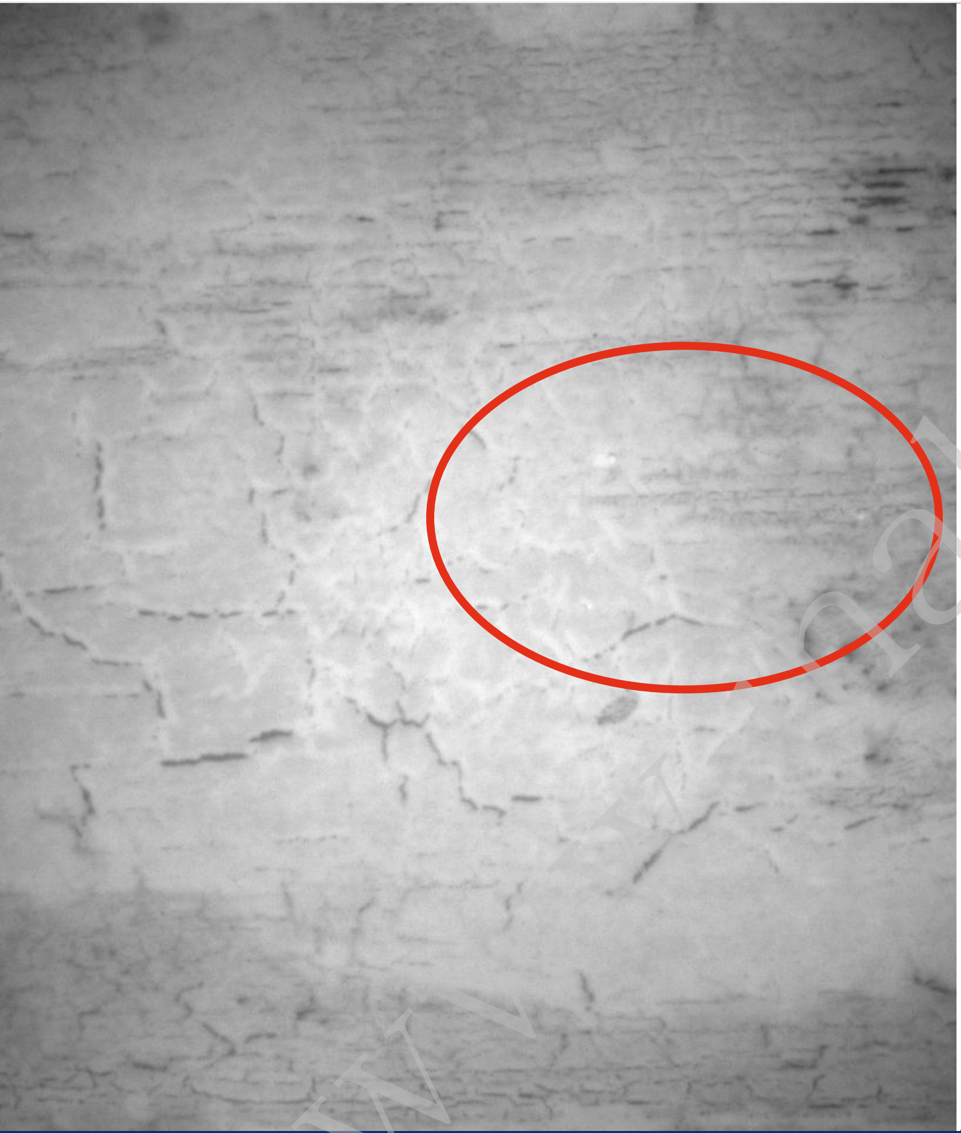
2. Ideal diffuse reflections



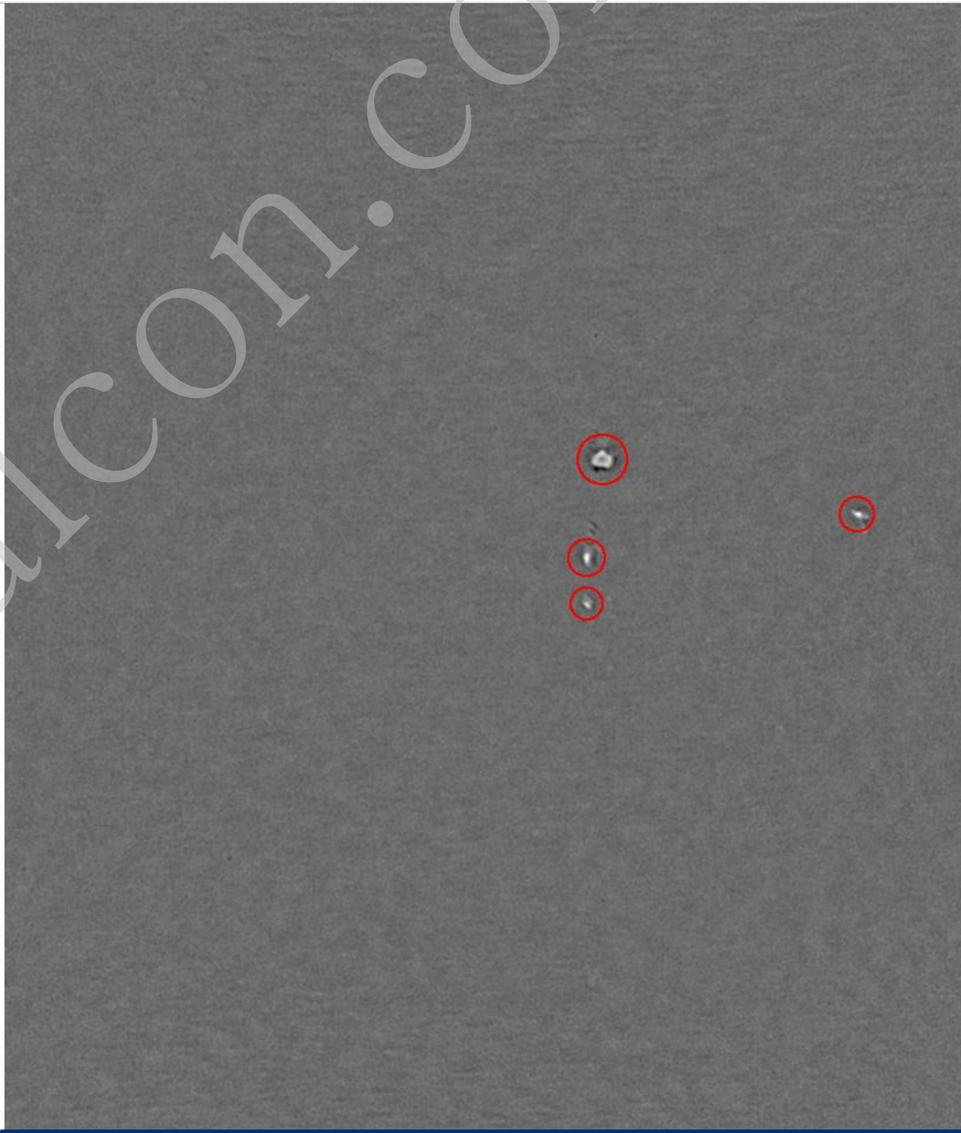
Photometric stereo needs continuous surfaces



Photometric stereo is best for surface inspection

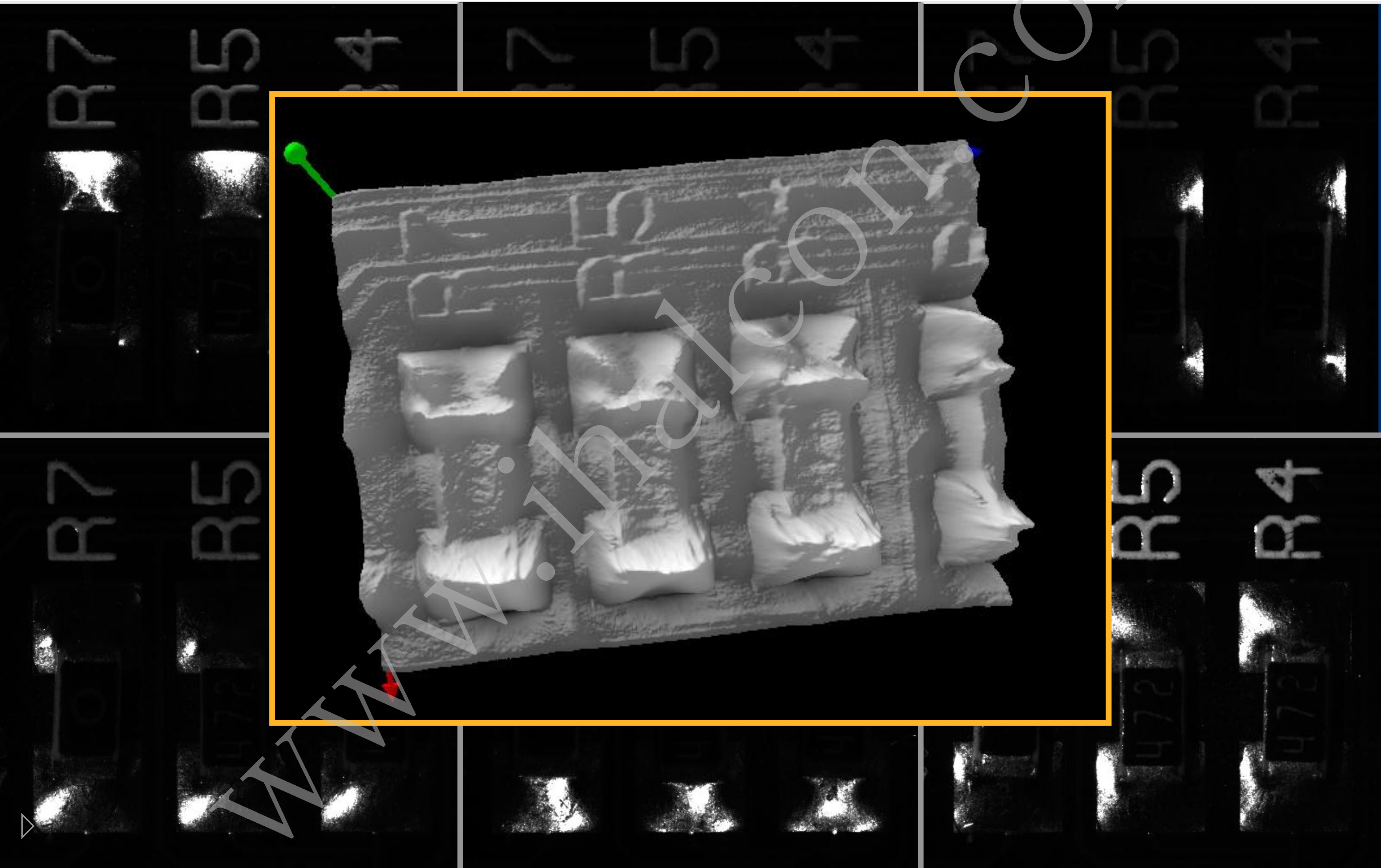


Albedo image



Surface curvature image

3D reconstruction with photometric stereo is limited



Photometric stereo

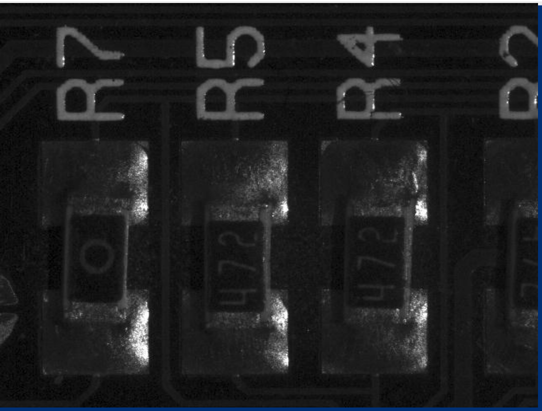
Principle

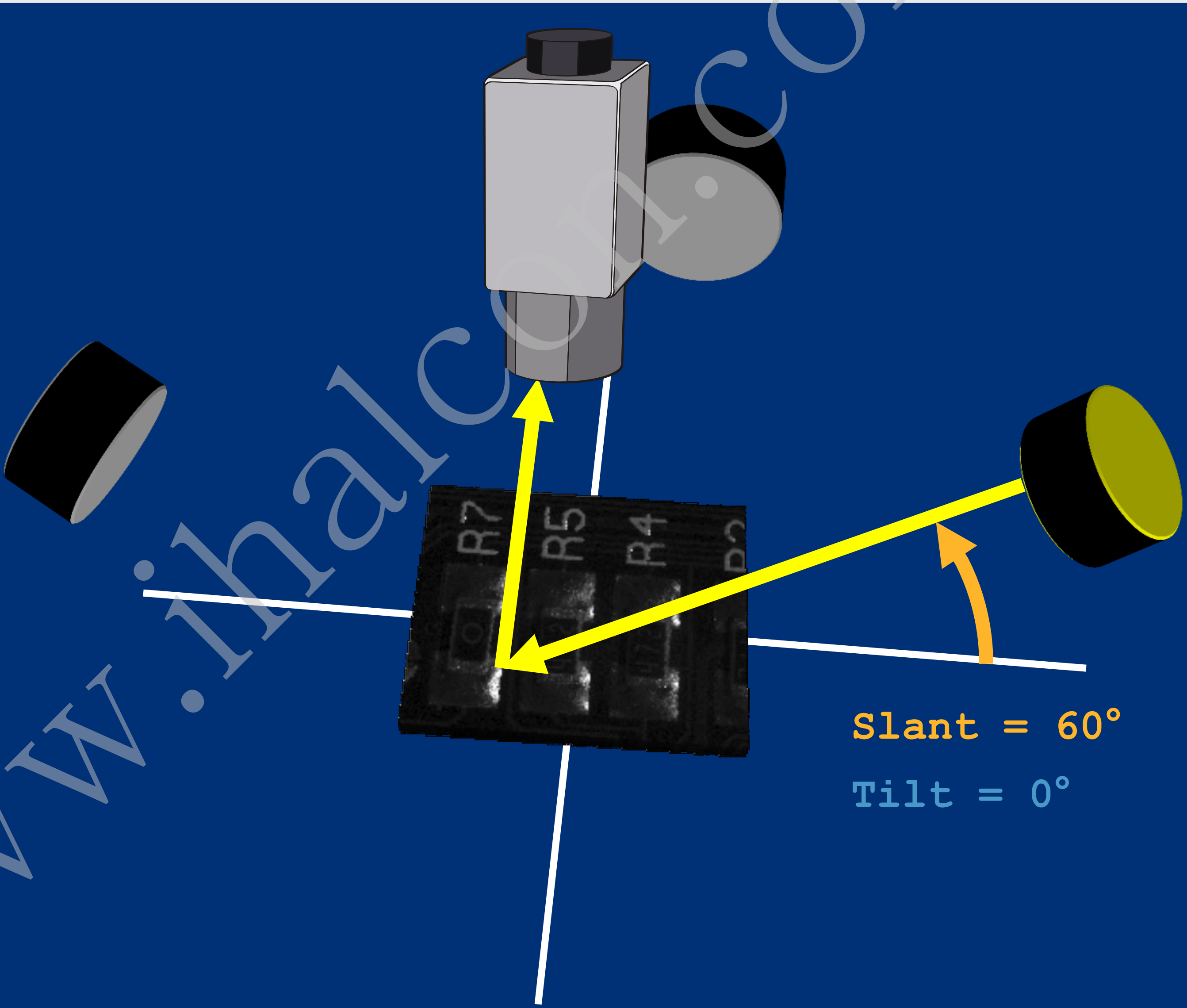
Handling

Examples

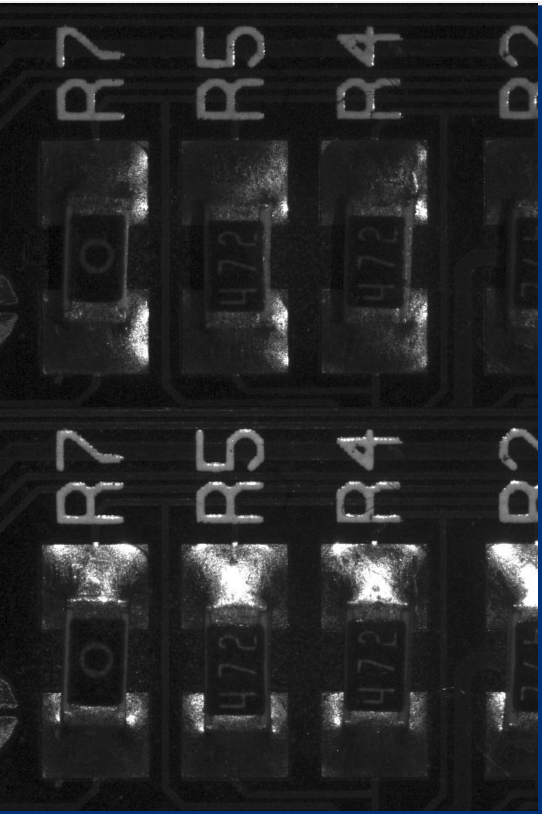
Variations

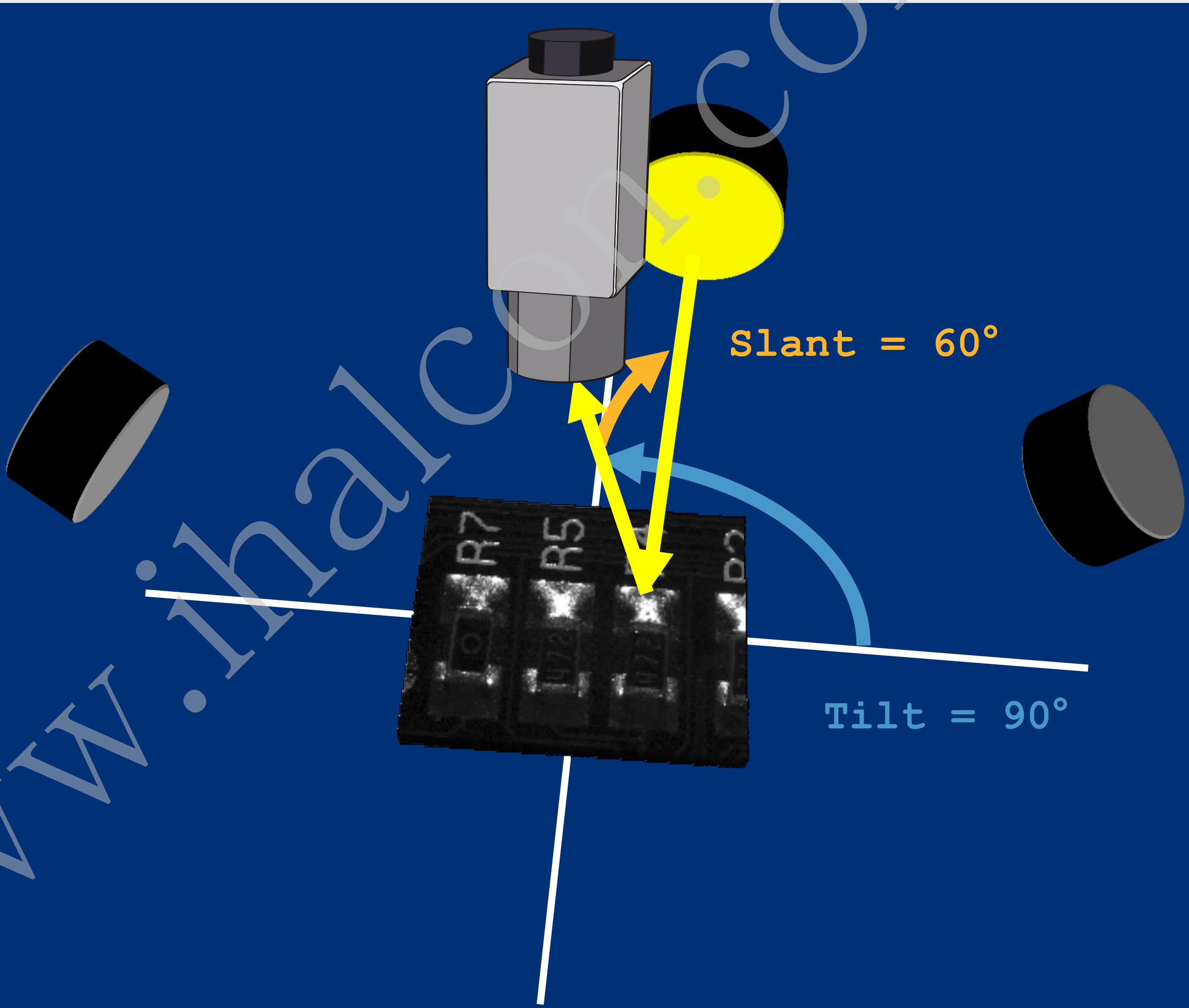
The position of the light sources must be known



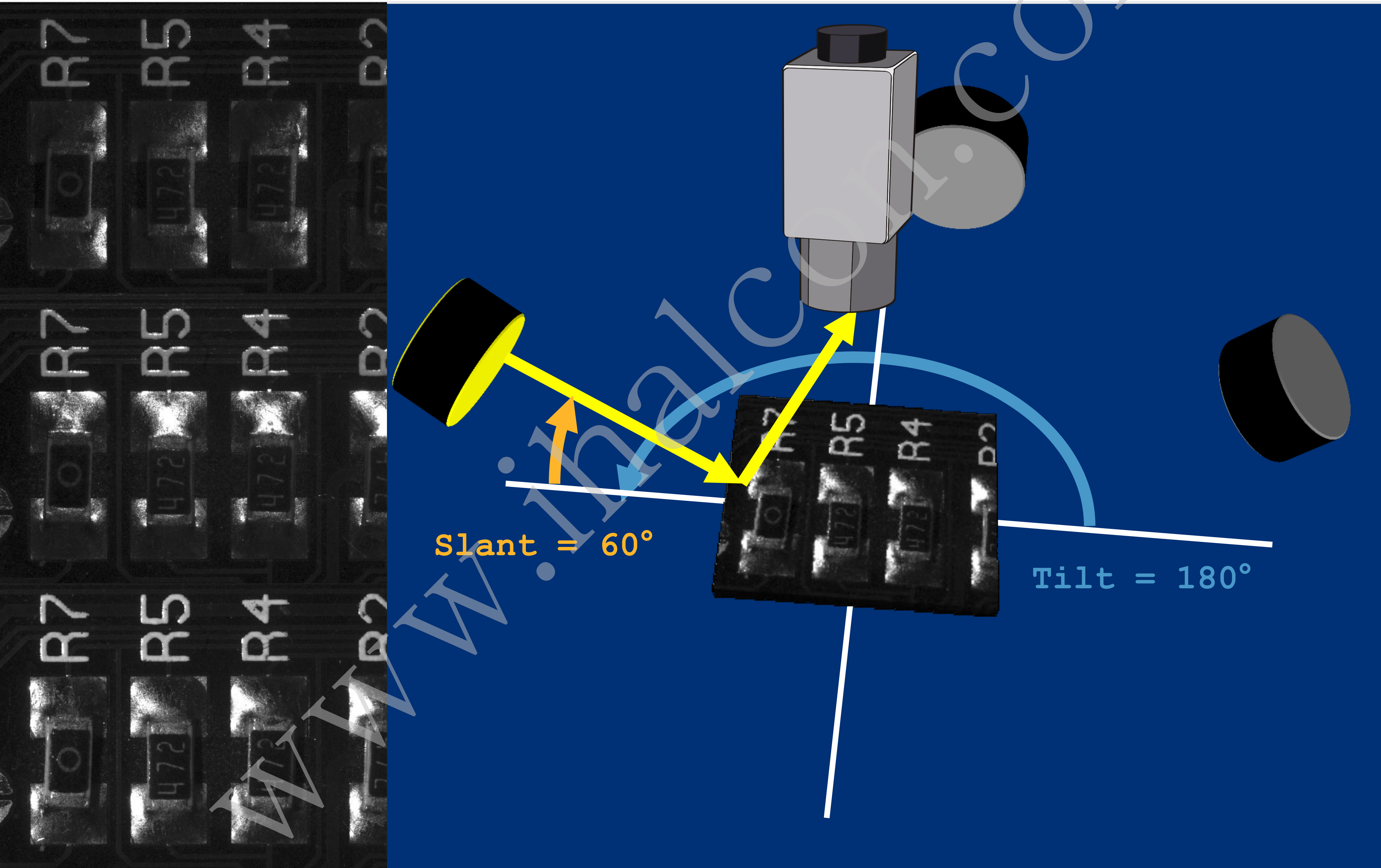


The position of the light sources must be known





The position of the light sources must be known

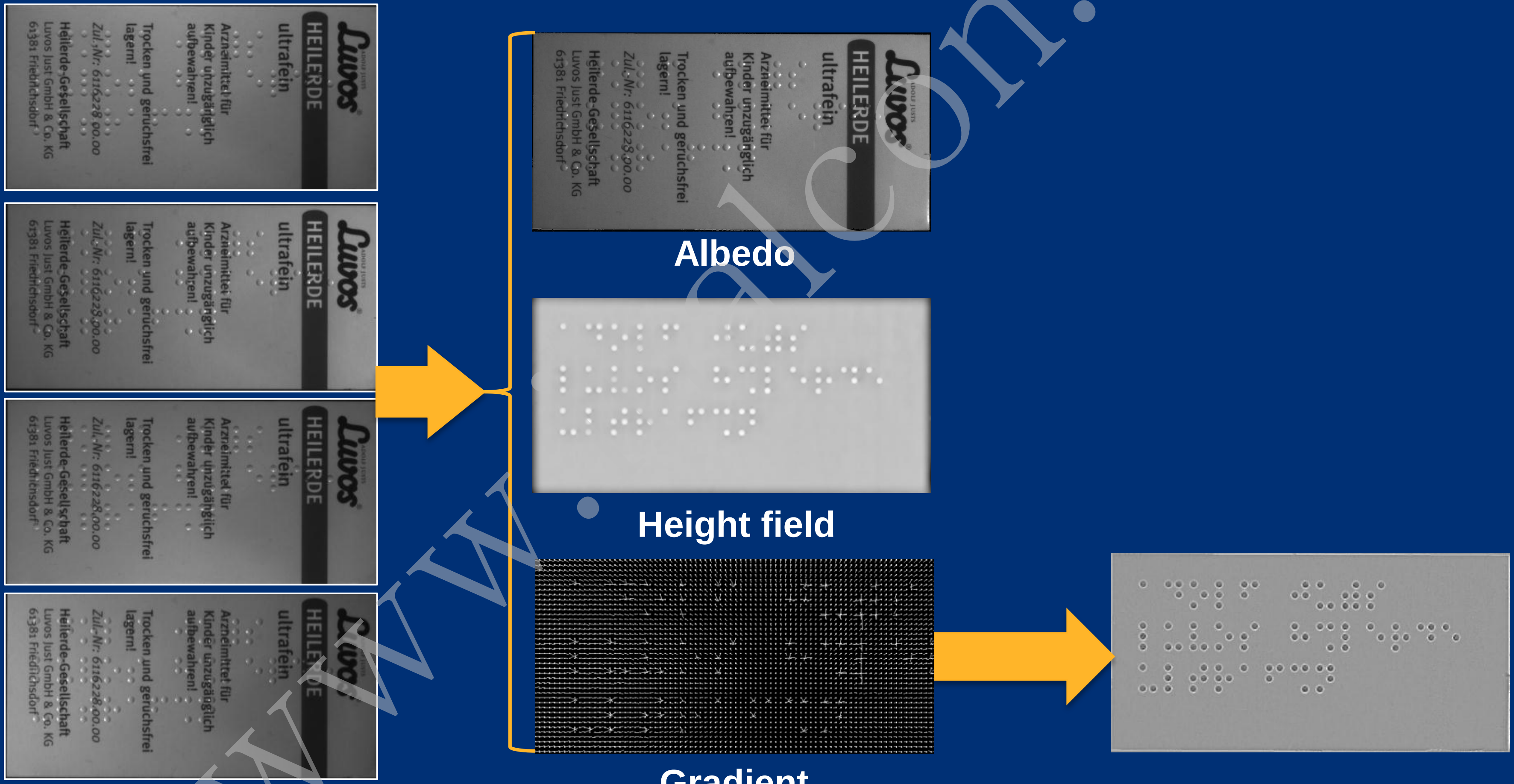


The photometric stereo workflow

1. Acquire images

2. Call `photometric_stereo`

3. Process result `derivate_vector_field`



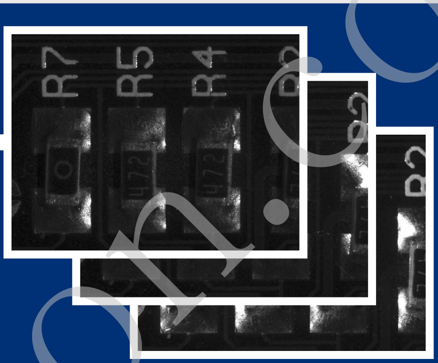
Albedo

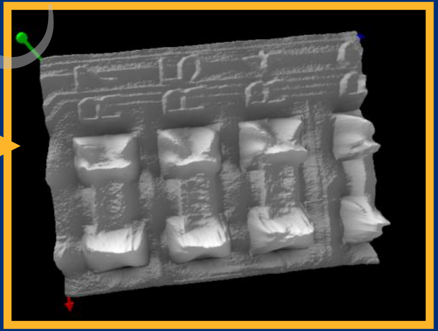
Height field

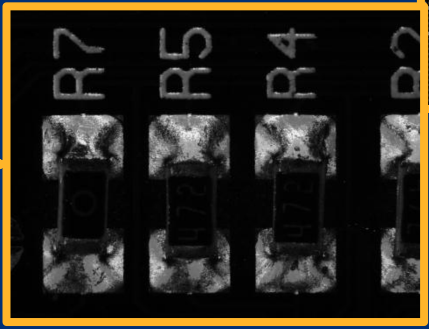
Gradient

The operator `photometric_stereo`

```
photometric_stereo(
  ▶ Images :
  ◀ HeightField,
  ◀ Gradient,
  ◀ Albedo :
  ▶ Slants,
  ▶ Tilts,
  ▶ ResultType,
  ▶ ReconstructionMethod,
  ▶ GenParamName,
  ▶ GenParamValue :)
```







Photometric stereo

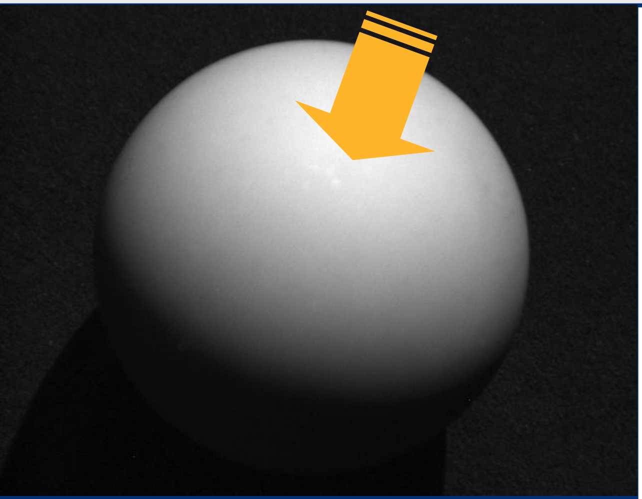
Principle

Handling

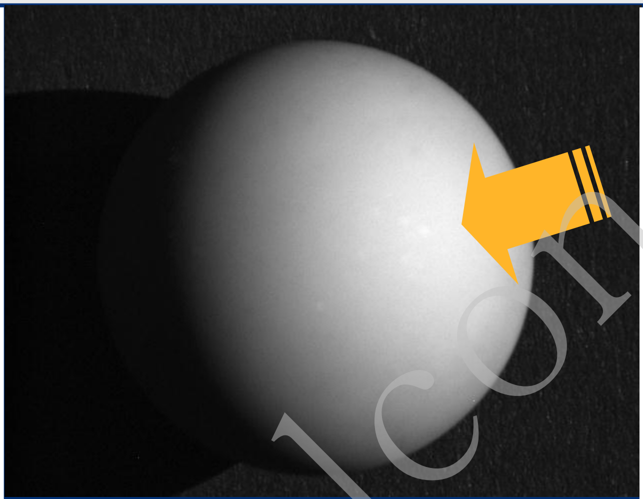
Examples

Variations

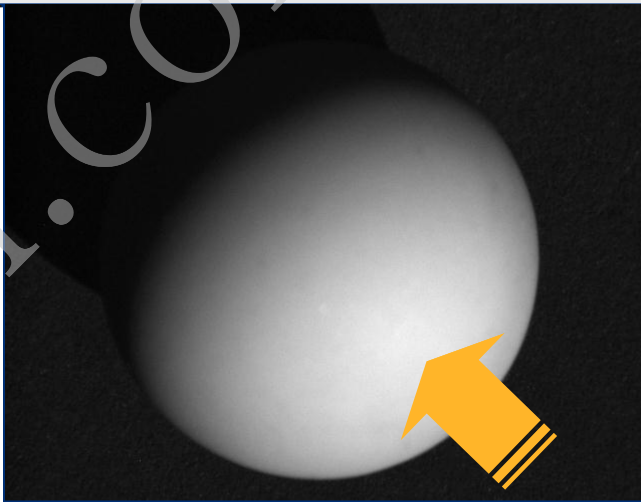
We can provide a procedure for calibration of the tilt and slant angles



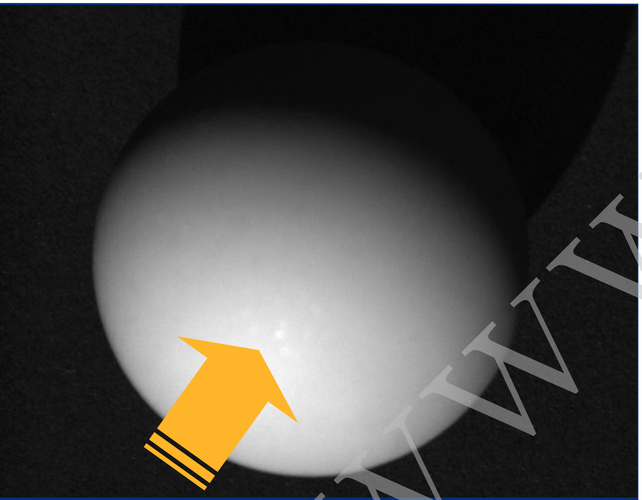
Tilt = 73.4° Slant = 42.0°



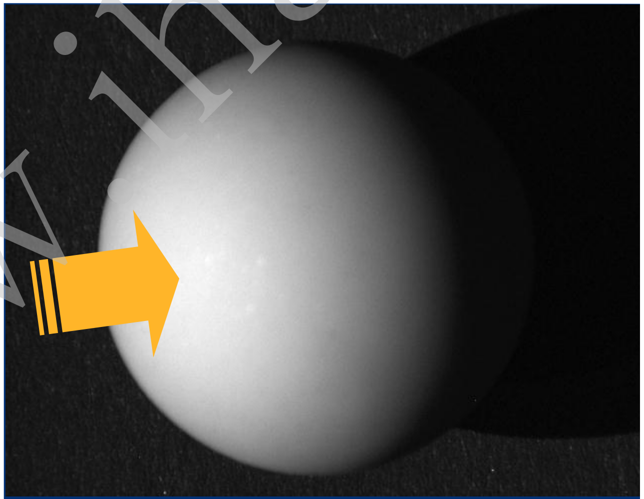
Tilt = 13.9 ° Slant = 40.0°



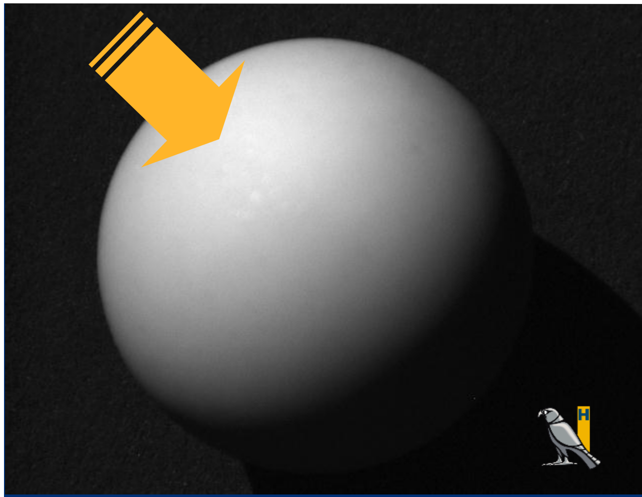
Tilt = -47.4° Slant = 38.8°



Tilt = -113.0° Slant = 40.2°

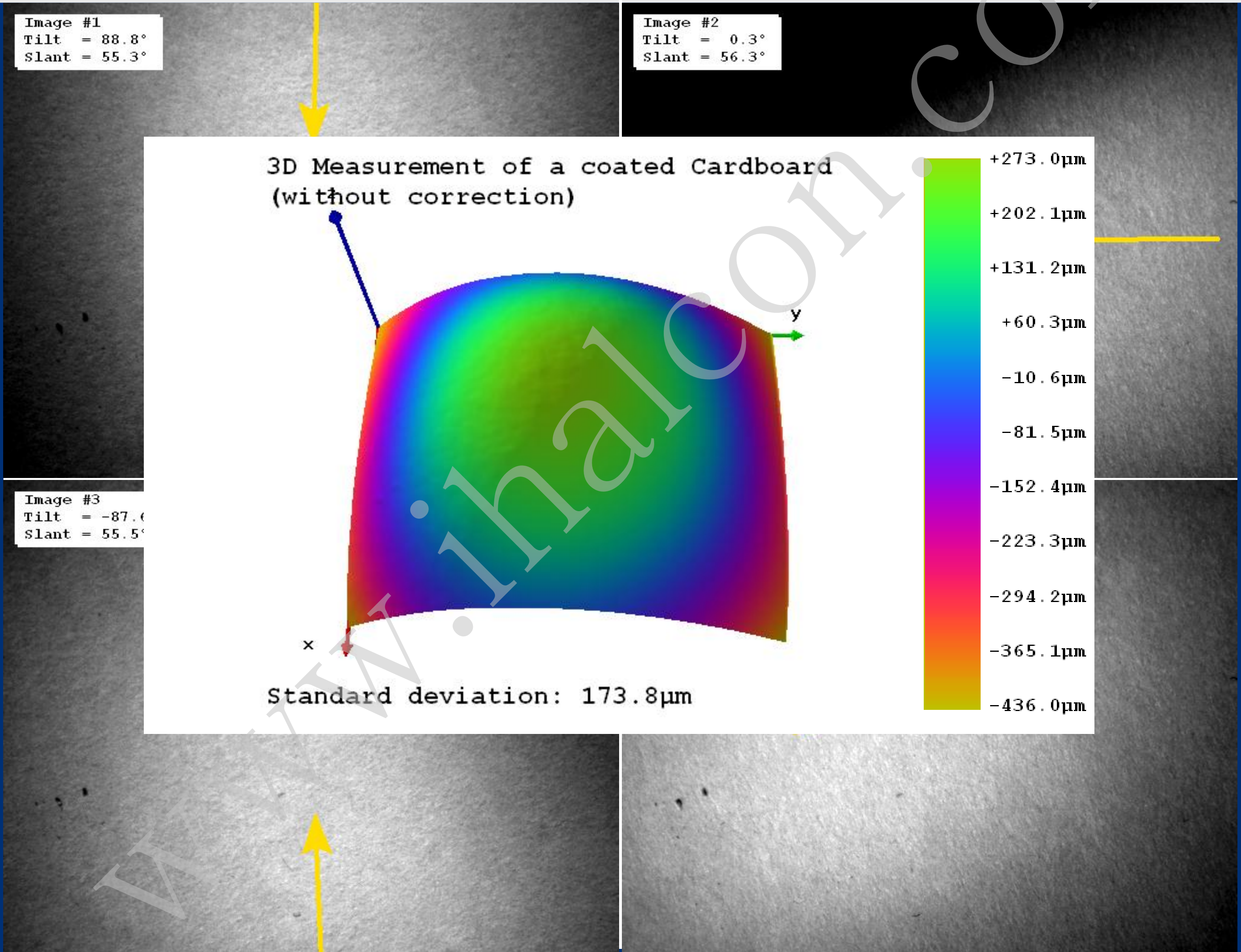


Tilt = -169.6° Slant = 41.2°

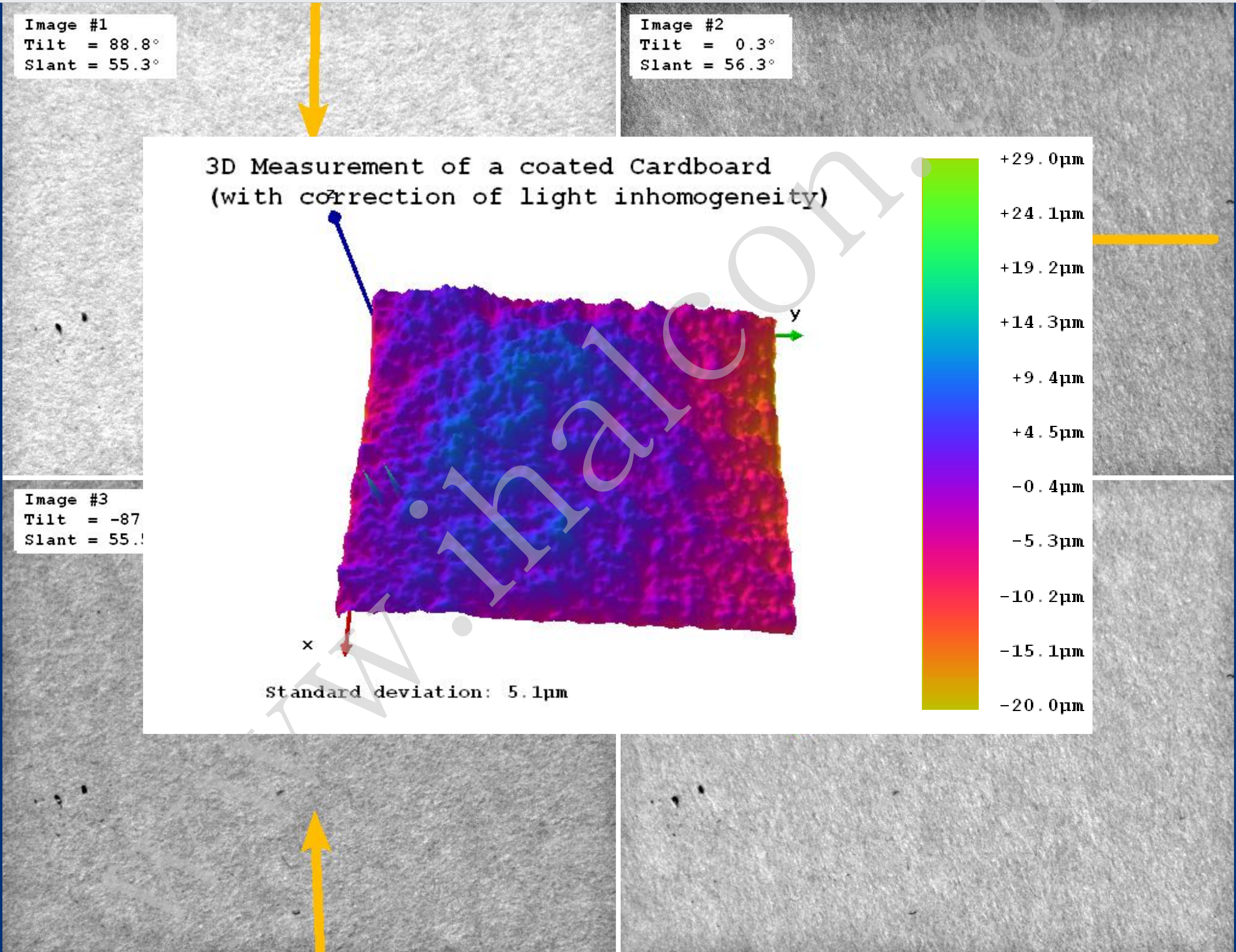


Tilt = 124.0° Slant = 42.7°

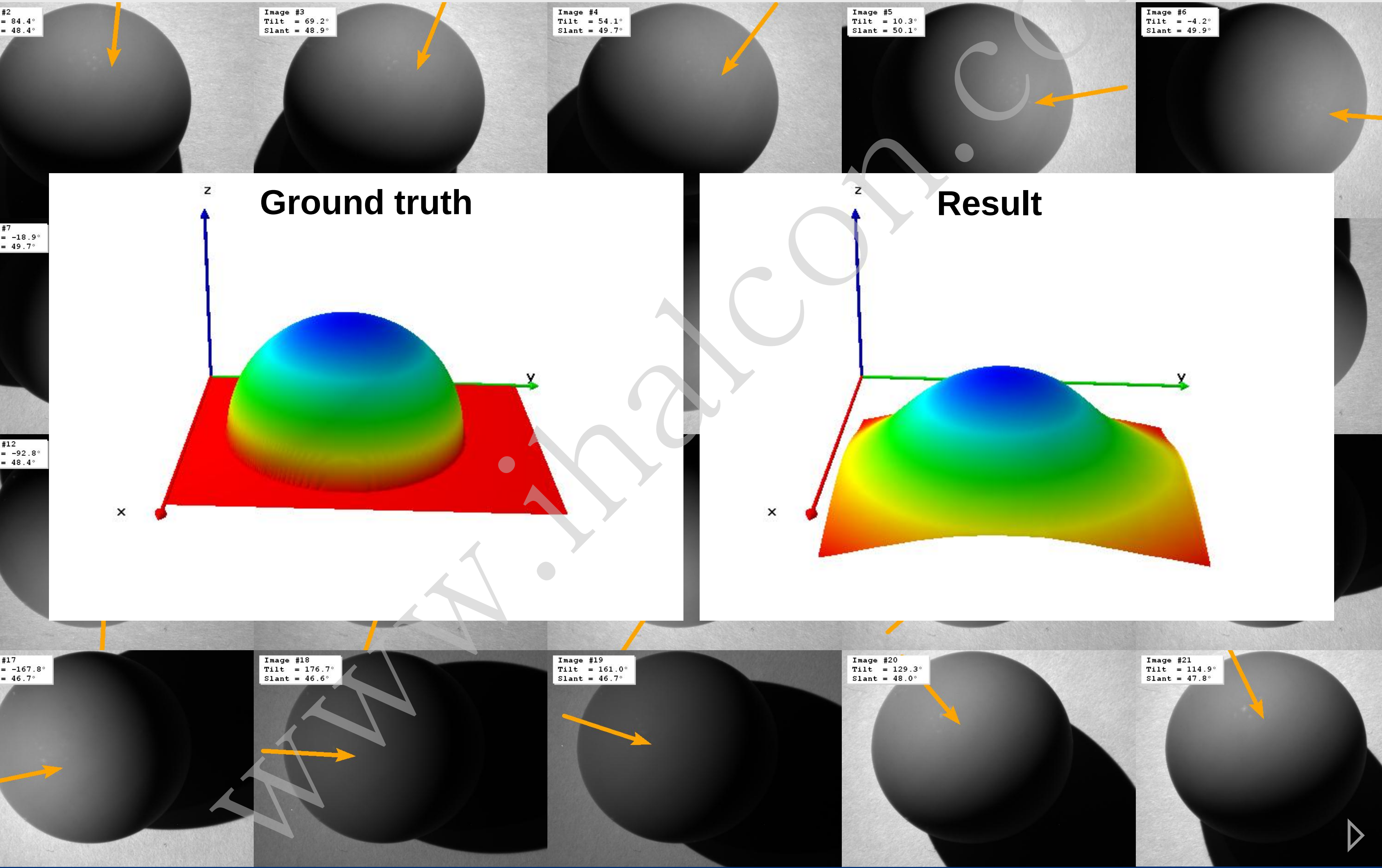
Inhomogeneous light distribution limit the measurement accuracy



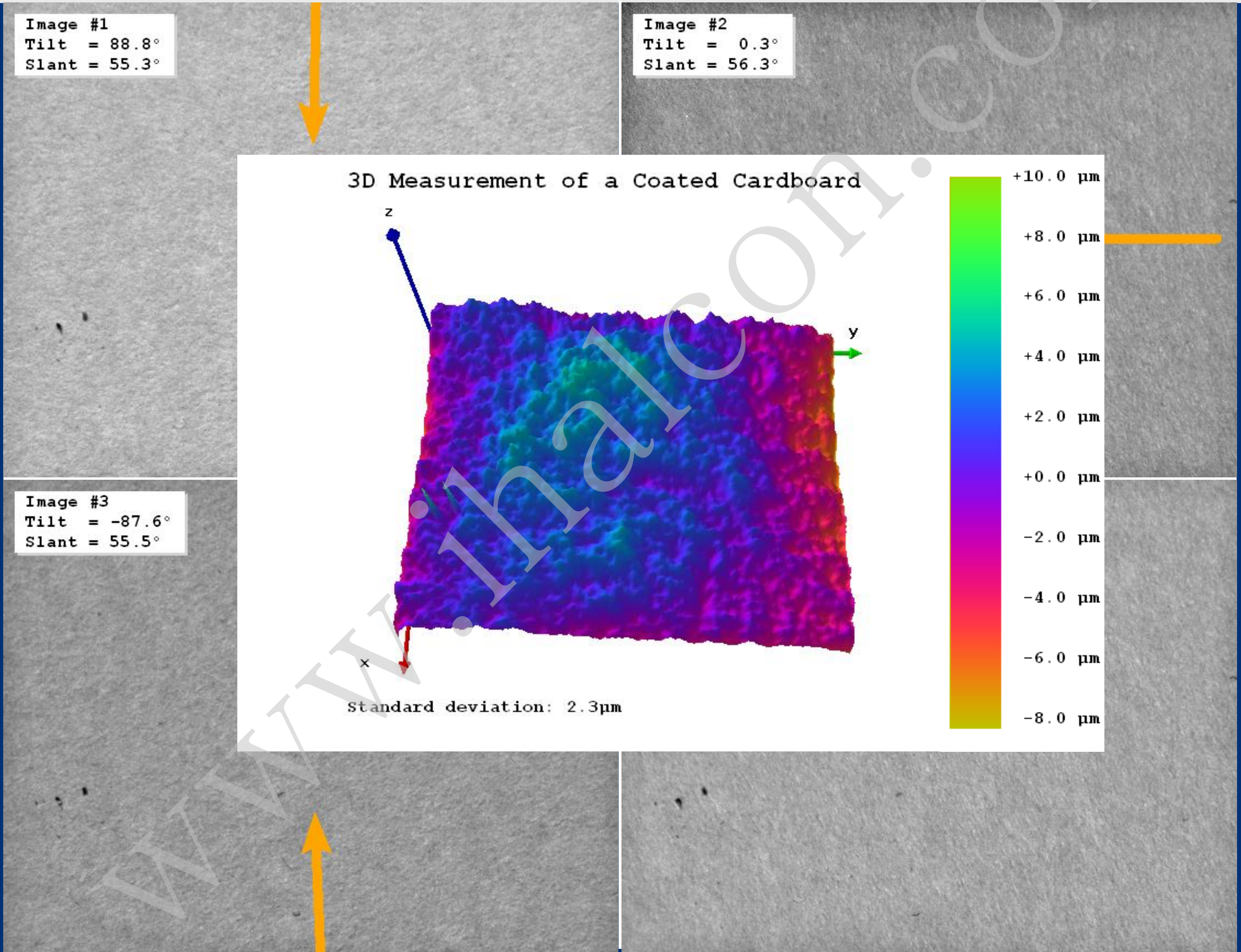
Homogenous light distribution provides high accuracy



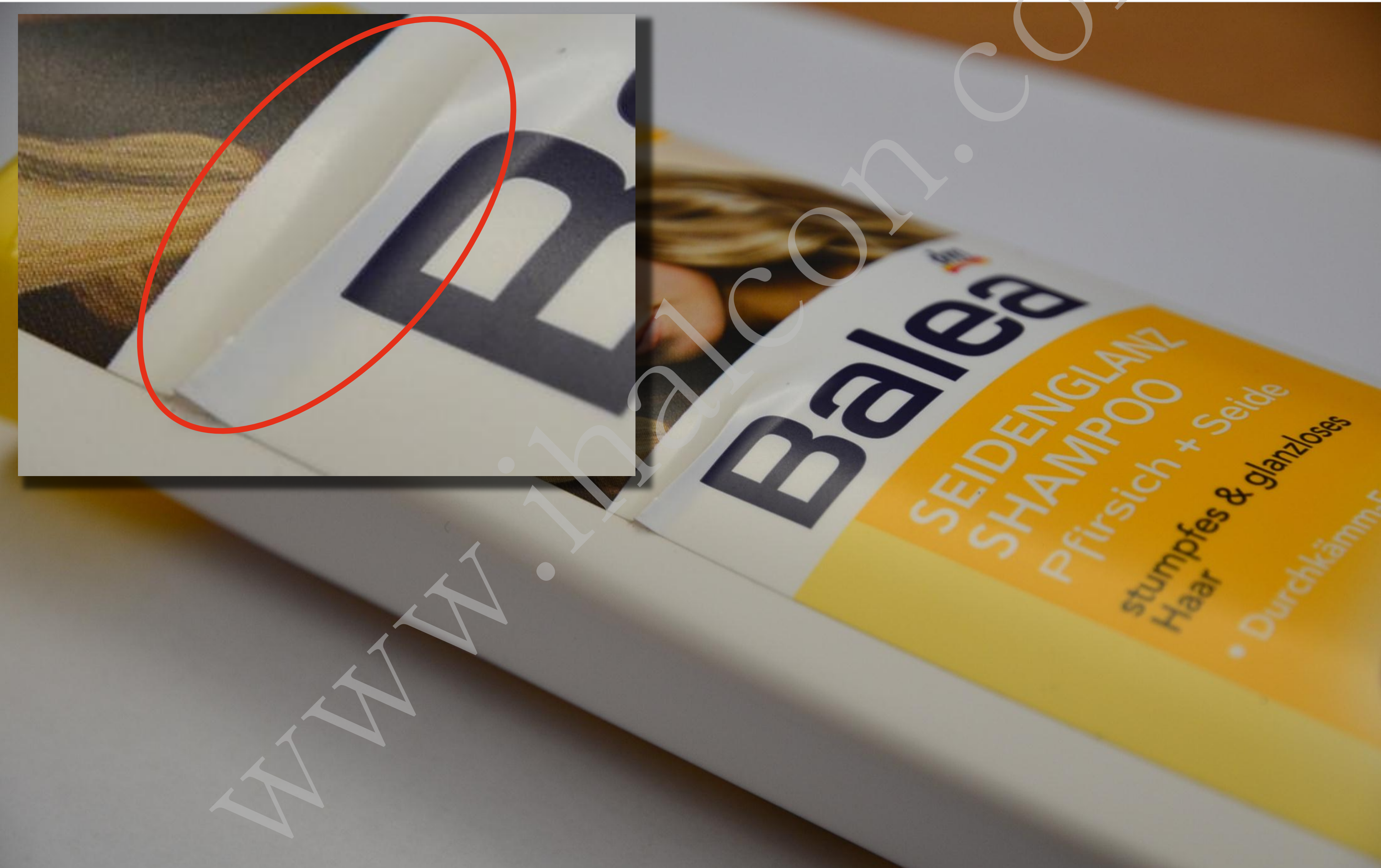
Shadows were a major drawback for correct 3D reconstruction



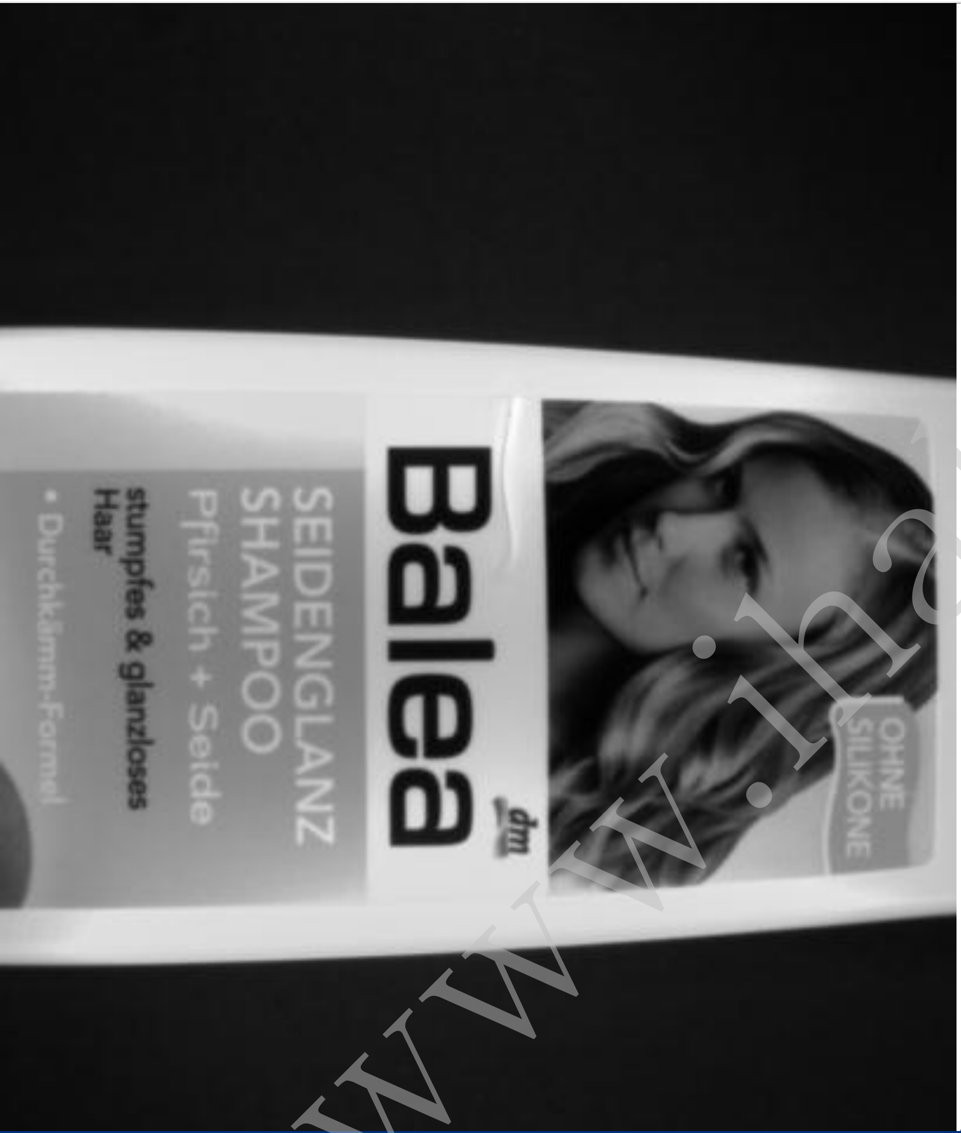
Photometric Stereo is very accurate under optimum conditions



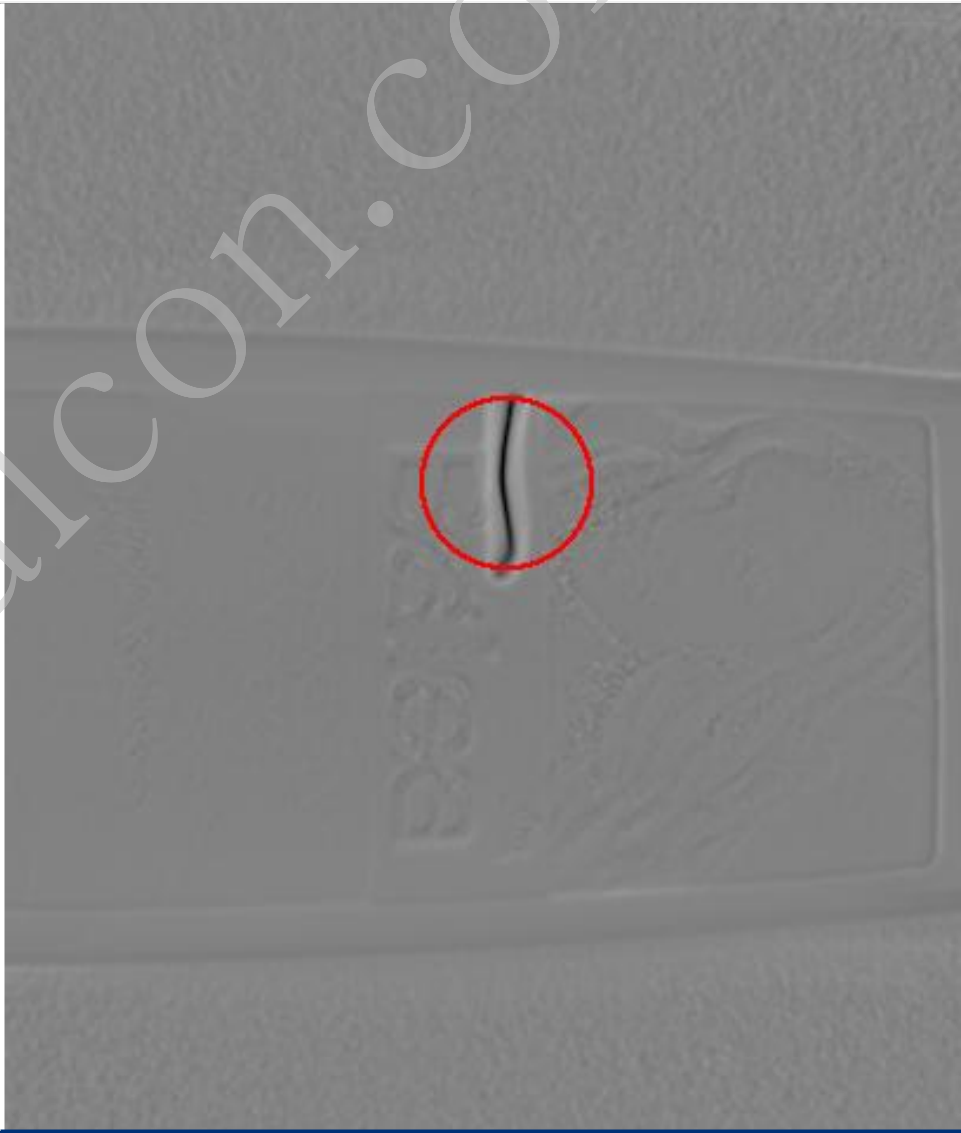
Detect label defects with photometric stereo



Detect label defects with photometric stereo

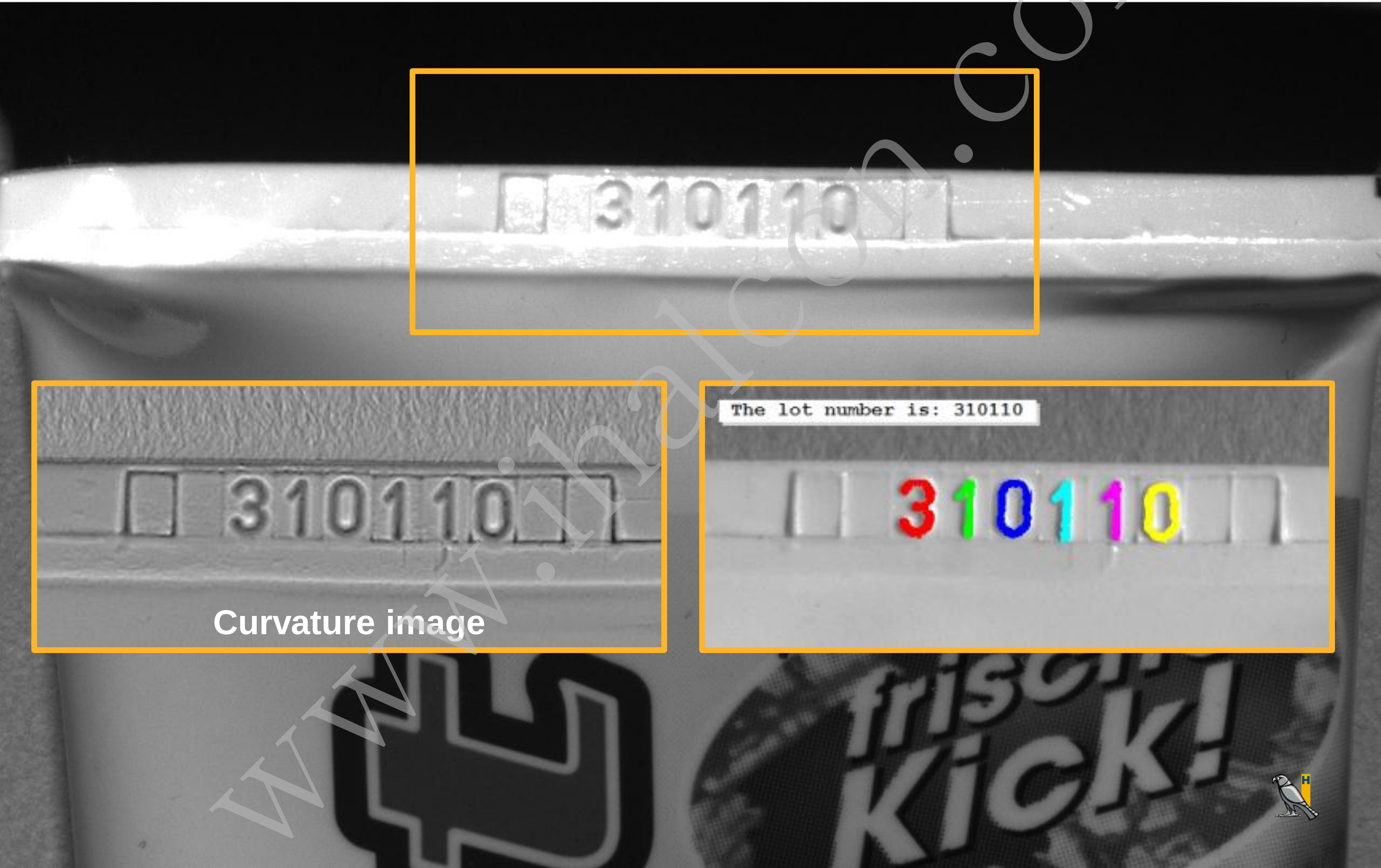


Albedo image



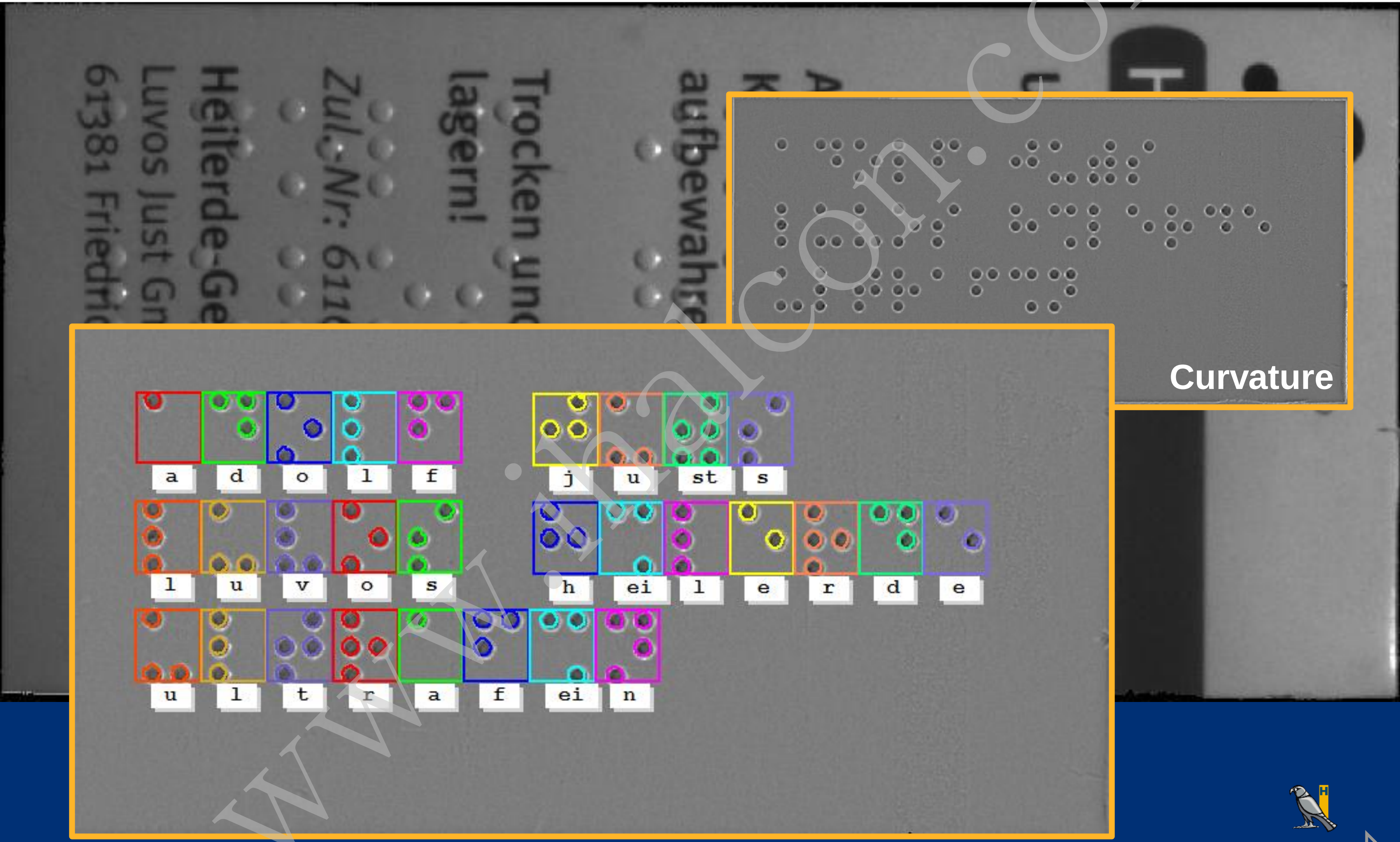
Surface curvature image

Read imprinted text with photometric stereo

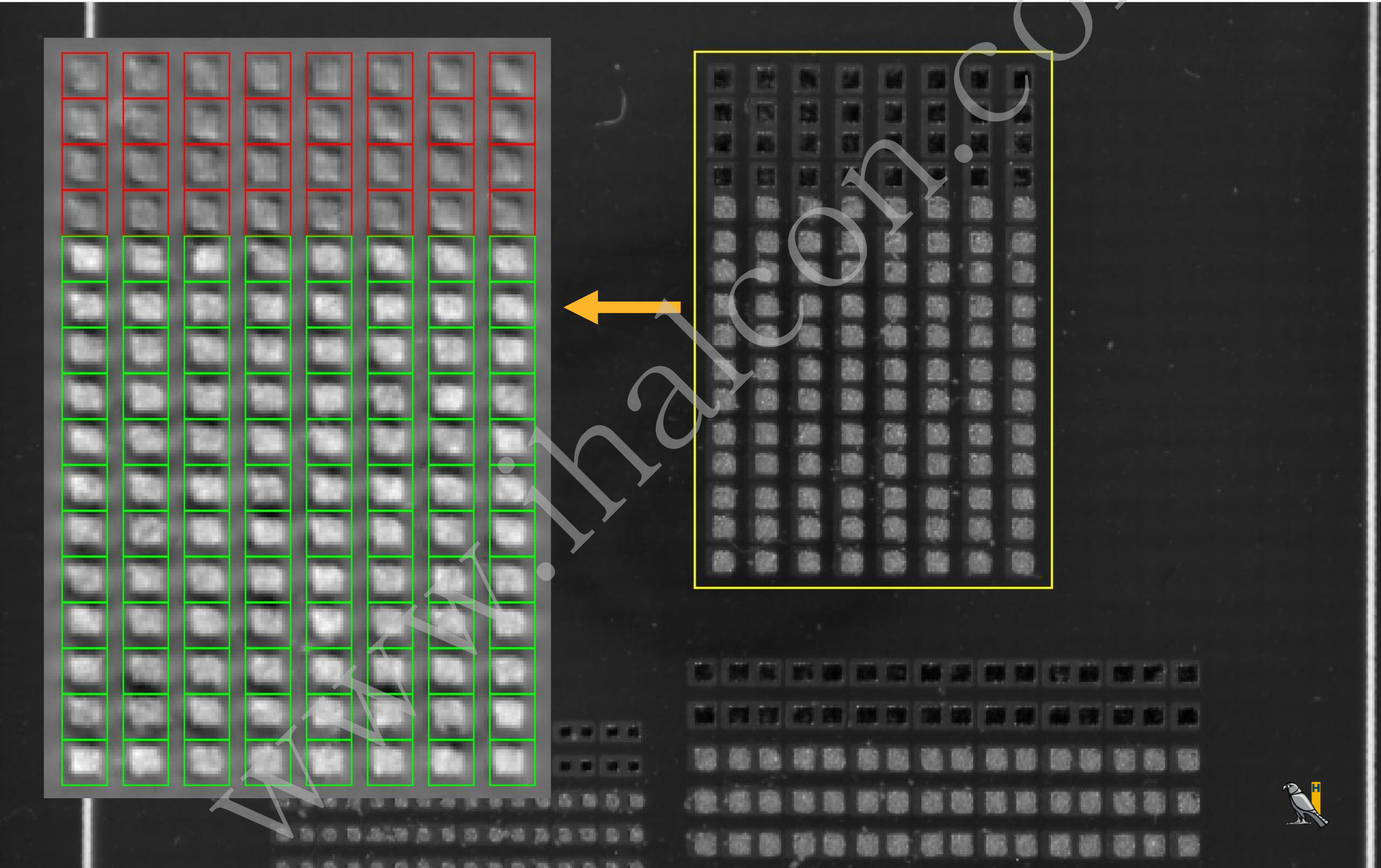


```
The lot number is: 310110
```

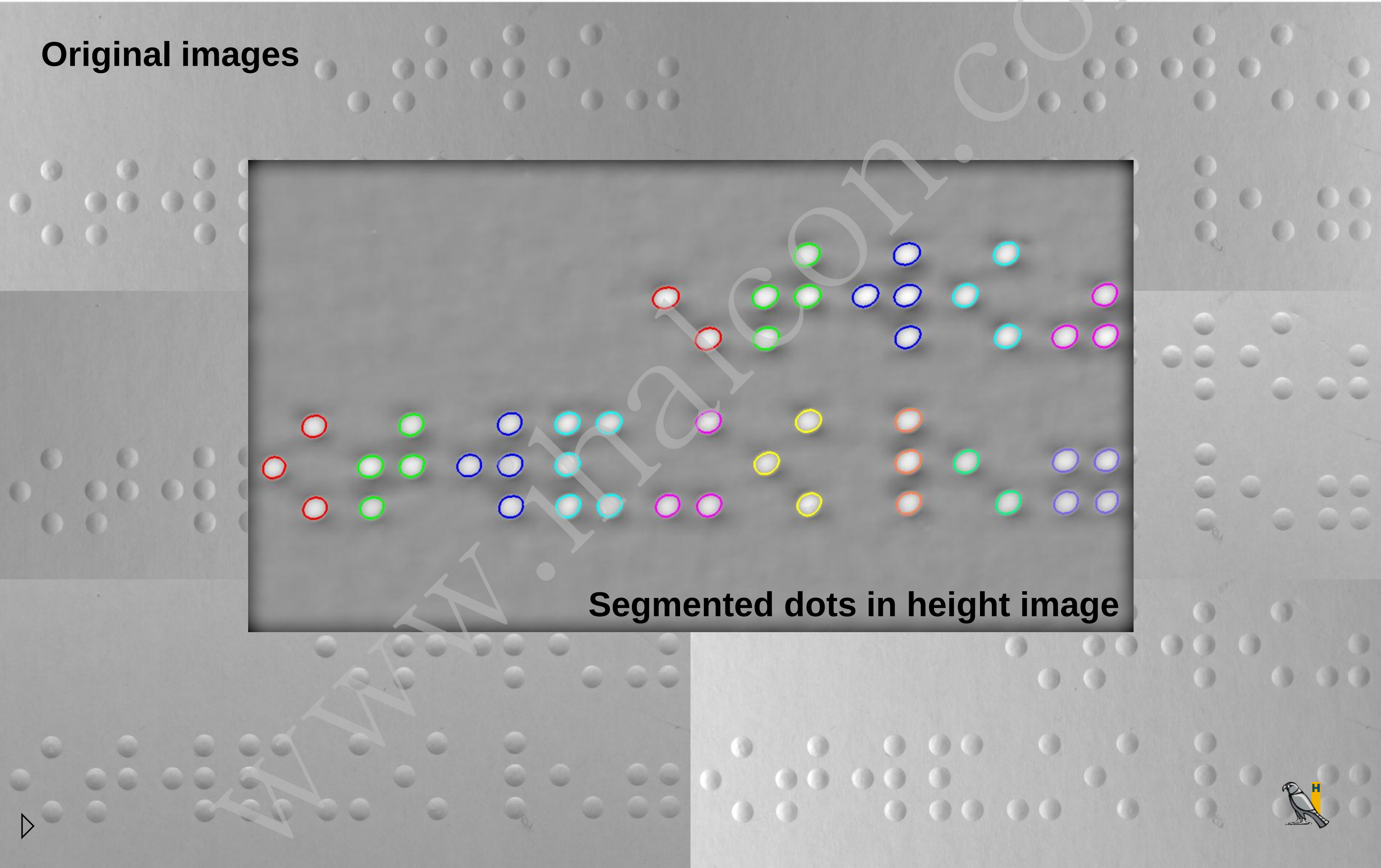
Read Braille with photometric stereo



Check solder paste on pads using photometric stereo



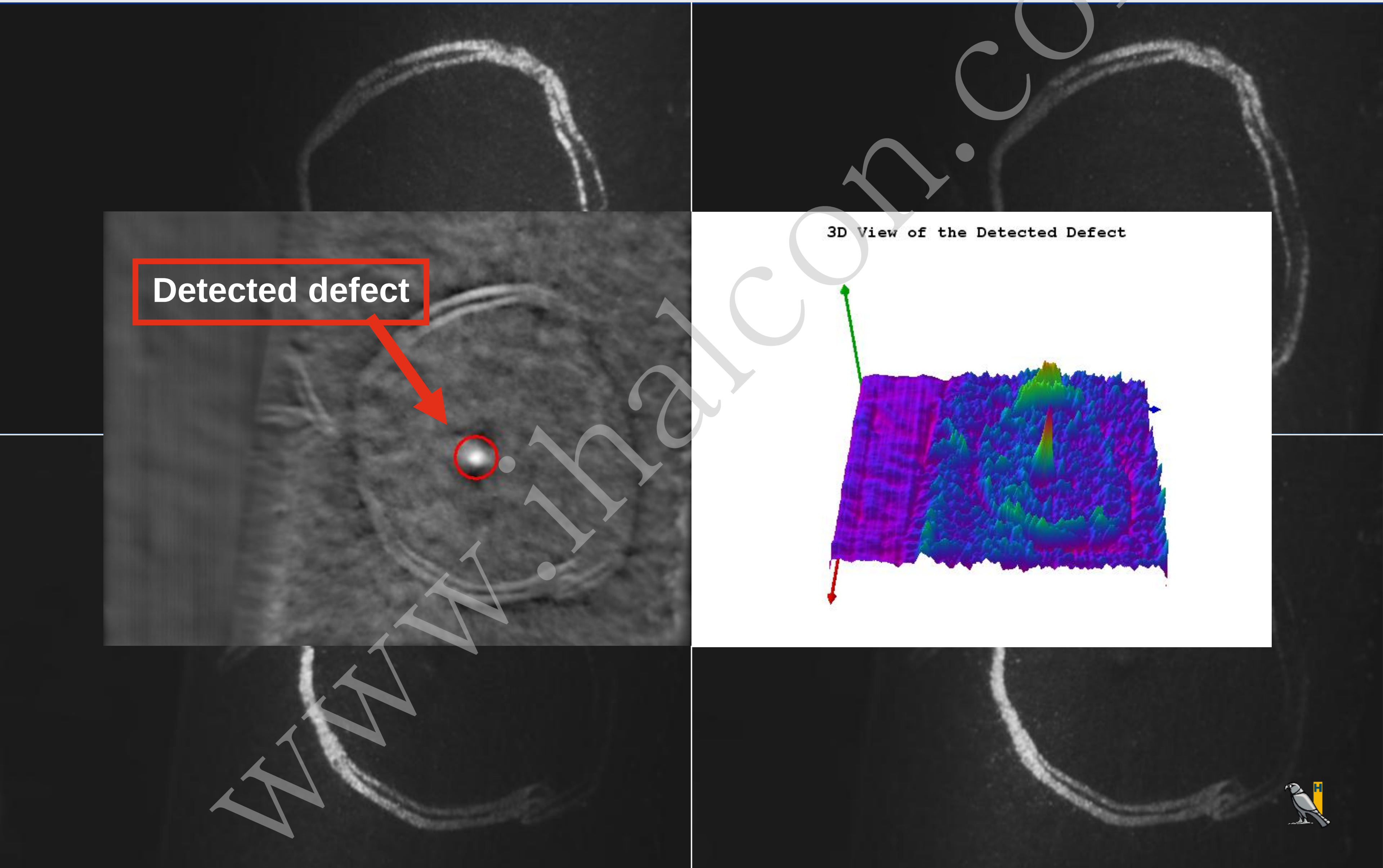
Segment braille dots using photometric stereo



Original images

Segmented dots in height image

Defects on rubber surfaces can be inspected by Photometric Stereo Technique



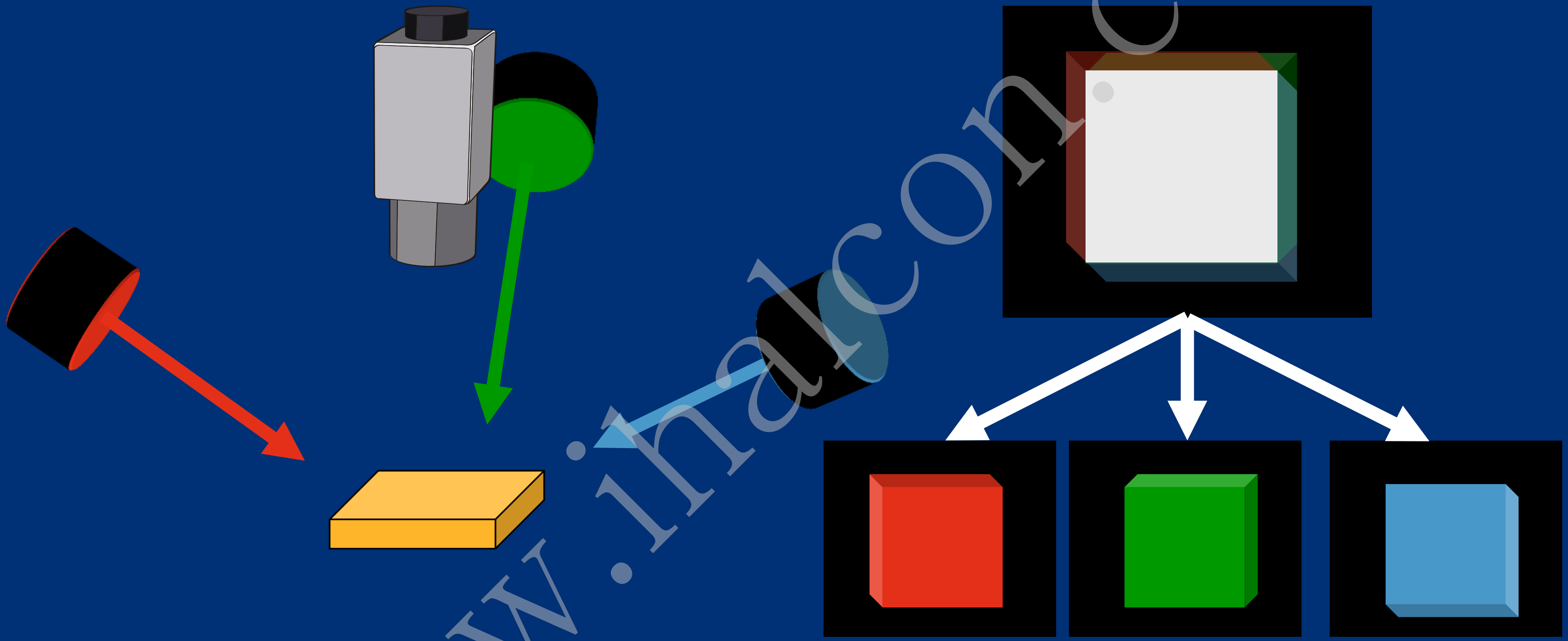
Principle

Handling

Examples

Variations

Reconstruct surface with a single multi-channel image



Illuminate in different colors

Information is stored in RGB image

HALCON

IMAVISION

HALCON can use many 3D Camera to get 3D data directly

Time of flight

Sheet of light

VRmagic

Structured light

+

DFF

Stereo

Sheet of light

Photometric stereo

3D

HALCON

the Power of Machine Vision

MVTEC

3D Object Model

HALCON offers a data structure for 3D data

Object model 3D

Points (X,Y,Z)

Point normals

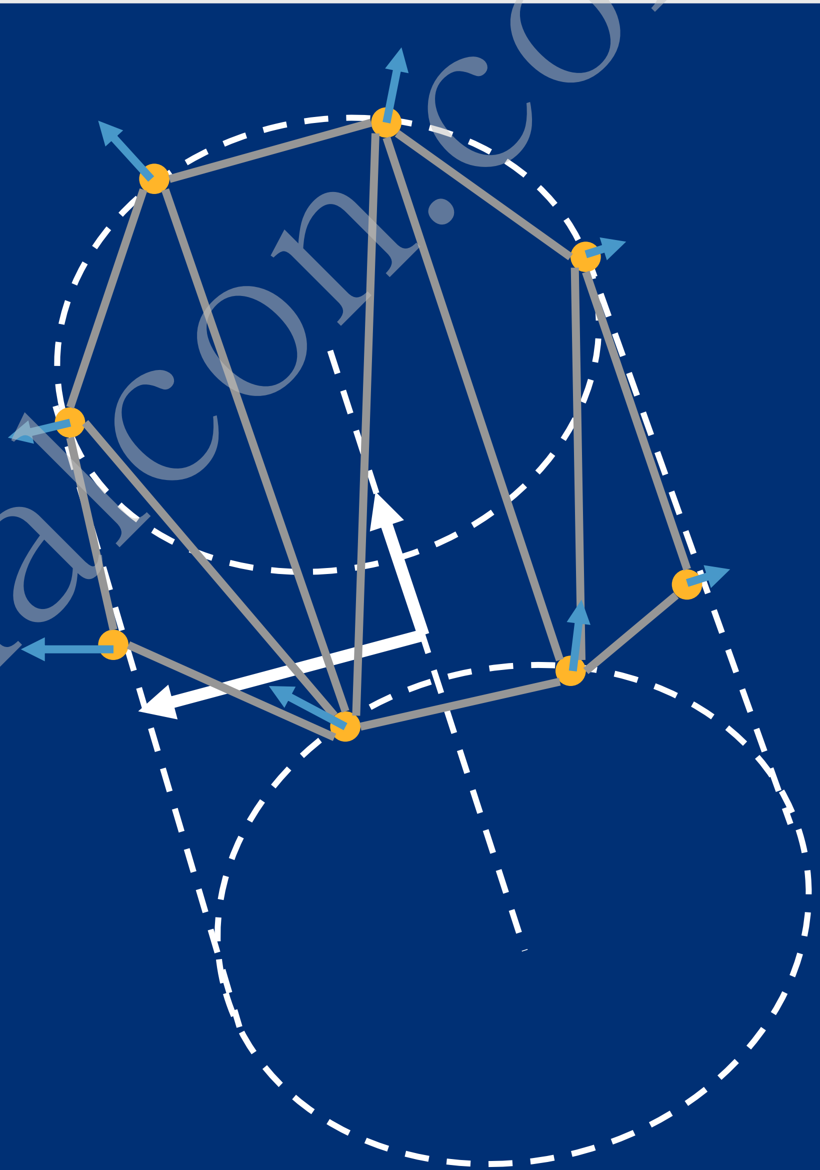
Triangles

Polygons

Primitive parameters

xyz-Mapping

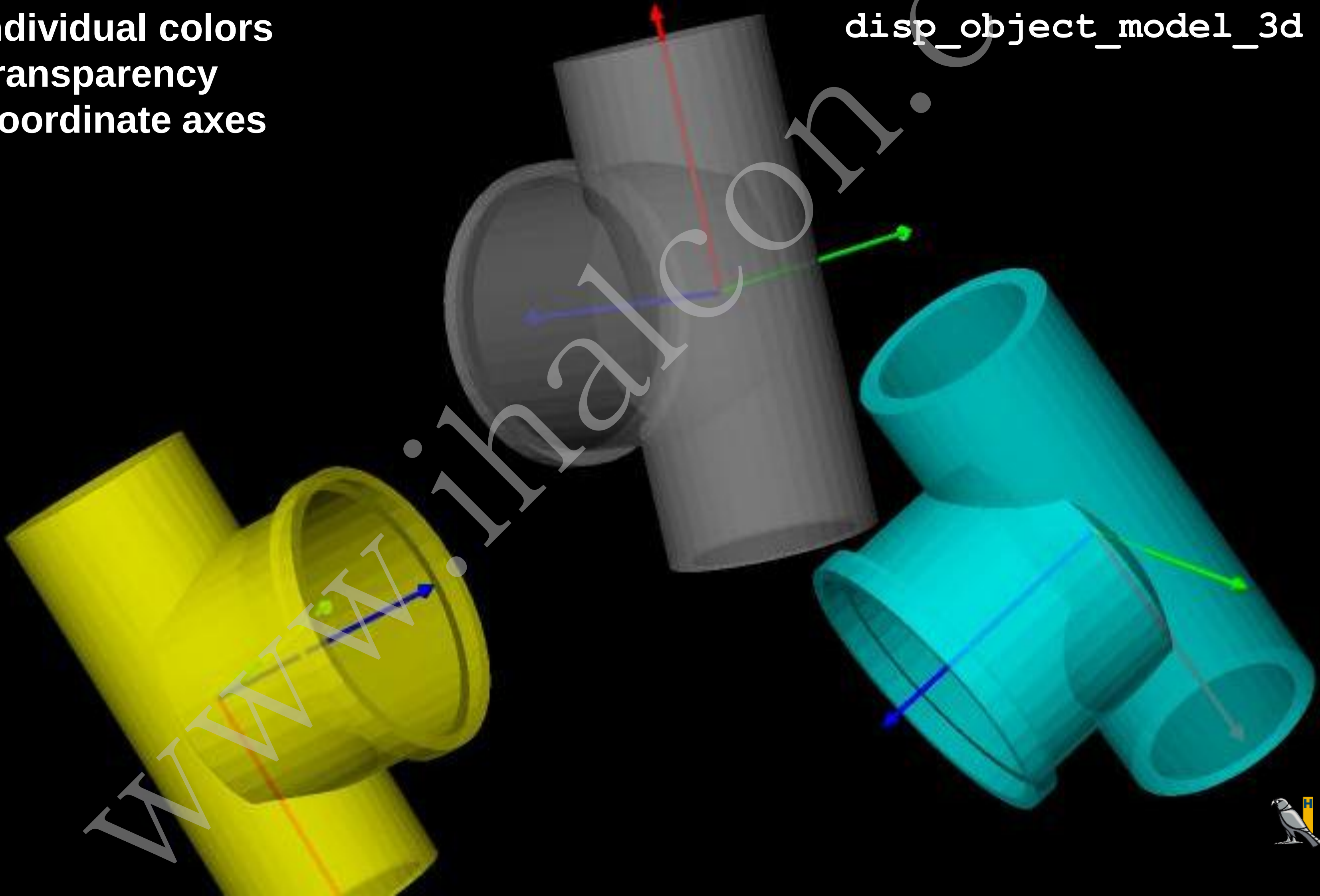
Matching information



Visualize 3D objects with OpenGL

- Individual colors
- Transparency
- Coordinate axes

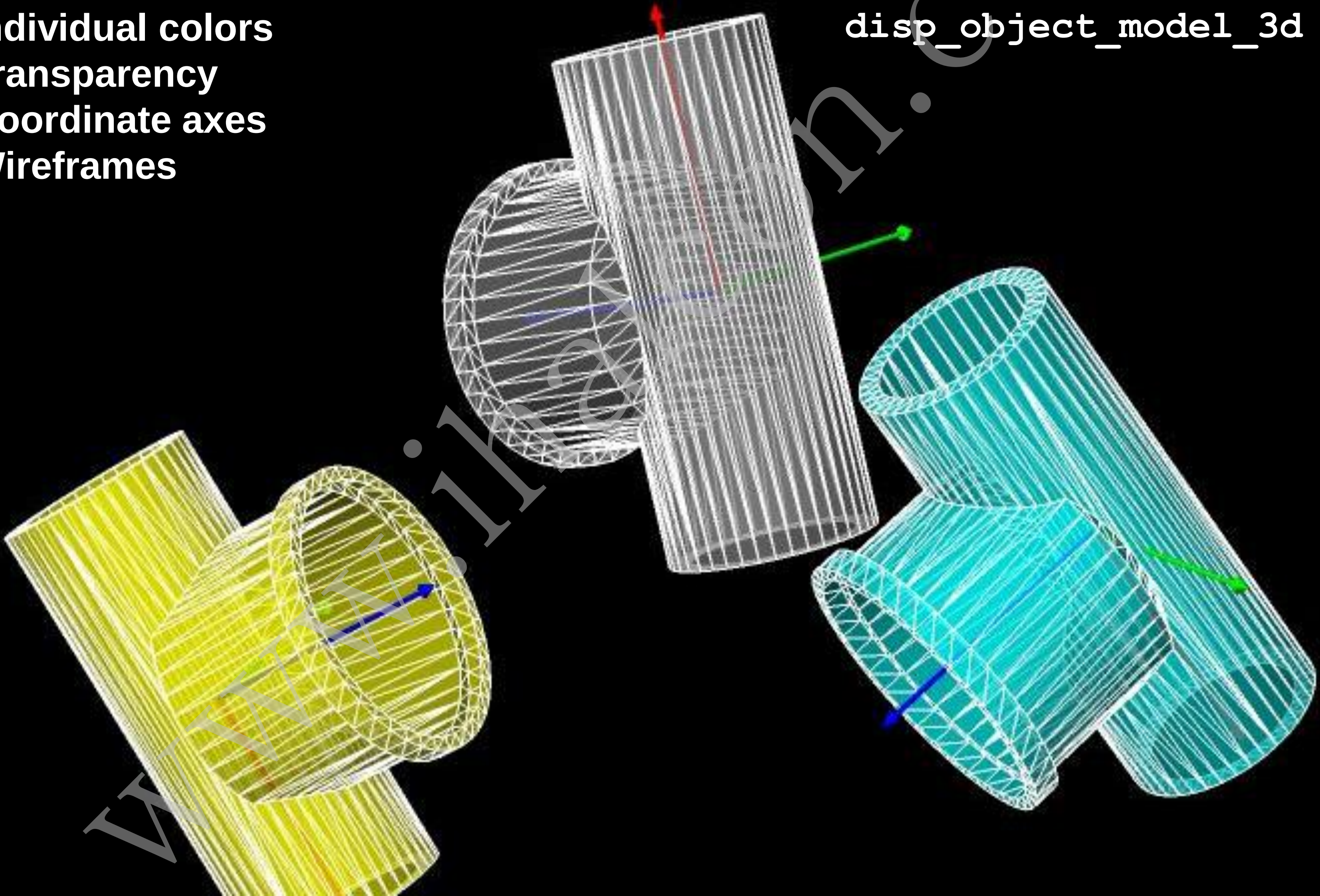
`disp_object_model_3d`



Visualize 3D objects with OpenGL

- Individual colors
- Transparency
- Coordinate axes
- Wireframes

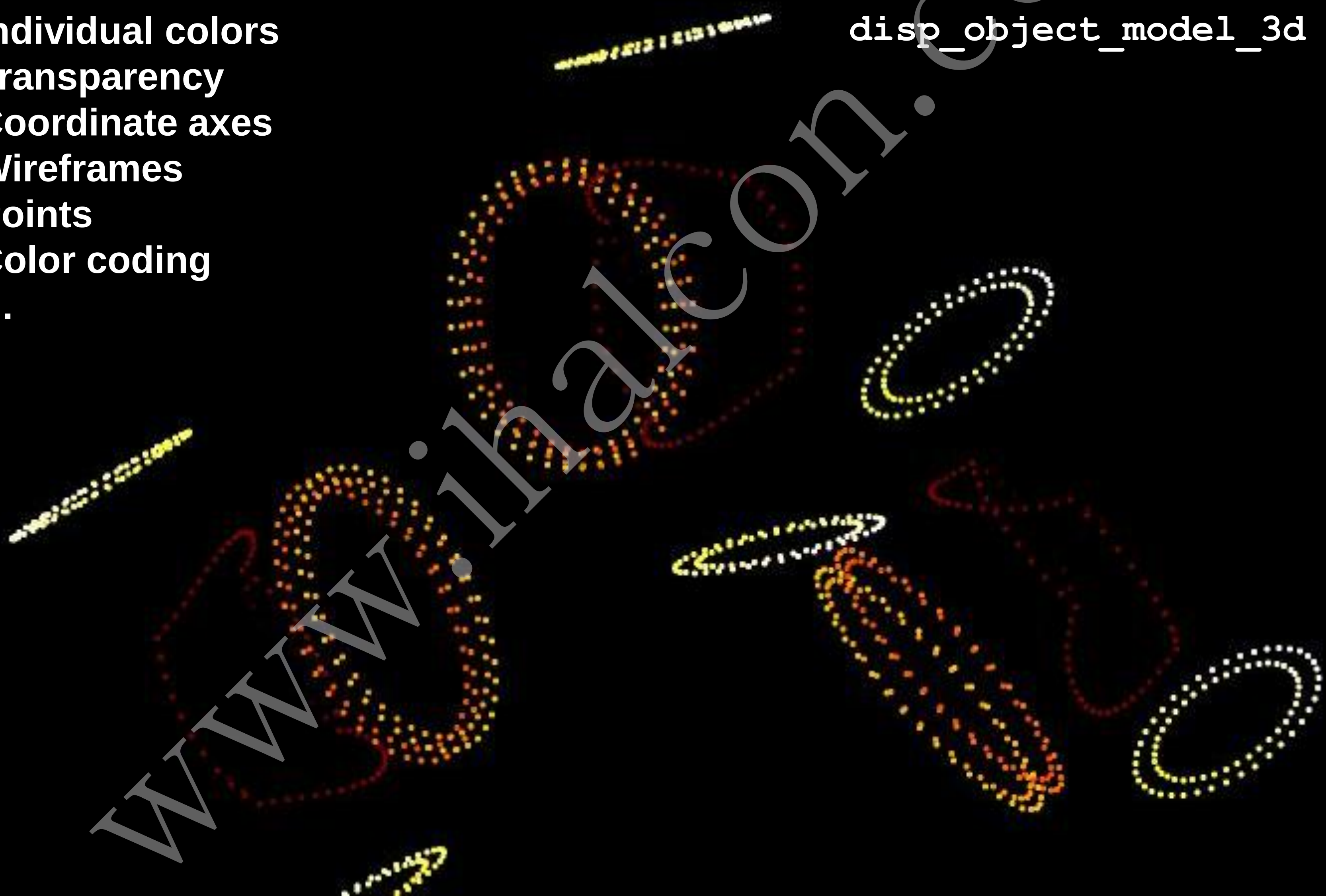
`disp_object_model_3d`



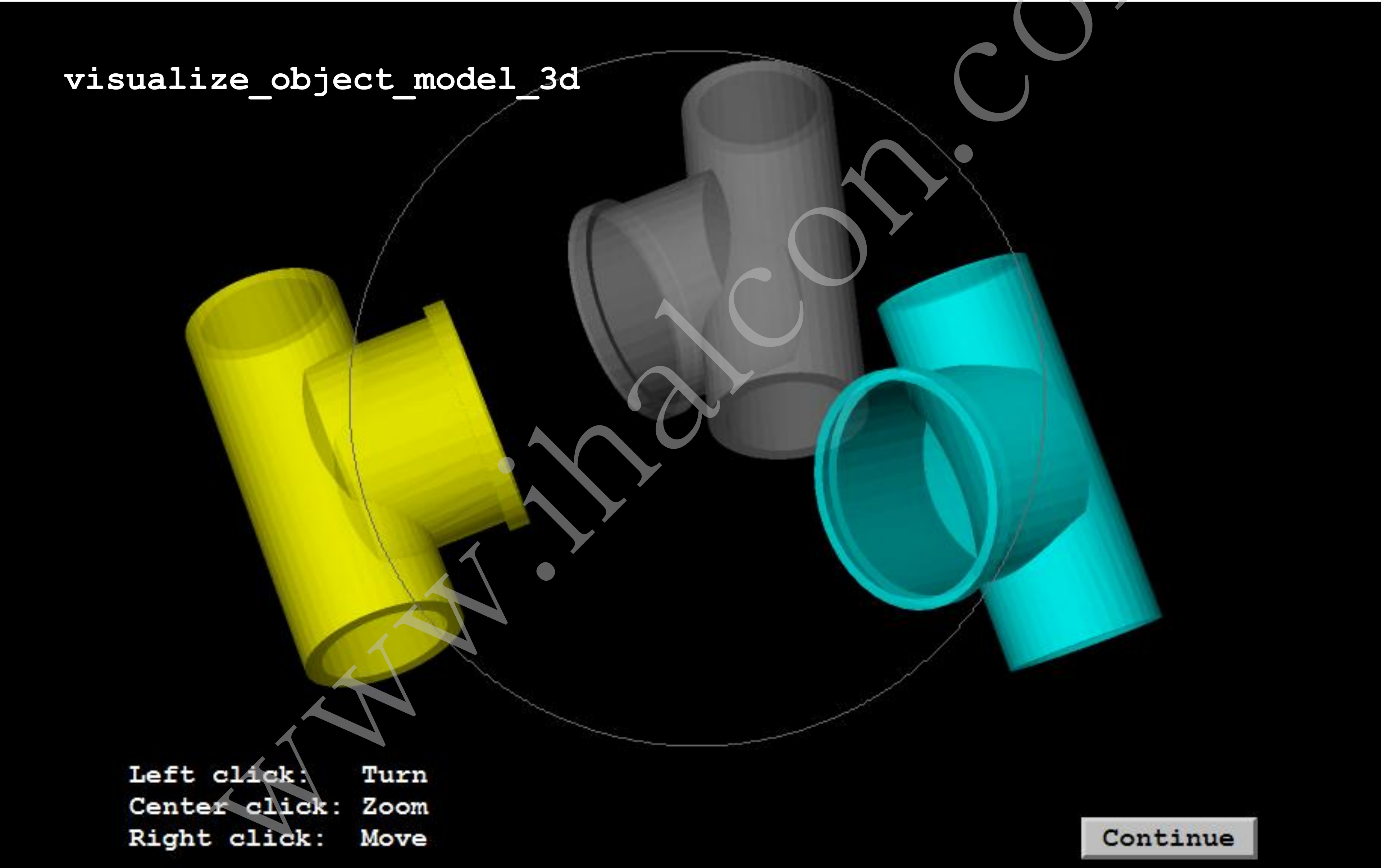
Visualize 3D objects with OpenGL

- Individual colors
- Transparency
- Coordinate axes
- Wireframes
- Points
- Color coding
- …

`disp_object_model_3d`



Visualize 3D objects interactively

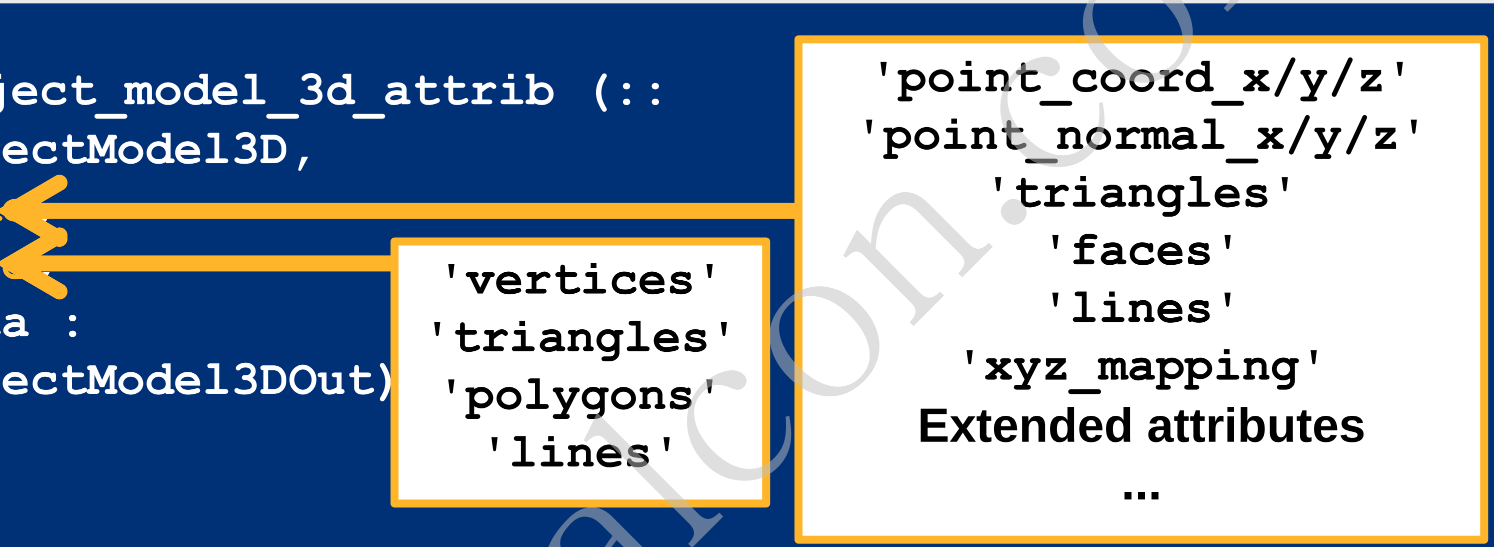


```
visualize_object_model_3d
```

```
Left click:   Turn
Center click: Zoom
Right click:  Move
```

Set 3D object model attributes manually

- `set_object_model_3d_attrib (::`
  - ▶ `ObjectModel3D,`
  - ▶ `Name,`
  - ▶ `Type,`
  - ▶ `Data :`
  - ◀ `ObjectModel3DOut)`



'point_coord_x/y/z'
'point_normal_x/y/z'
'triangles'
'faces'
'lines'
'xyz_mapping'
Extended attributes
...

'vertices'
'triangles'
'polygons'
'lines'

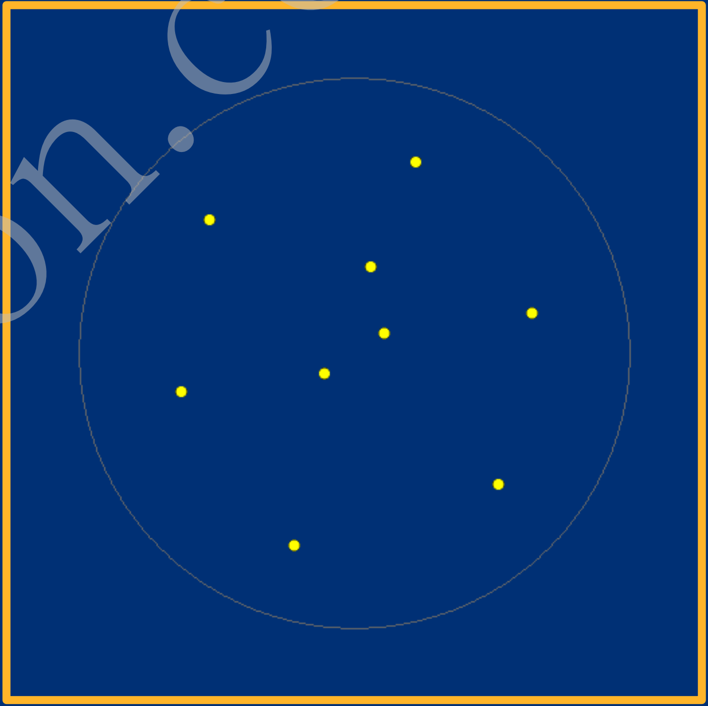
- `set_object_model_3d_attrib_mod (::`
  - ▶ `ObjectModel3D,`
  - ▶ `Name,`
  - ▶ `Type,`
  - ▶ `Data : )`

Create an object model 3D manually

- gen_empty_object_model_3d (Object3D)

- X := [0.5,0,1,1,0,0,1,1,0]
- Y := [0,1,1,1,1,0,0,0,0]
- Z := [0.5,0,0,1,1,0,0,1,1]

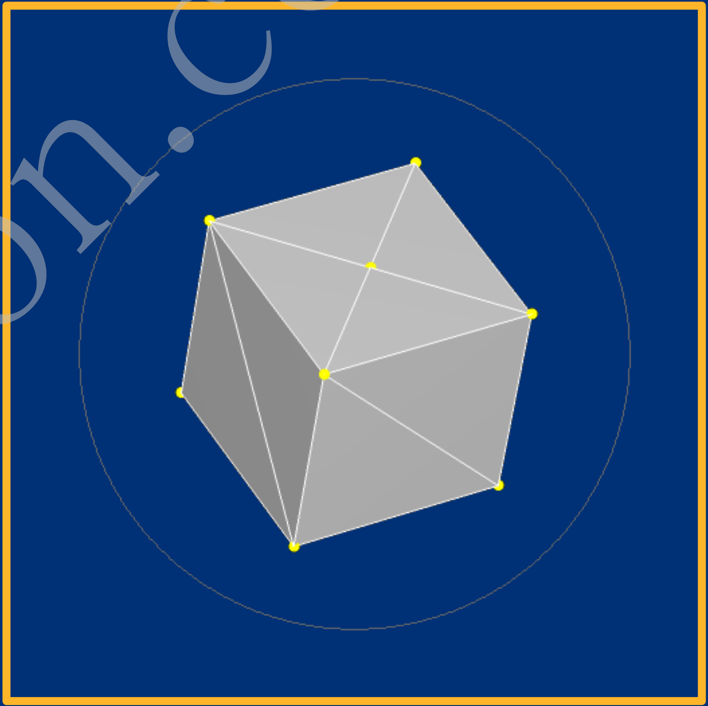
- set_object_model_3d_attrib_mod (\
  Object3D, \
-   ['point_coord_x','point_coord_y','point_coord_z'], \
-   [], [X, Y, Z])



Triangulate points manually

- T1 := [0,8,5]
- T2 := [0,7,8]
- T3 := [0,6,7]
- T4 := [0,5,6]
- …
- T14 := [2,4,1]

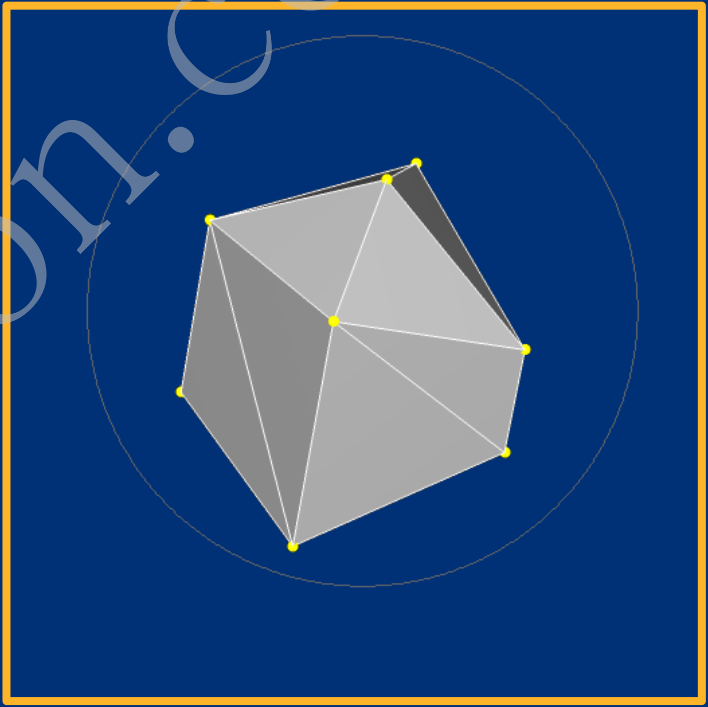
```
set_object_model_3d_attrib ( \
  Object3D, \
  'triangles', [], \
  [T1,T2,...,T14], \
  Object3D2)
```



Modify 3d points

- `Y [0,3,6,7] := [-1,0.3,-.8,-.3]`

-
  ```
  set_object_model_3d_attrib_mod ( \
    Object3D2, 'point_coord_y', [], Y)
  ```



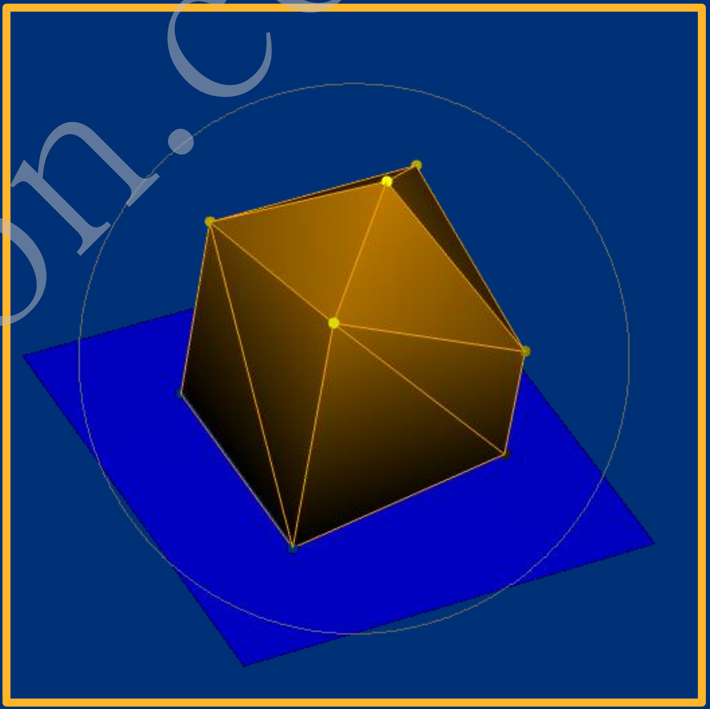
Attributes of Object Model 3D: Add Attributes for Visualization

- PlaneX := [0,.55,0,90,0,0,0]
- PlaneY := [-1,-1,1,1]
- PlaneX := [-1,1,1,-1]

- gen_plane_object_model_3d (\
-    [PlaneX,PlaneY,PlaneZ], Plane3D)

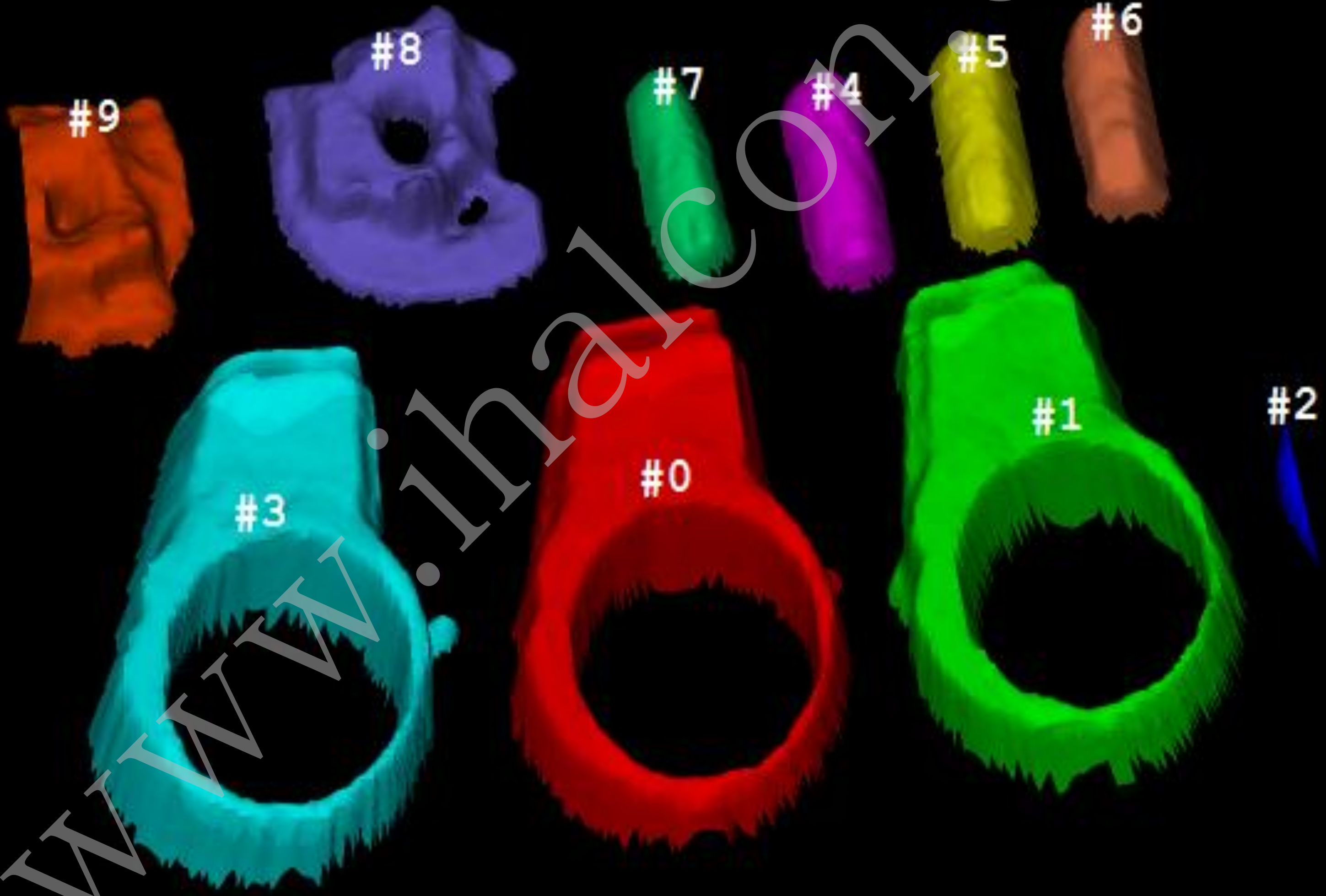
- distance_object_model_3d (\
  Object3D2, Plane3D, [], 0, [], [])

- VisualParamNames := ['intensity']
- VisualParamValues := ['&distance']

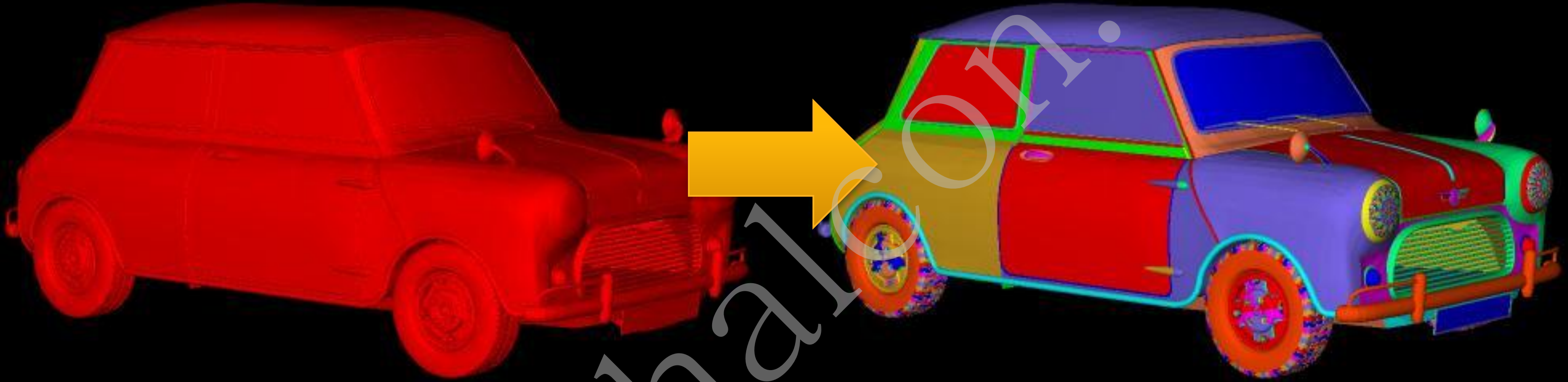


Calculate connected components

```
connection_object_model_3d()
```



Calculate connected components



Select points of a 3D object model by its features

```
select_points_object_model_3d()
```

Possible features:

- Position
- Normals
- Mapping position
- Extended attributes

Select 3D object models by features

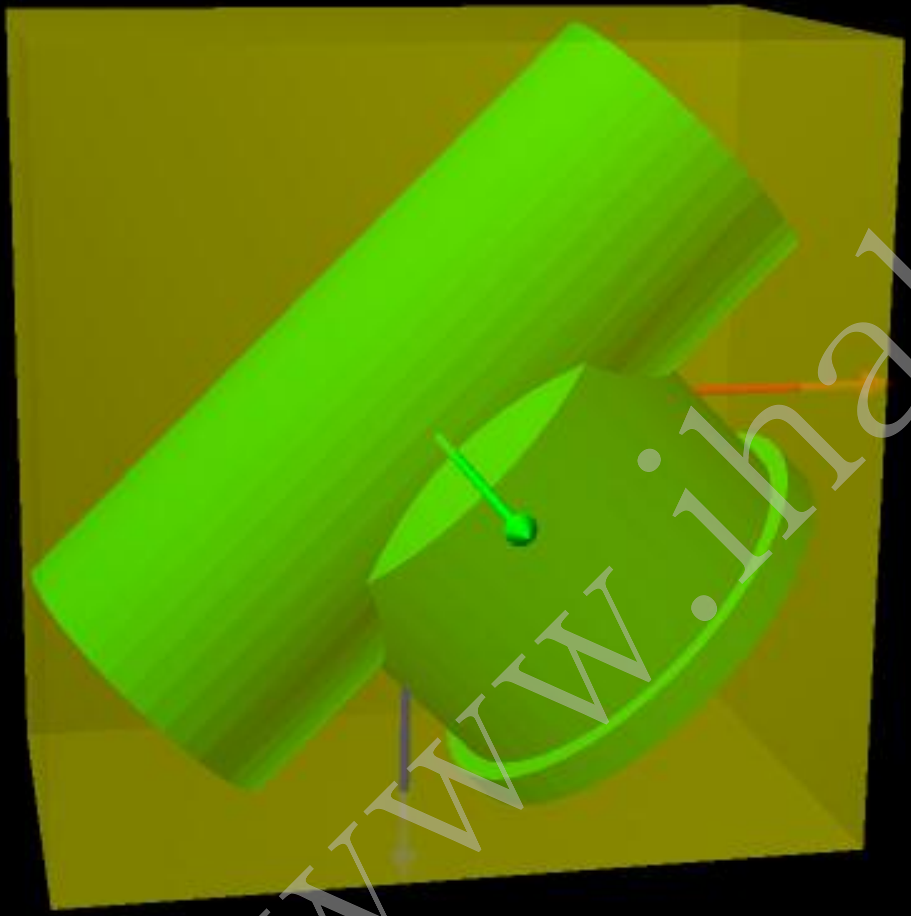
```
select_object_model_3d()
```

Possible features:

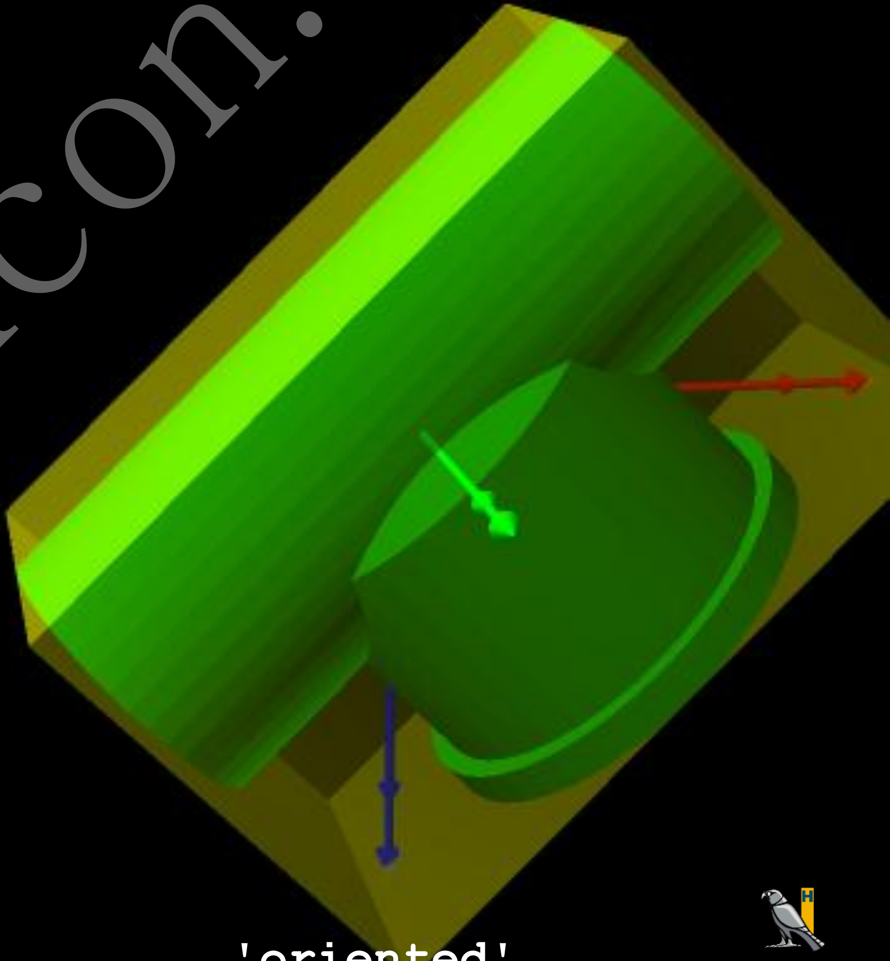
- Position
- Diameter
- Volume
- Surface area
- Moments
- …

Calculate the bounding box of 3D object models

```
smallest_bounding_box_object_model_3d
```

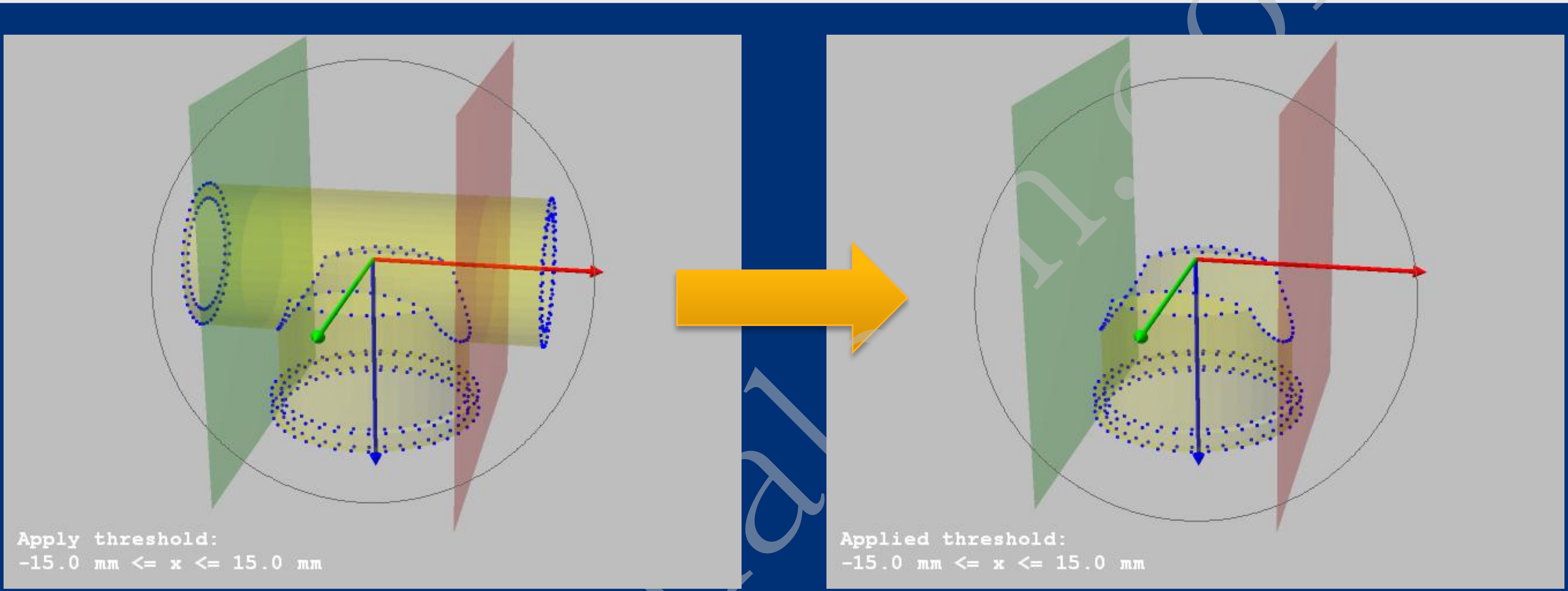


'axis_aligned'



'oriented'

Select points of a 3D object model by its features



Apply threshold:
-15.0 mm <= x <= 15.0 mm

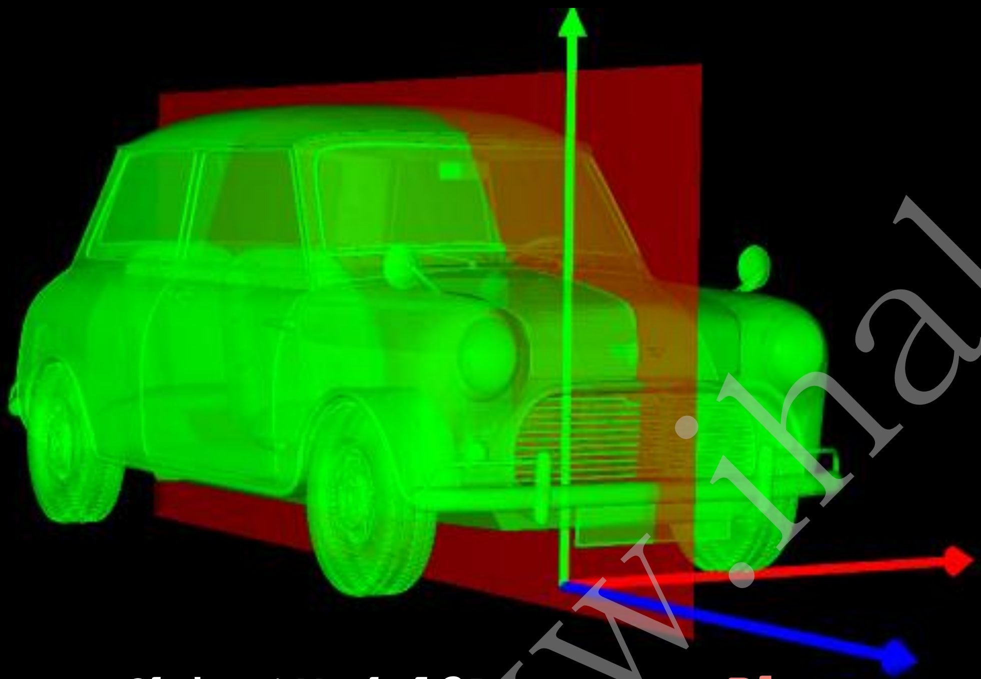
Applied threshold:
-15.0 mm <= x <= 15.0 mm

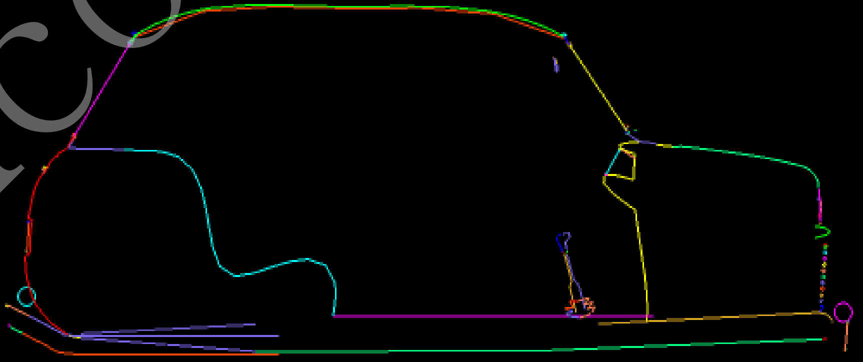
- select_points_object_model_3d (::
- ▶ ObjectModel3D,
- ▶ Attrib,
- ▶ MinValue, MaxValue,
- ◀ ObjectModel3DThresholded)

'point_coord_x/y/z'
'point_normal_x/y/z'
'mapping_row'
'mapping_col'
Extended attributes
...

Calculate slices with planes of 3D objects

```
intersect_plane_object_model_3d()
```

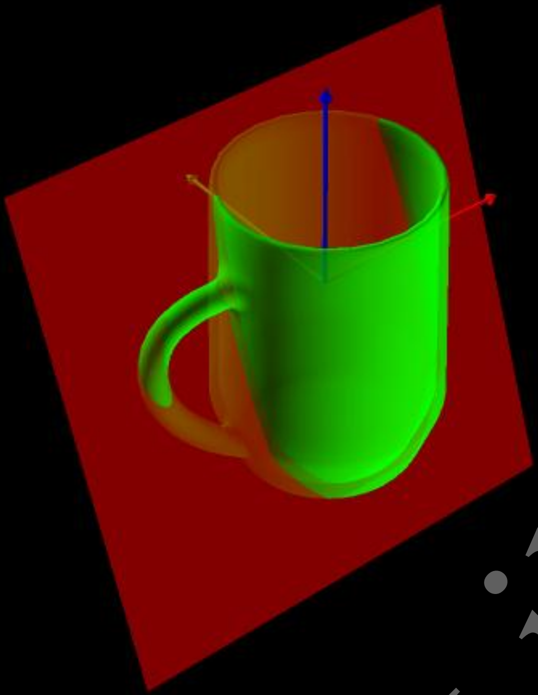


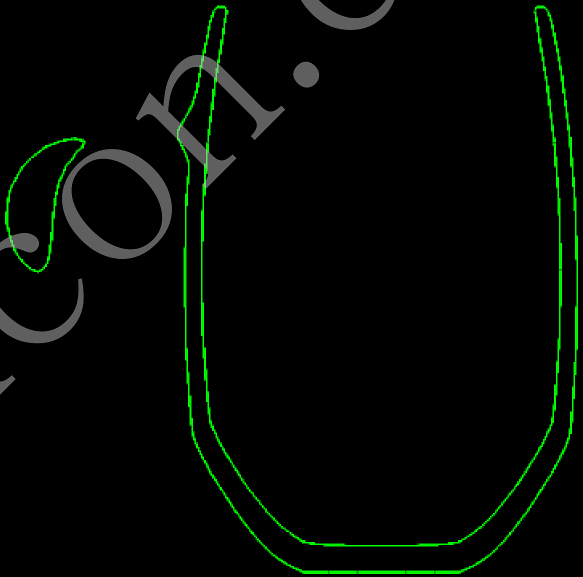


ObjectModel3D Plane ObjectModel3DIntersection

[0,0,0,0,90,0,0]

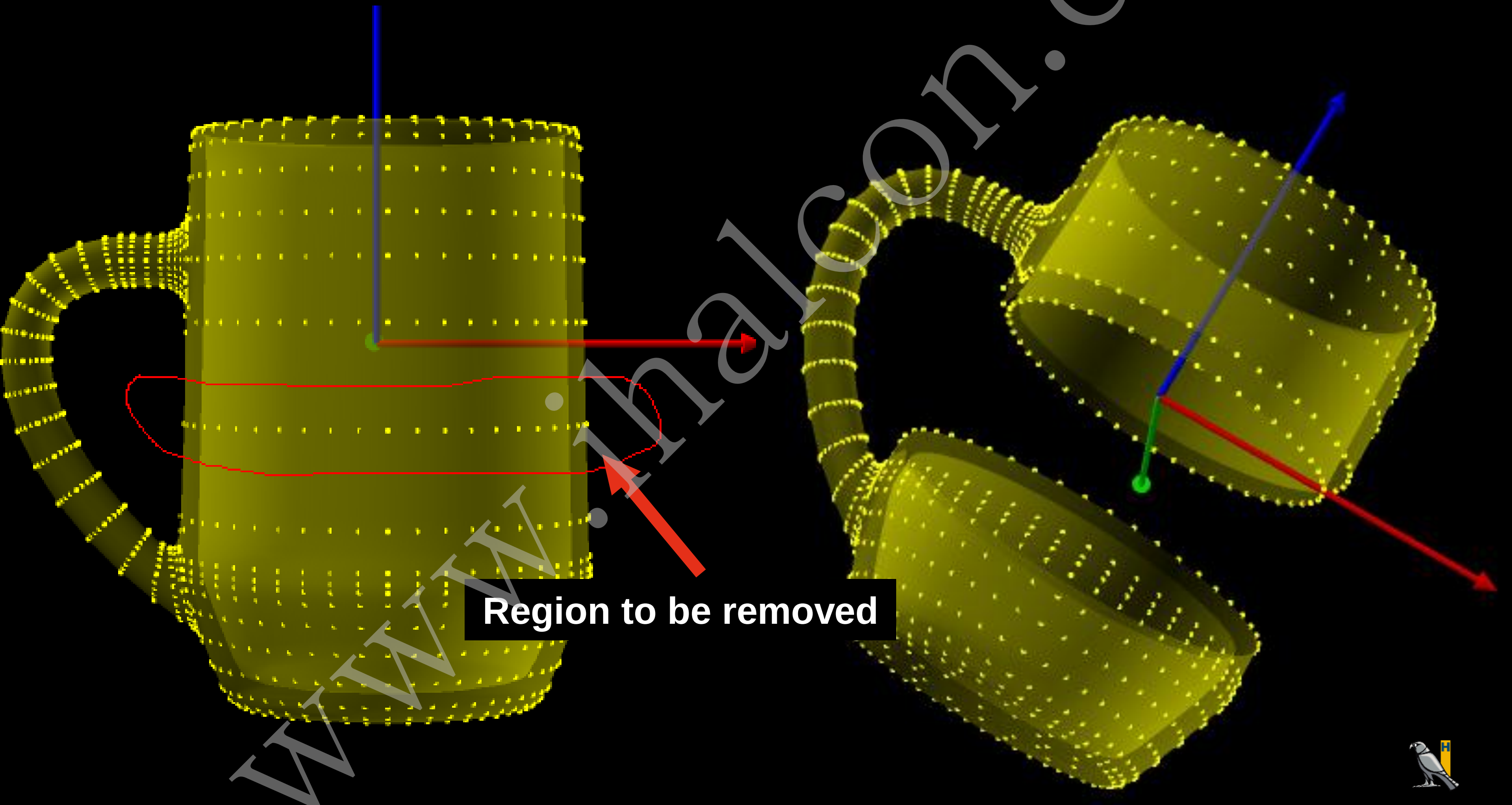
Calculate slices with planes of 3D objects





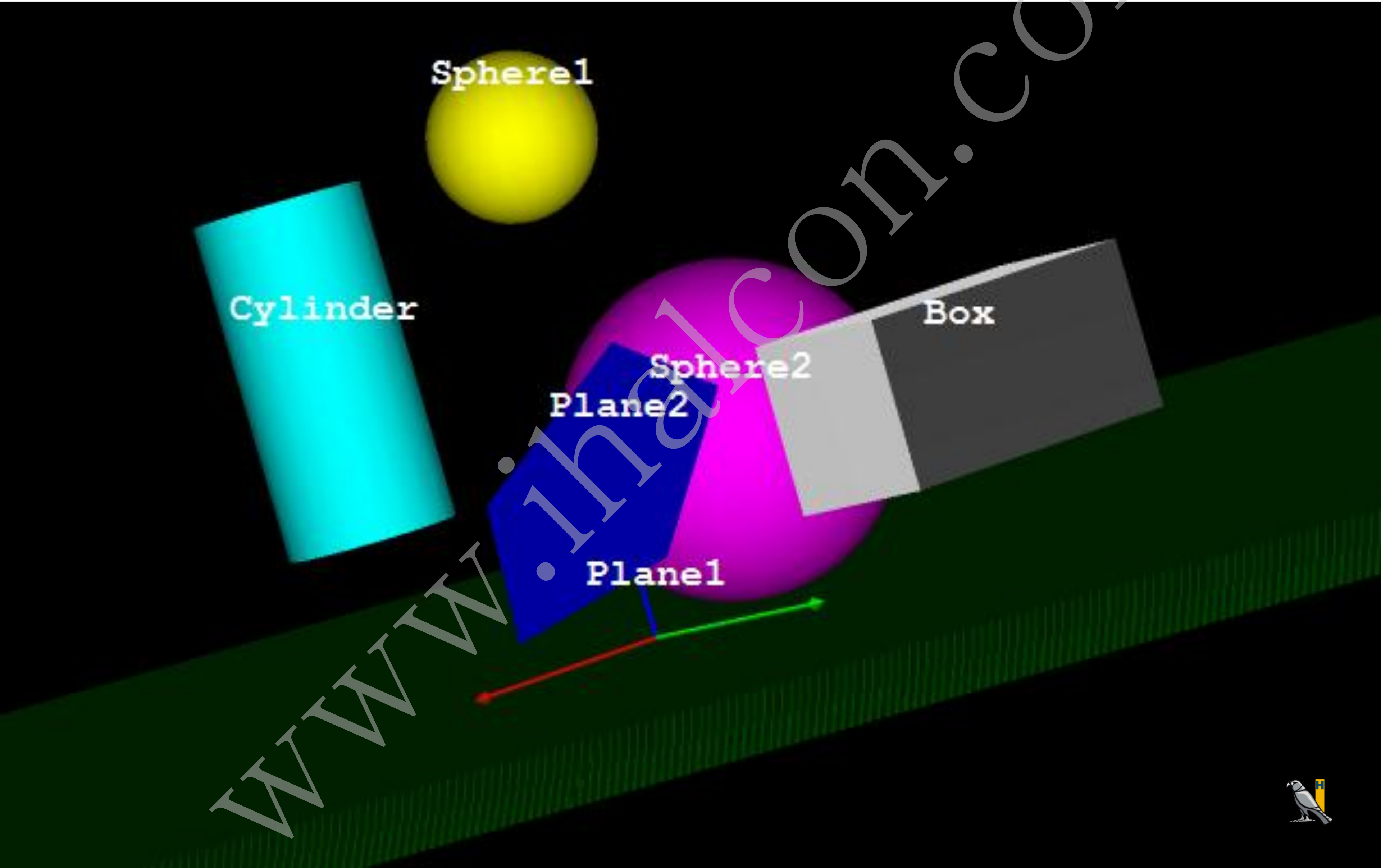
Select parts of 3D object models via regions

```
reduce_object_model_3d_by_view()
```

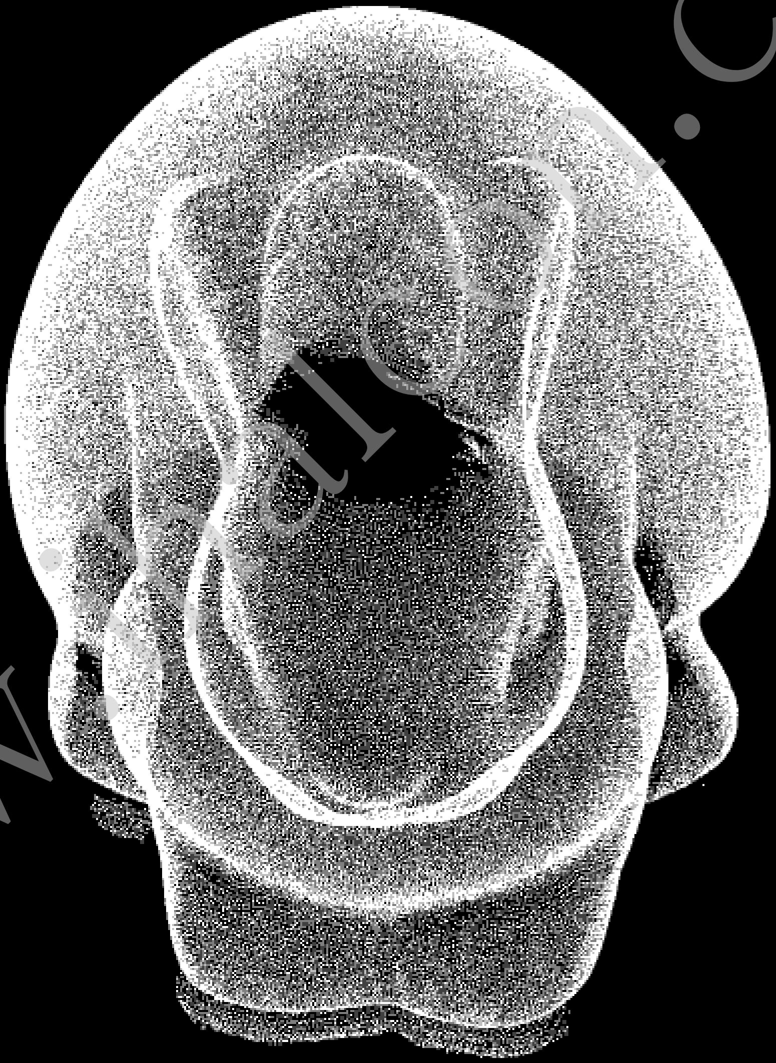


Region to be removed

Creatie 3D primitives



Triangulate 3D point clouds



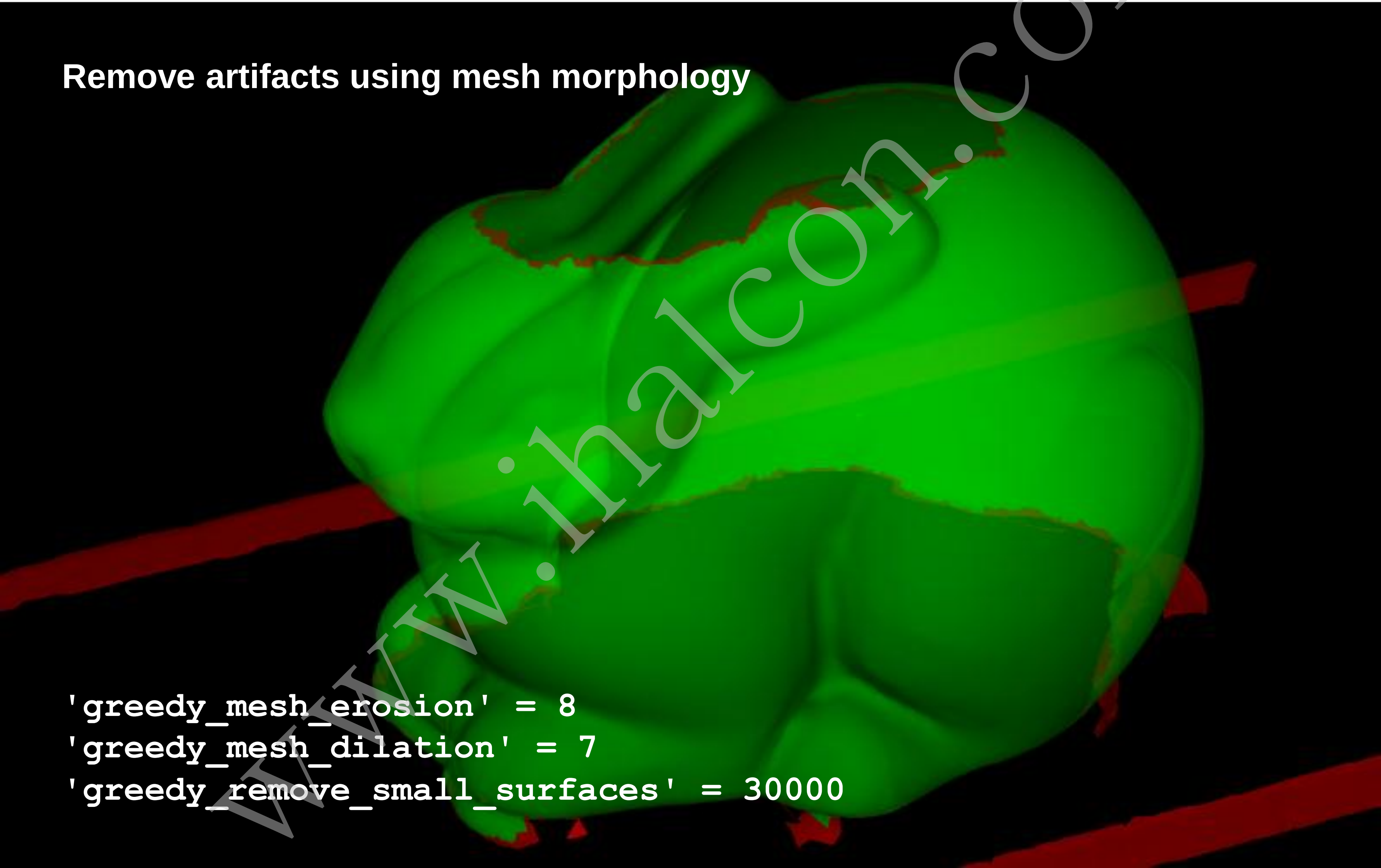
Raw data

Triangulate 3D point clouds

Greedy triangulation with default settings

Triangulate 3D point clouds

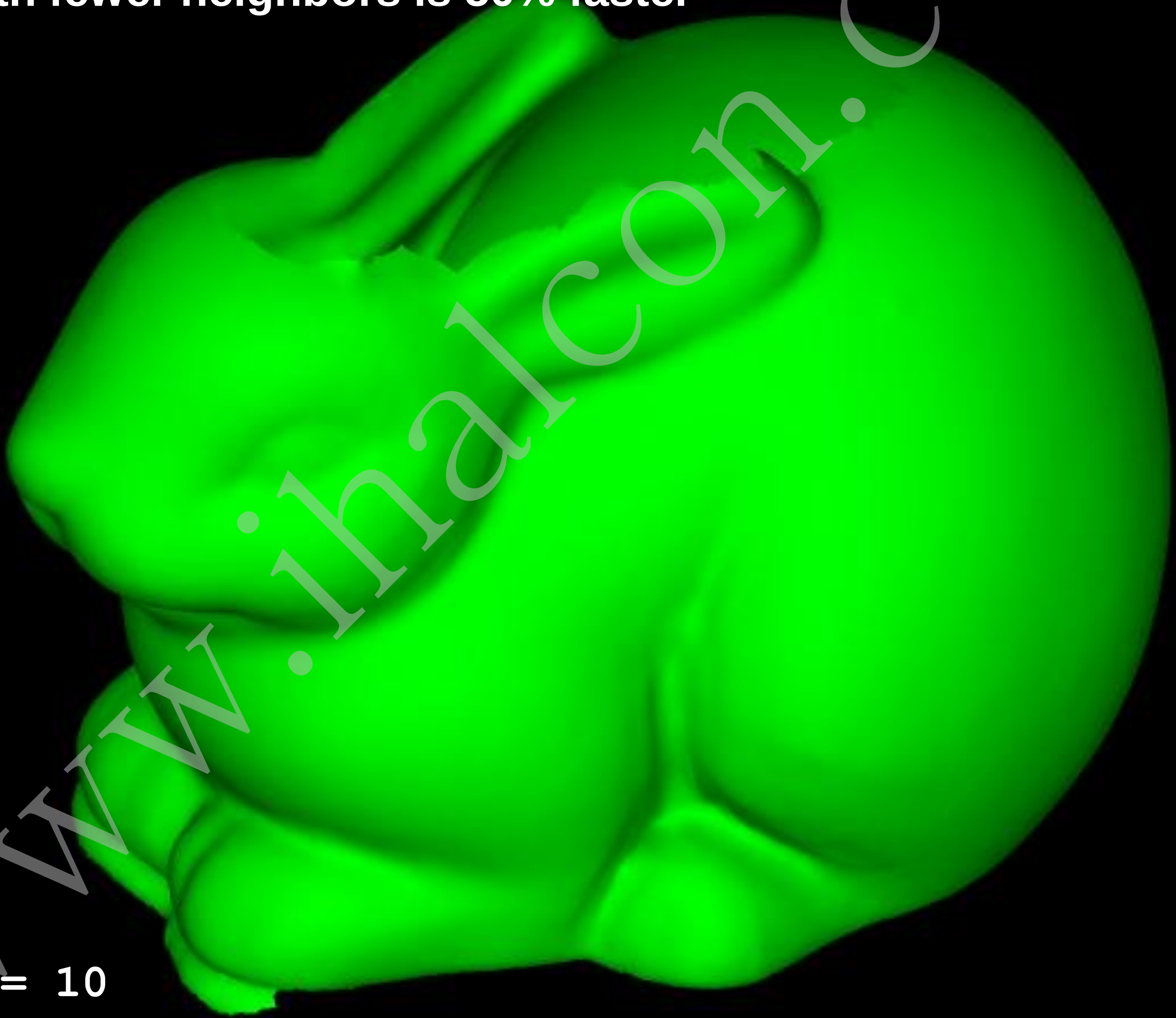
Remove artifacts using mesh morphology



```
'greedy_mesh_erosion' = 8
'greedy_mesh_dilation' = 7
'greedy_remove_small_surfaces' = 30000
```

Triangulate 3D point clouds

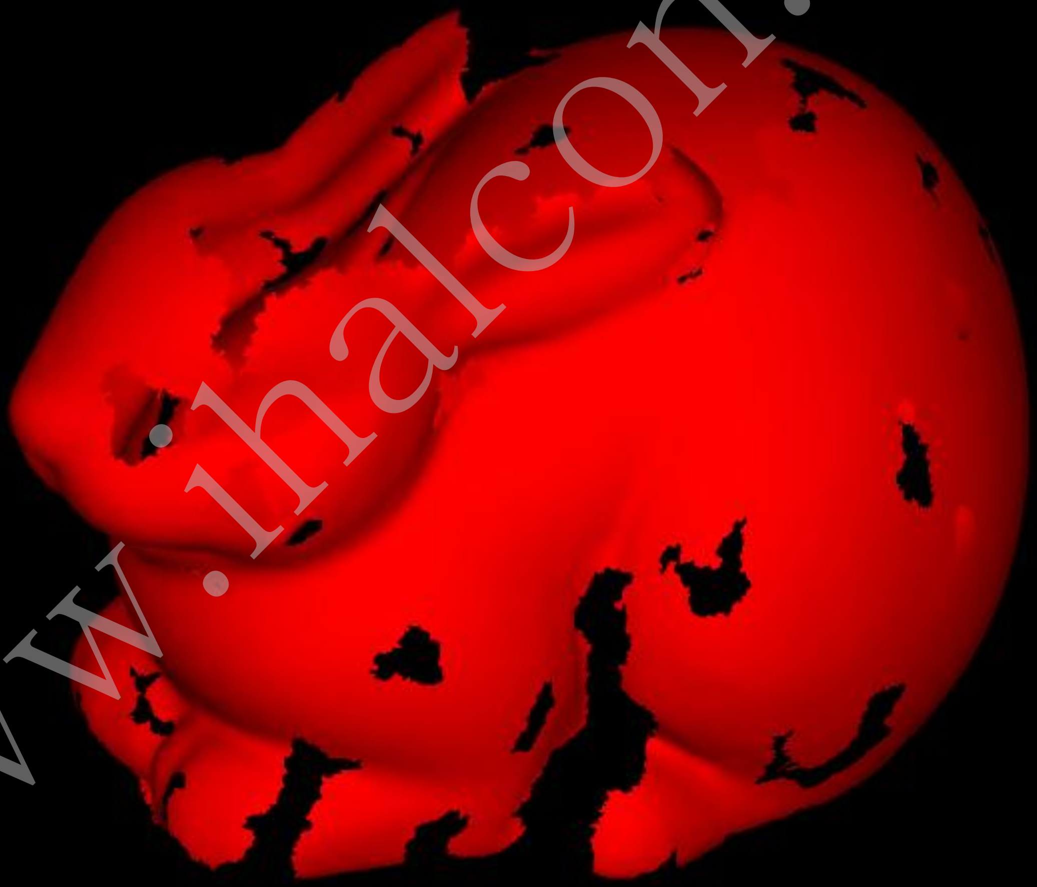
Triangulation with fewer neighbors is 50% faster



`'greedy_kNN' = 10`

Triangulate 3D point clouds

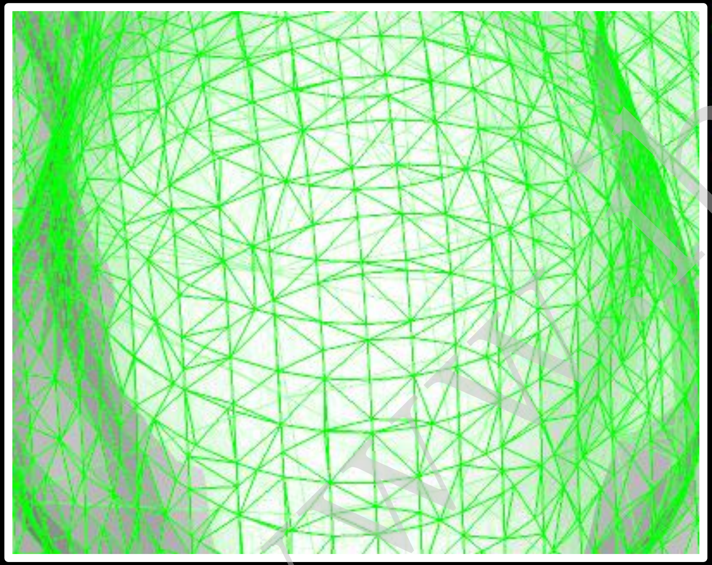
Triangulation with too few neighbors

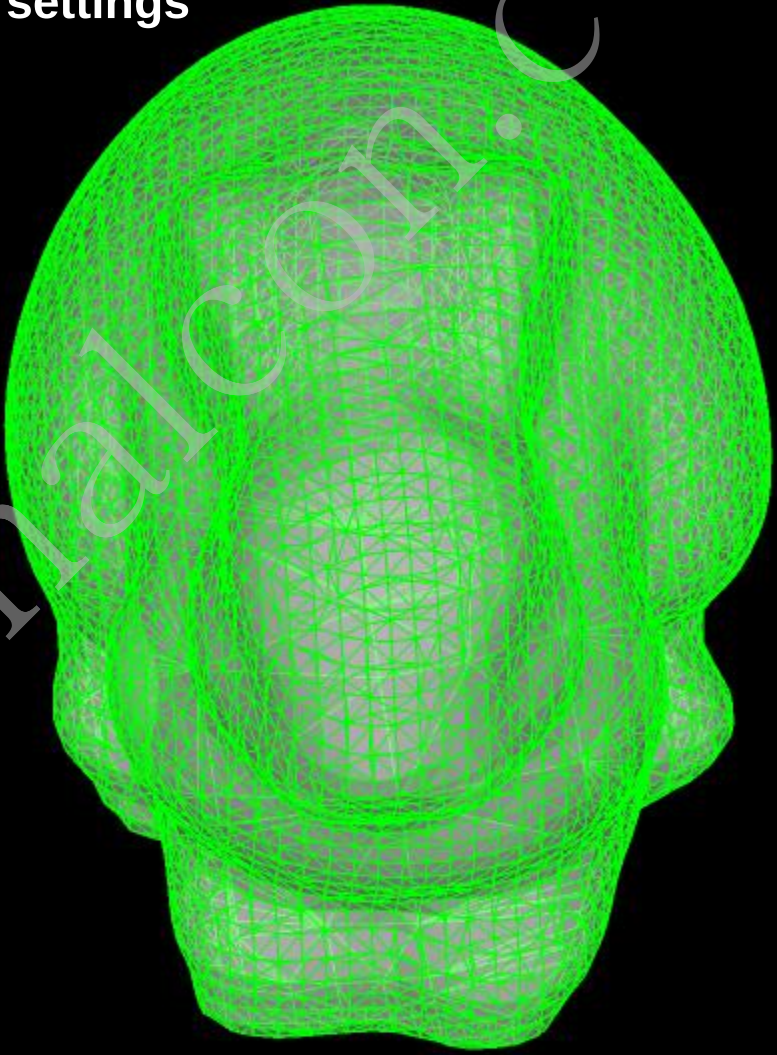


'greedy_kNN' = 5

Implicit triangulation leads to water-tight surfaces

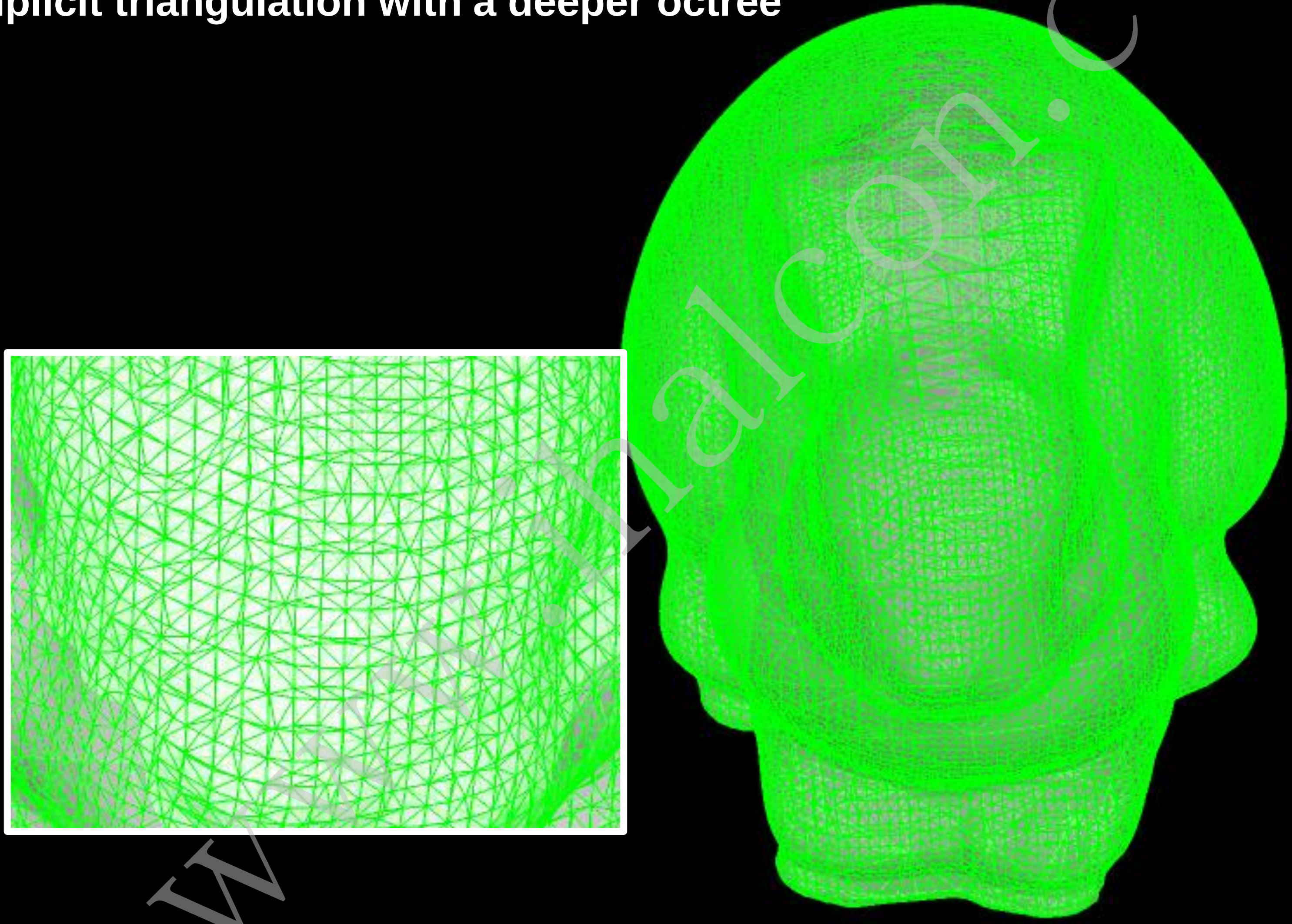
Implicit triangulation with default settings



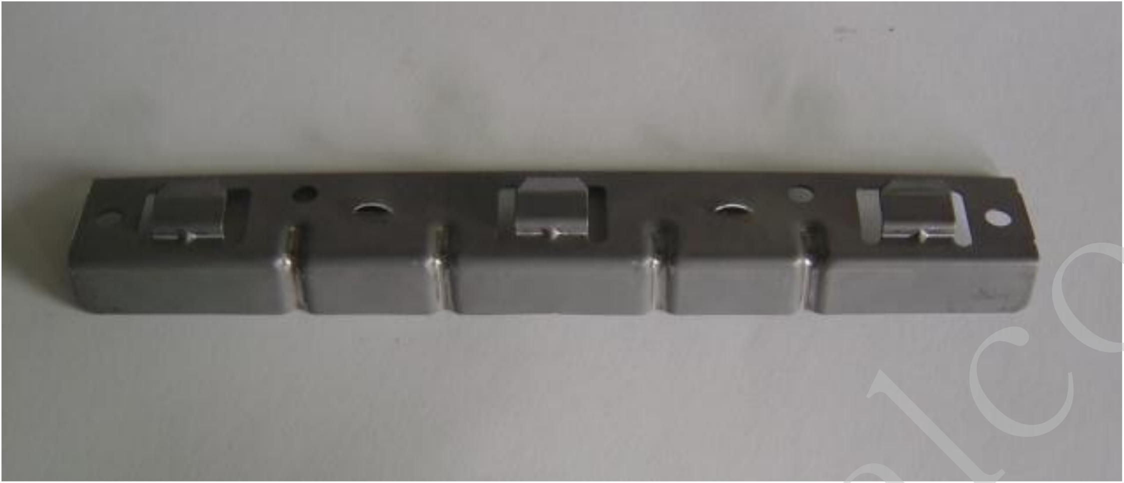


Implicit triangulation leads to water-tight surfaces

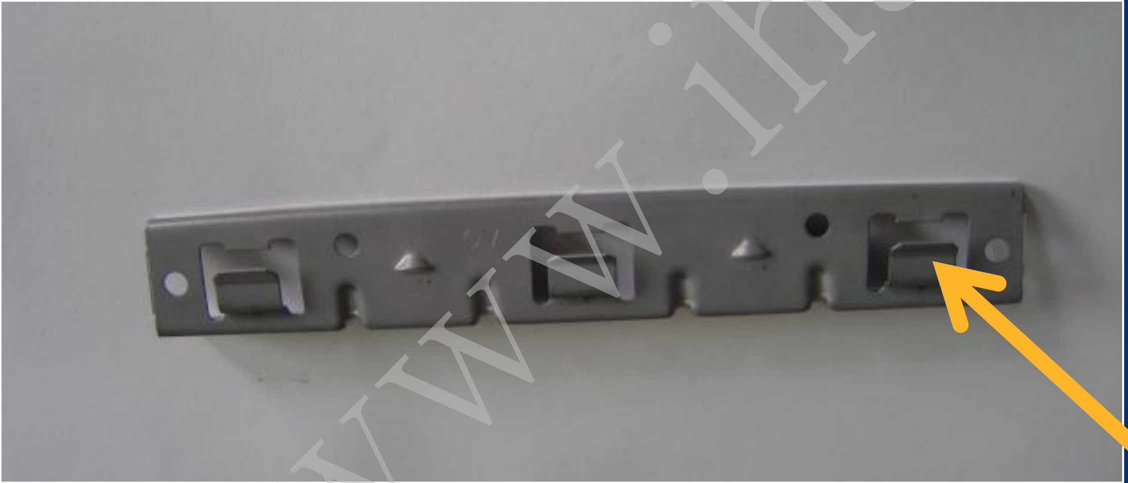
Implicit triangulation with a deeper octree



3D surface comparison: Example



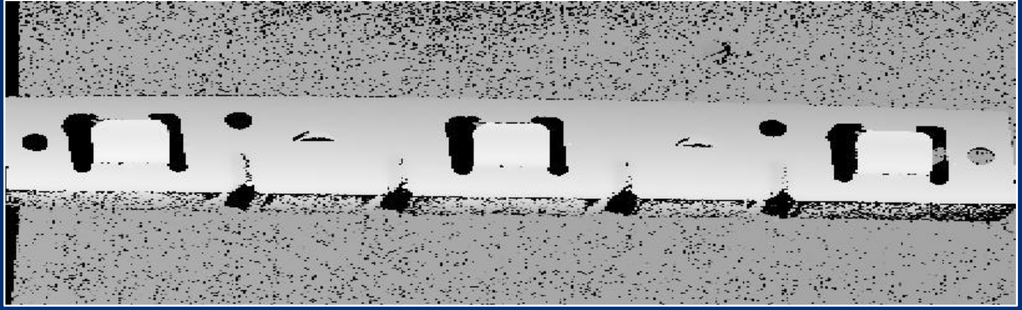
Reference object



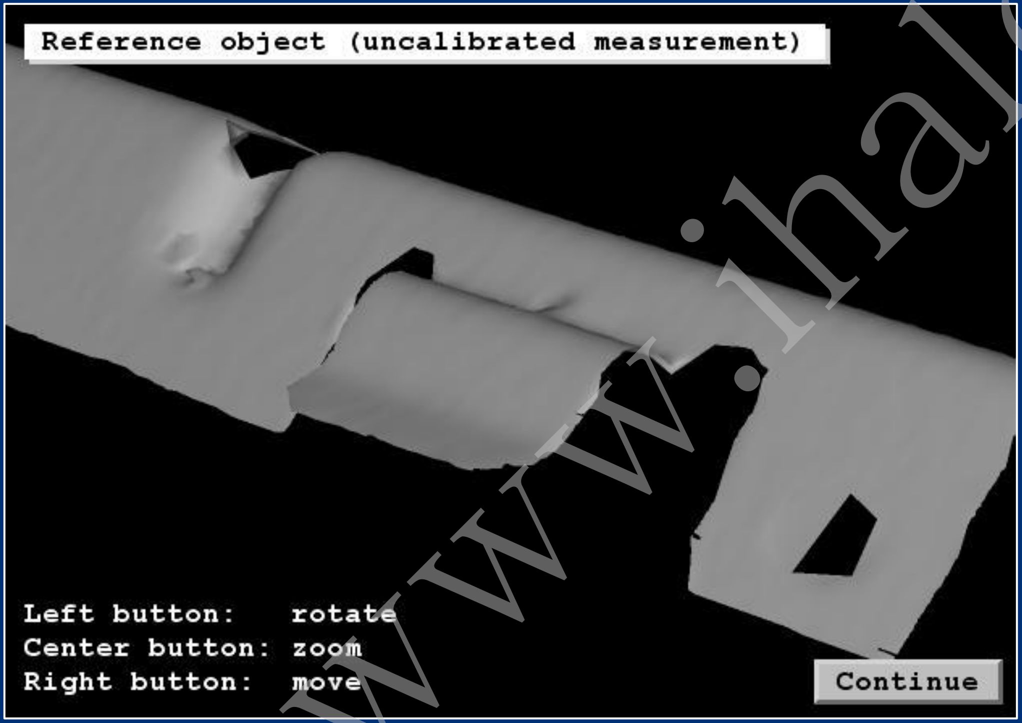
Test objects

Typical defect: bent parts

3D surface comparison: Example



Disparities from the sheet-of-light reconstruction

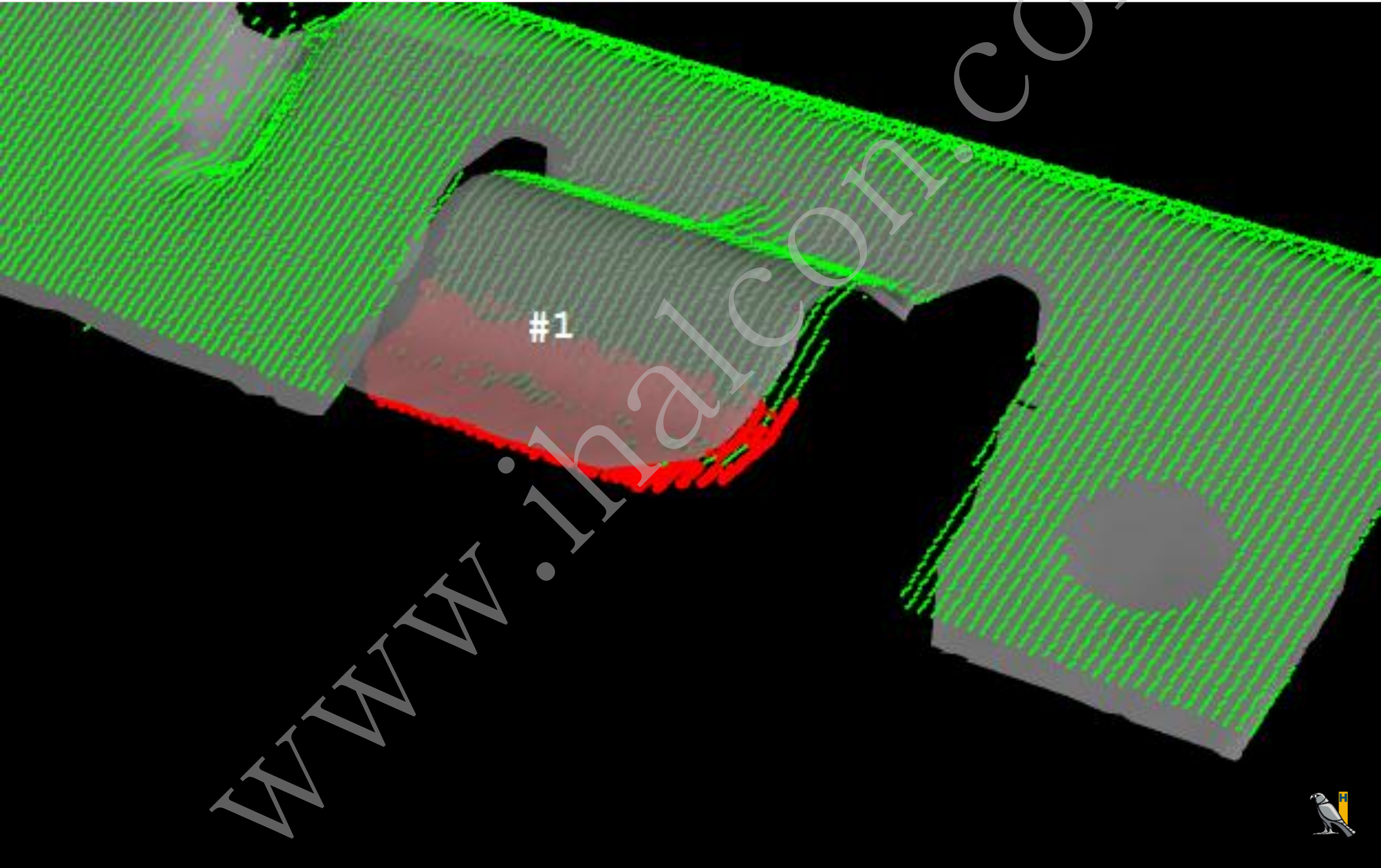


Reference object (uncalibrated measurement)

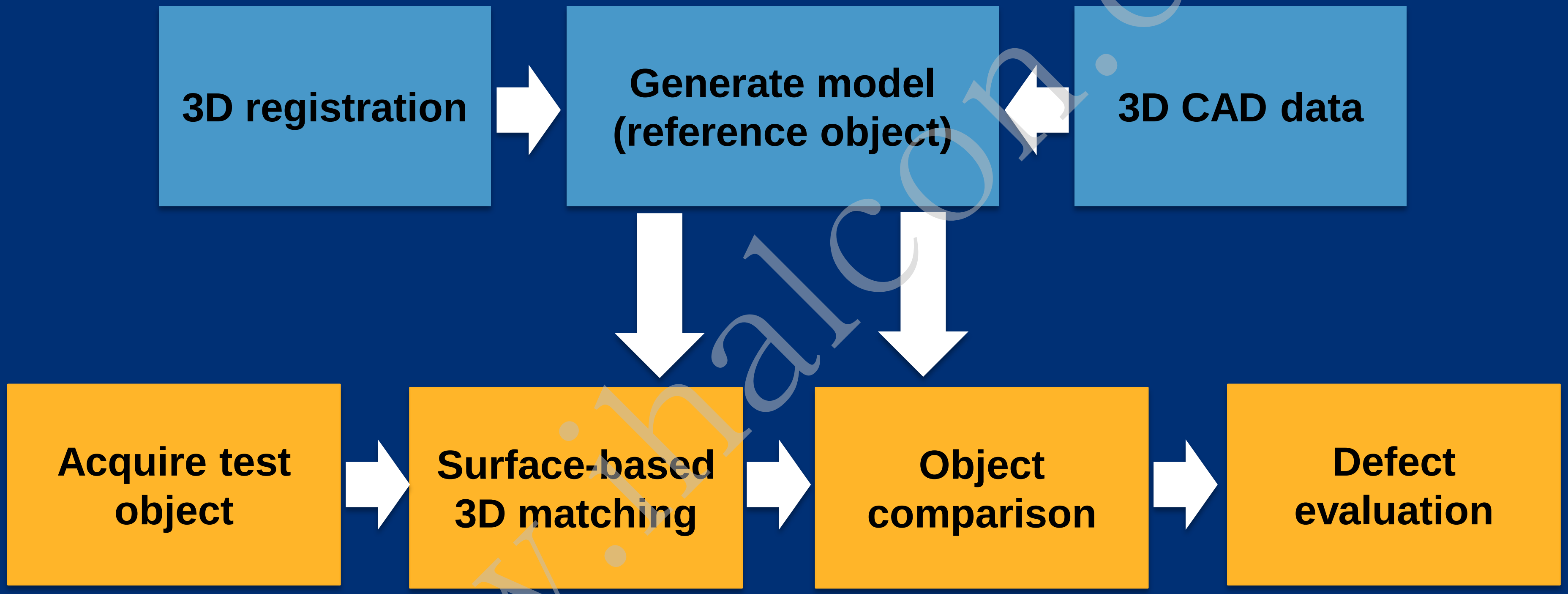
Left button: rotate
Center button: zoom
Right button: move

Reconstructed 3D object model

3D surface comparison: Example



Typical 3D surface comparison process



3D registration

Generate model (reference object)

3D CAD data

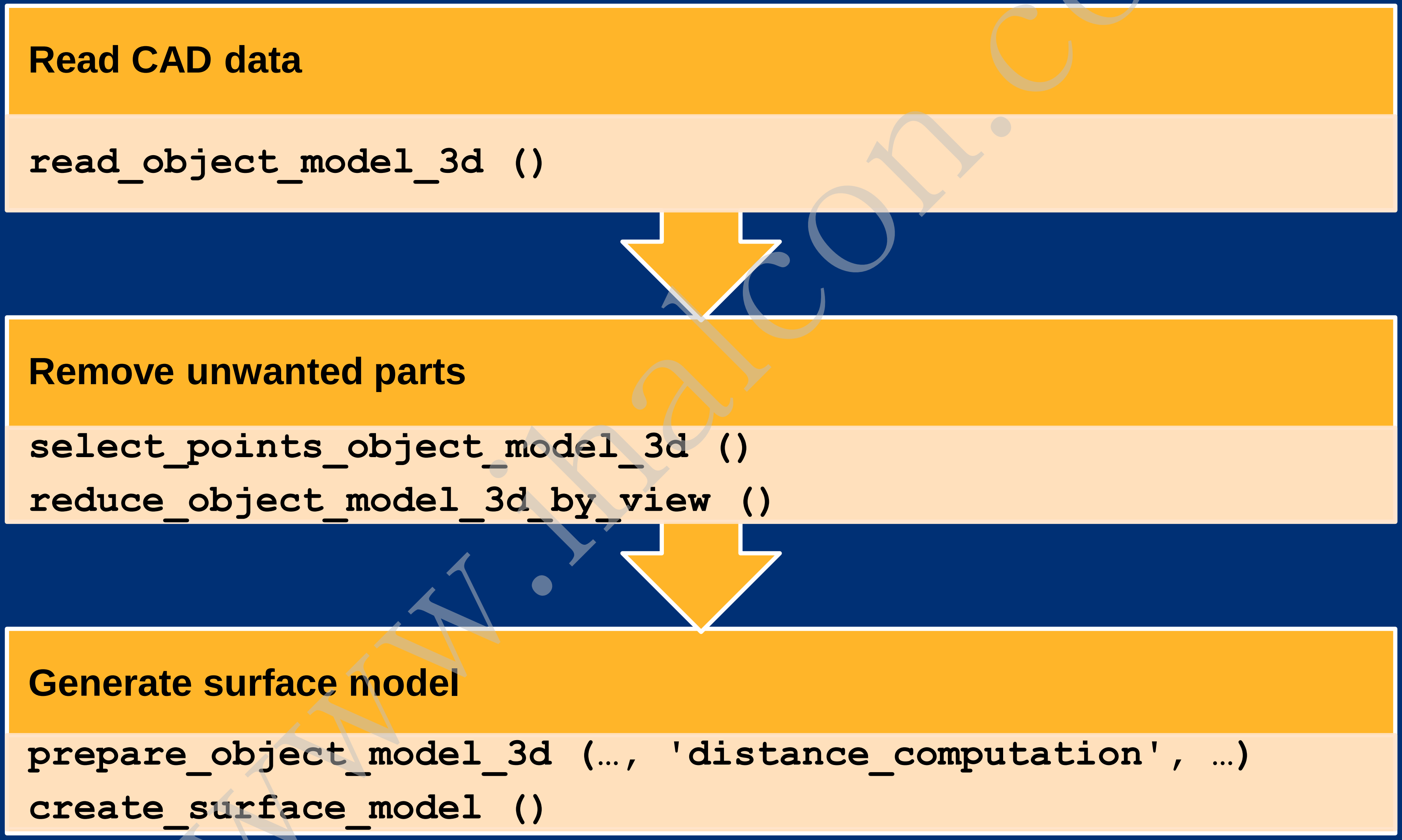
Acquire test object

Surface-based 3D matching

Object comparison

Defect evaluation

Create reference object
for 3D surface comparison from CAD data



Read CAD data

```
read_object_model_3d ()
```

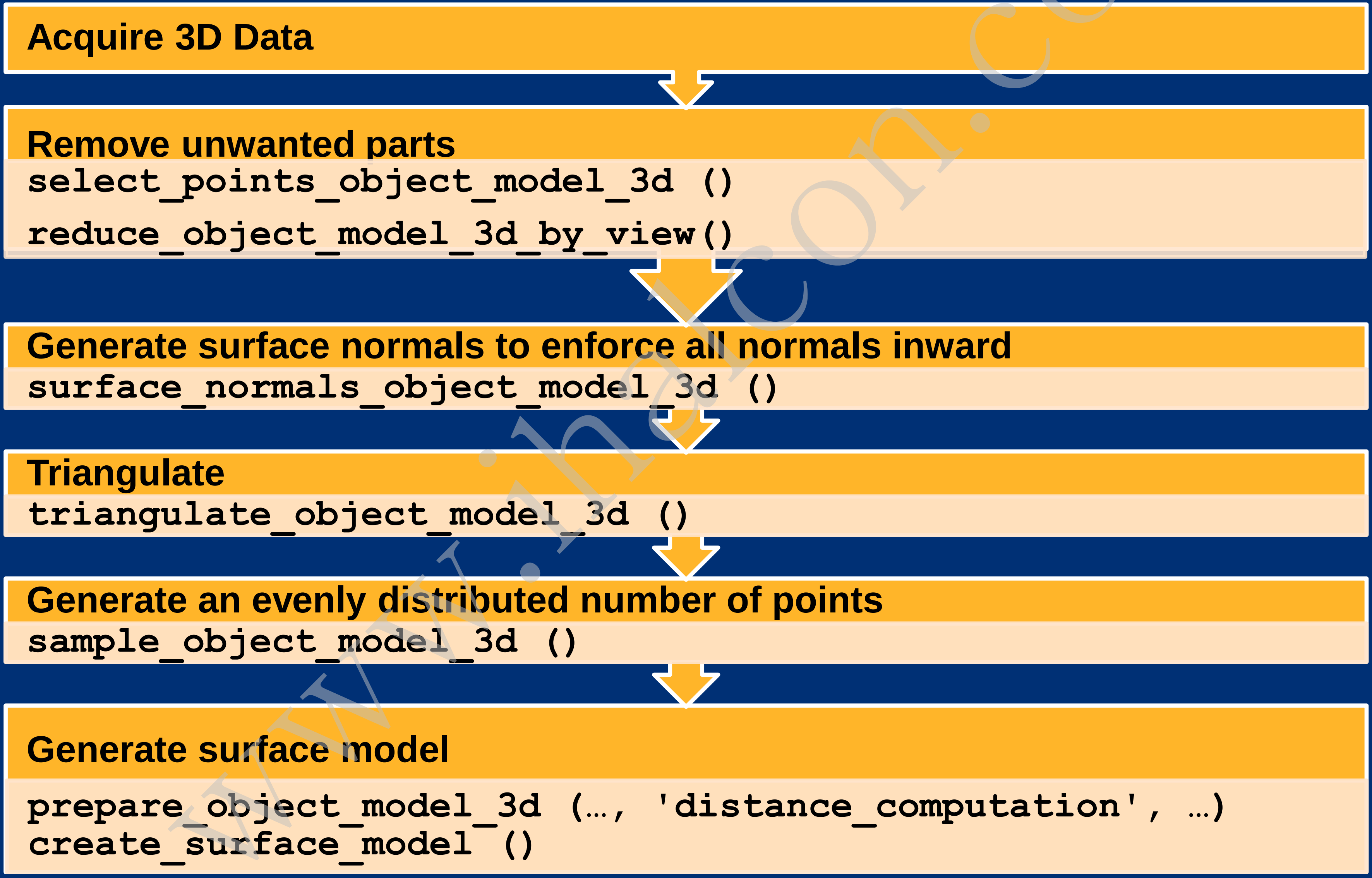
Remove unwanted parts

```
select_points_object_model_3d ()
reduce_object_model_3d_by_view ()
```

Generate surface model

```
prepare_object_model_3d (…, 'distance_computation', …)
create_surface_model ()
```

Create reference object
for 3D surface comparison from sensor data



Acquire 3D Data

Remove unwanted parts

```
select_points_object_model_3d ()
reduce_object_model_3d_by_view()
```

Generate surface normals to enforce all normals inward

```
surface_normals_object_model_3d ()
```

Triangulate

```
triangulate_object_model_3d ()
```

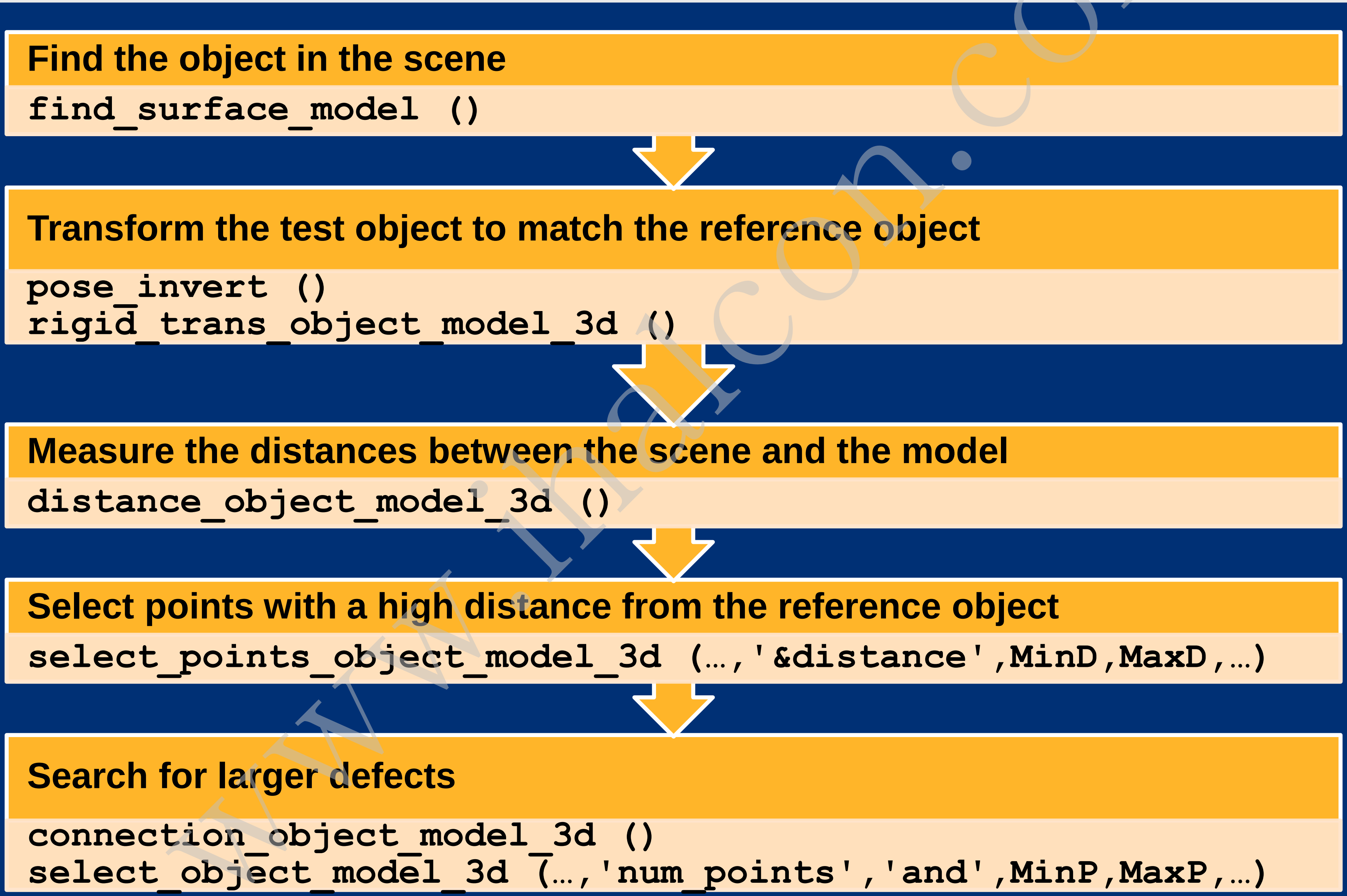
Generate an evenly distributed number of points

```
sample_object_model_3d ()
```

Generate surface model

```
prepare_object_model_3d (…, 'distance_computation', …)
create_surface_model ()
```

Perform 3D surface comparison



Find the object in the scene

```
find_surface_model ()
```

Transform the test object to match the reference object

```
pose_invert ()
rigid_trans_object_model_3d ()
```

Measure the distances between the scene and the model

```
distance_object_model_3d ()
```

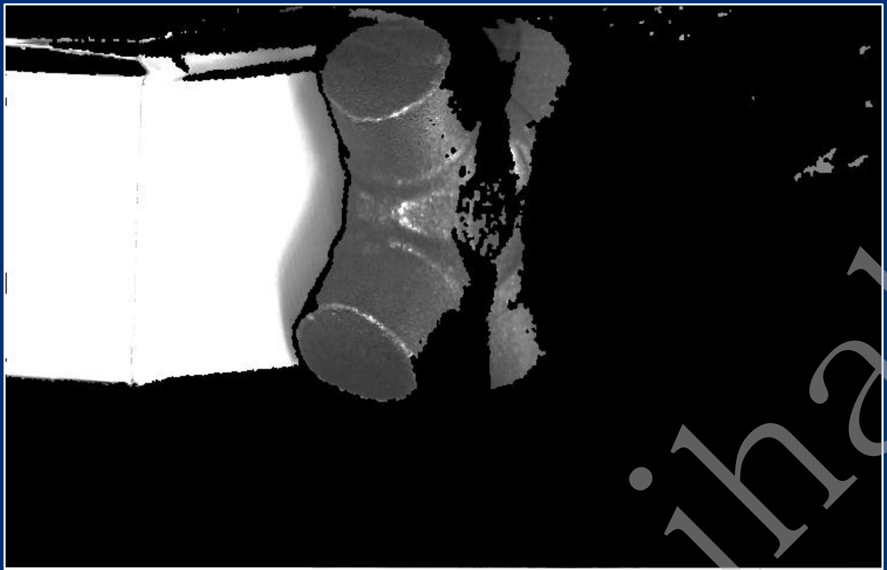
Select points with a high distance from the reference object

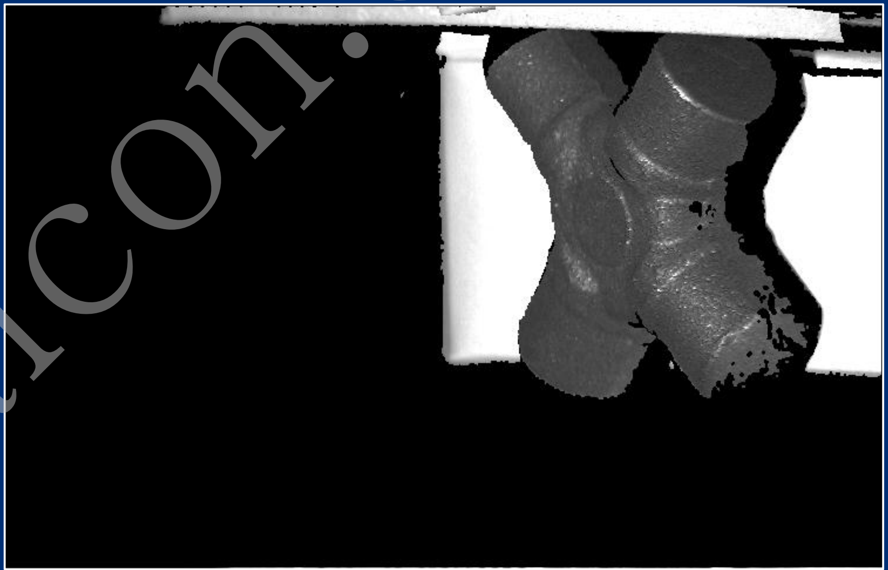
```
select_points_object_model_3d (…,'&distance',MinD,MaxD,…)
```

Search for larger defects

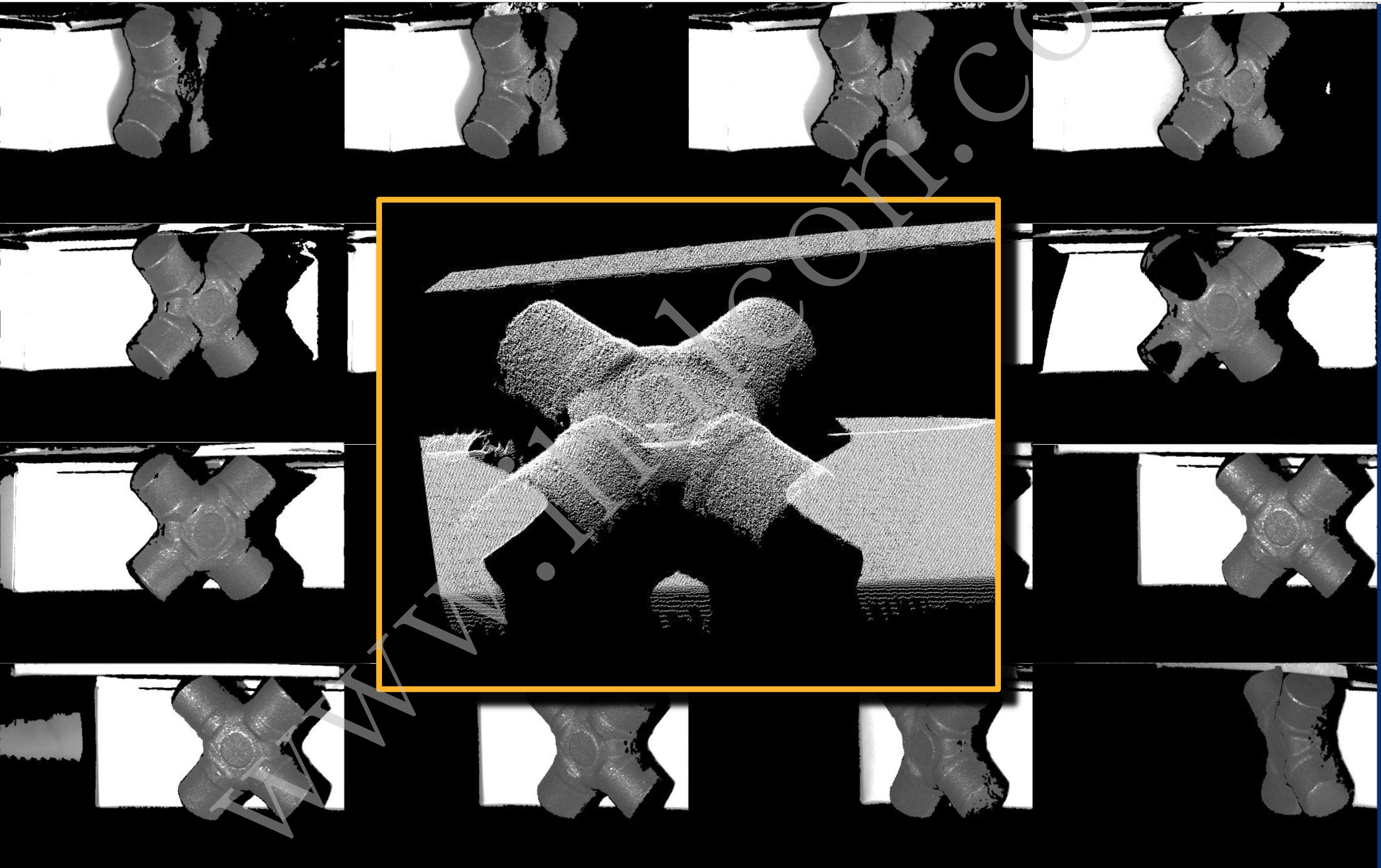
```
connection_object_model_3d ()
select_object_model_3d (…,'num_points','and',MinP,MaxP,…)
```

3D registration is a useful step to generate reference models





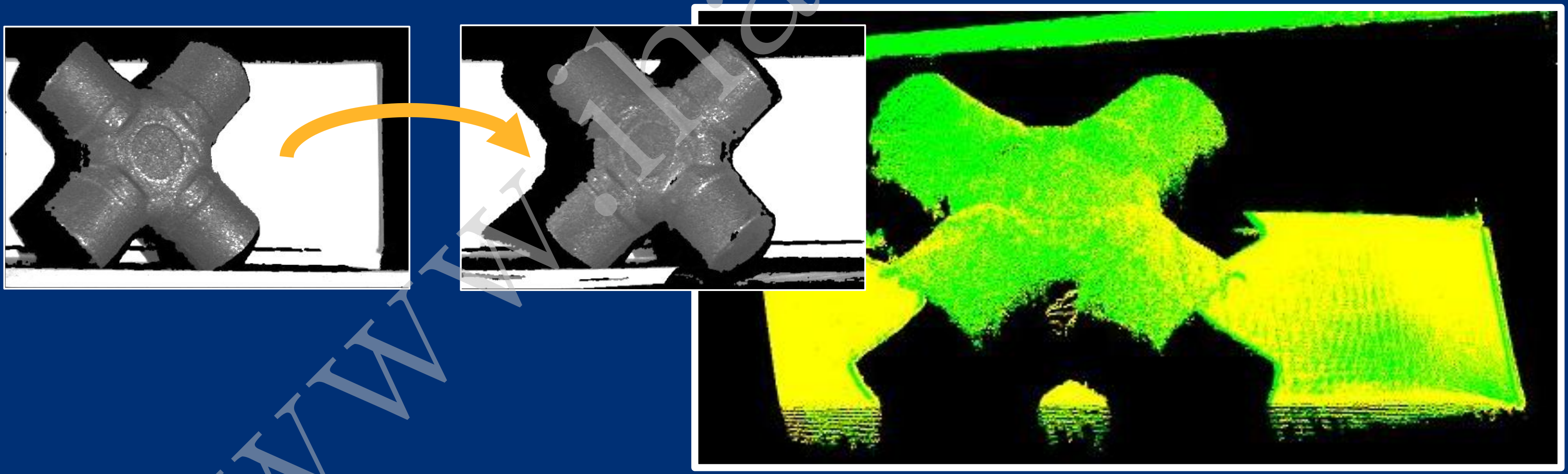
3D registration:
Combine multiple views to a single object



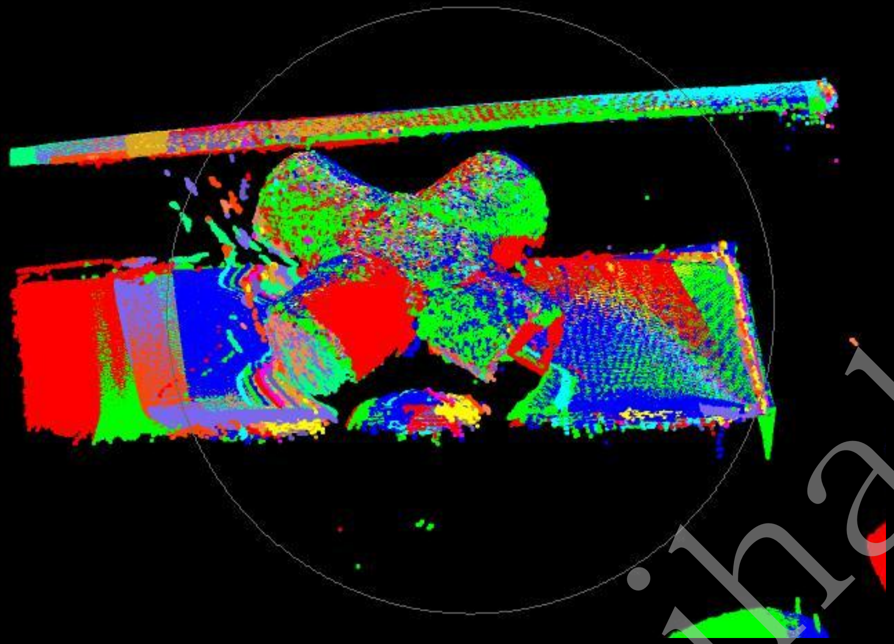
Pairwise 3D registration

- register_object_model_3d_pair (: :
  - ▶ ObjectModel3D1,
  - ▶ ObjectModel3D2,
  - ▶ Method,
  - ▶ GenParamName,
- ▶ GenParamValue :
  - ◀ Pose,
  - ◀ Score)

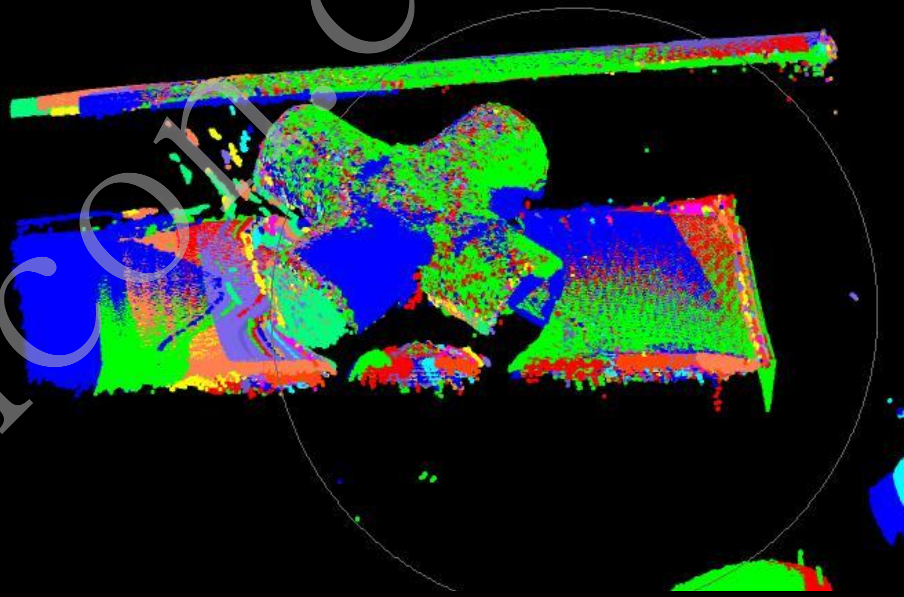
'matching'
'icp'



Globally optimized registration is more accurate



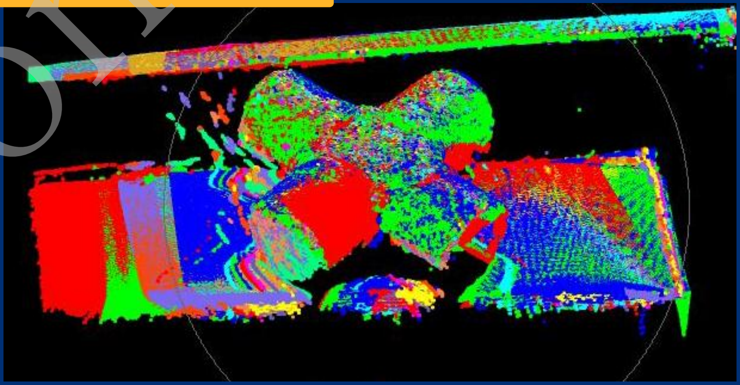
Overlaid pairwise registration



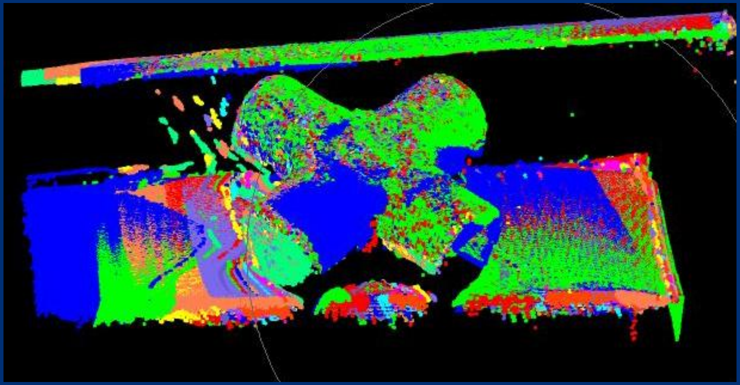
Globally optimized registration

Globally optimized registration is more accurate

- `register_object_model_3d_global (: :`
  - ▶ `ObjectModels3D,`
  - ▶ `HomMats3D,` ← From pairwise registration or robot
  - ▶ `From, To,`
  - ▶ `GenParamName, GenParamValue :`
  - ◀ `HomMats3DOut,`
  - ◀ `Scores)`



Overlaid pairwise registration



Globally optimized registration

Unify 3D object models

```
union_object_model_3d
```

Sub-sample 3D object models

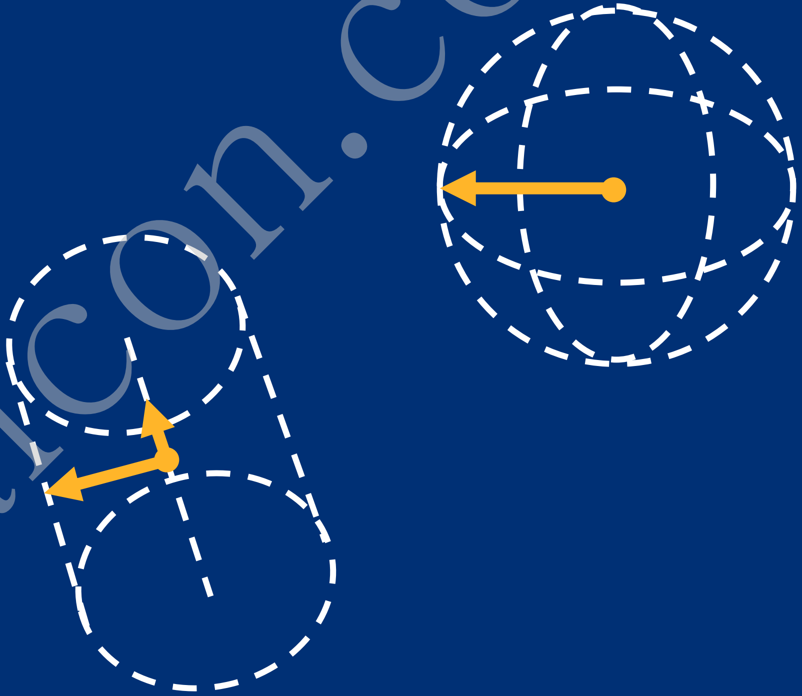
`sample_object_model_3d`

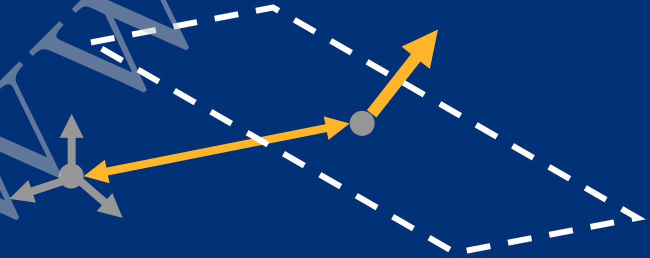
Smooth and triangulate 3D object models

```
smooth_object_model_3d
triangulate_object_model_3d
```

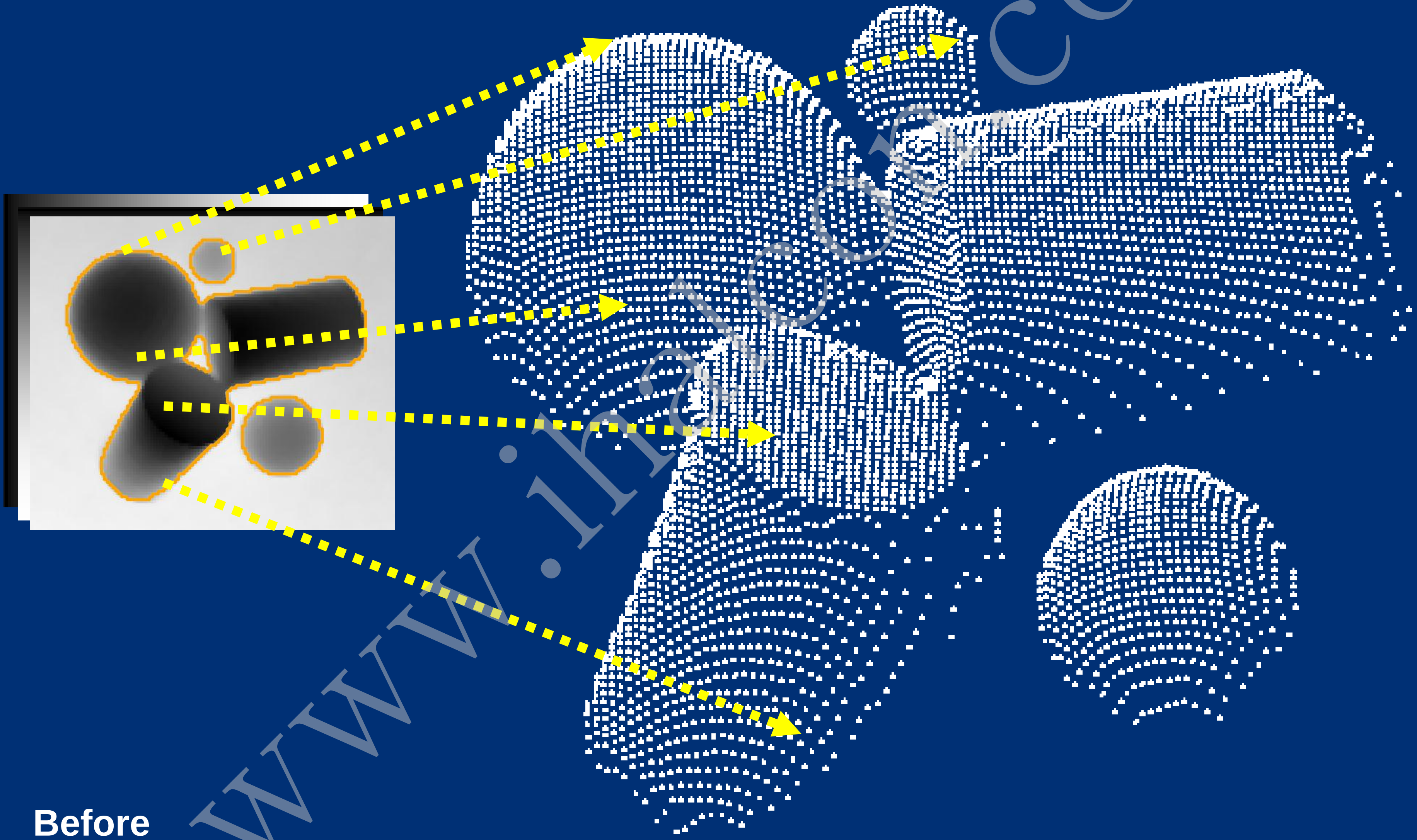
Primitive parameters

- Sphere
  - Center
  - Radius
- Cylinder
  - Center
  - Direction
  - Radius
- Plane
  - aX + bY + cZ + d = 0



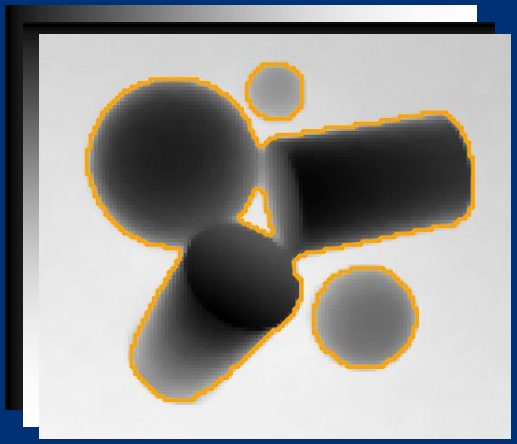


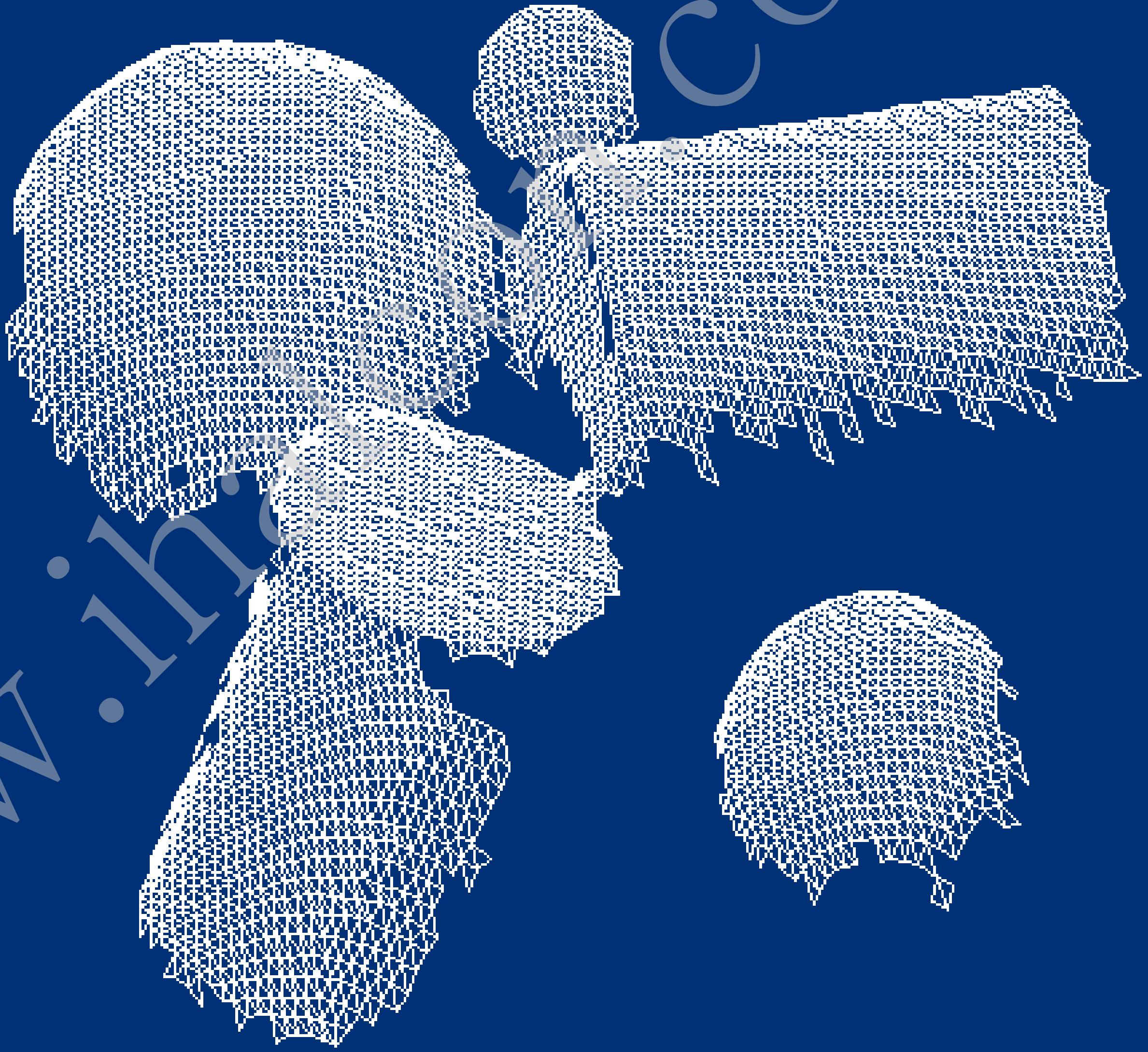
The XYZ-mapping is needed for triangulation



Before

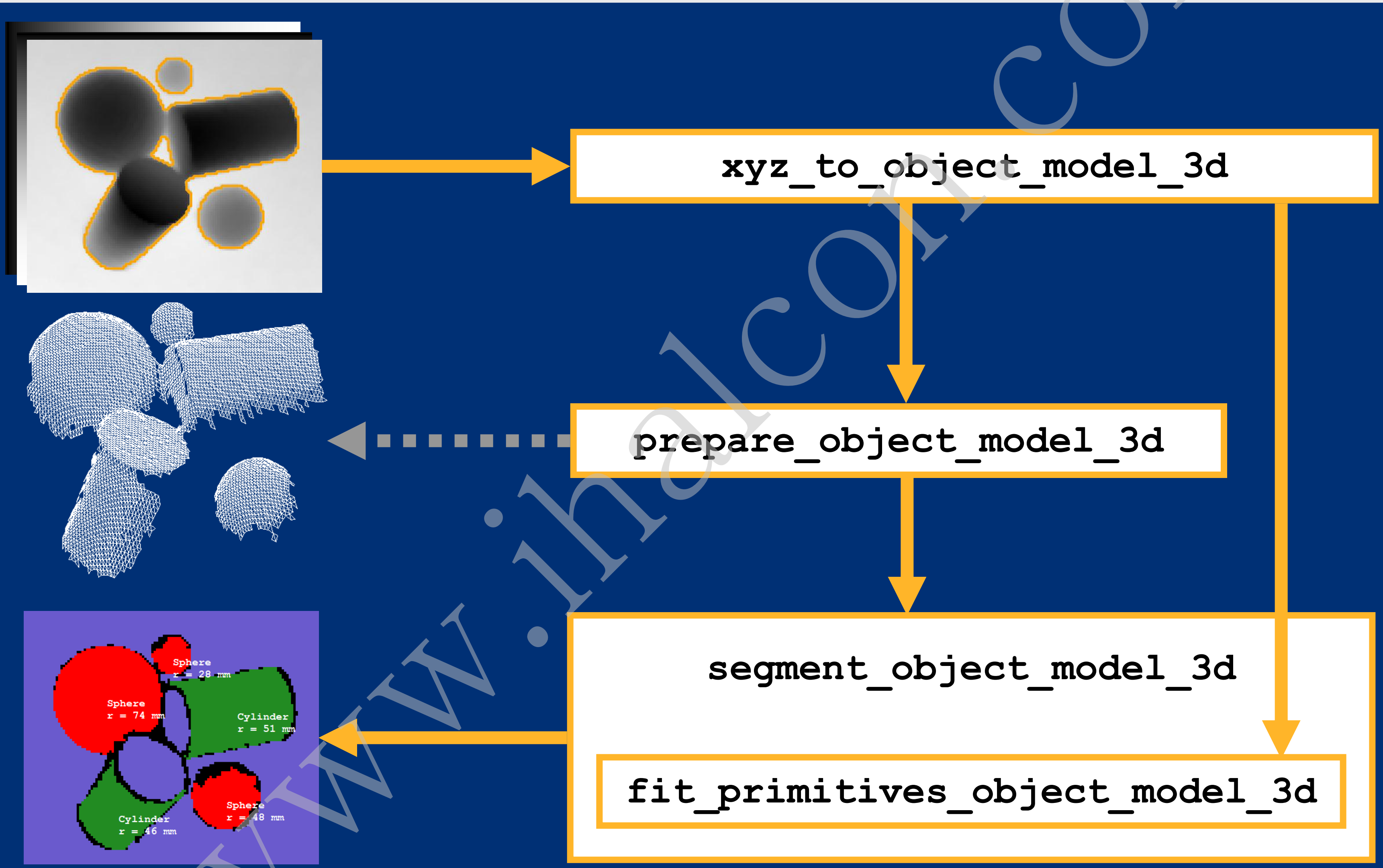
The XYZ-mapping is needed for triangulation





After

Segmentation and fitting can be done in one step



xyz_to_object_model_3d

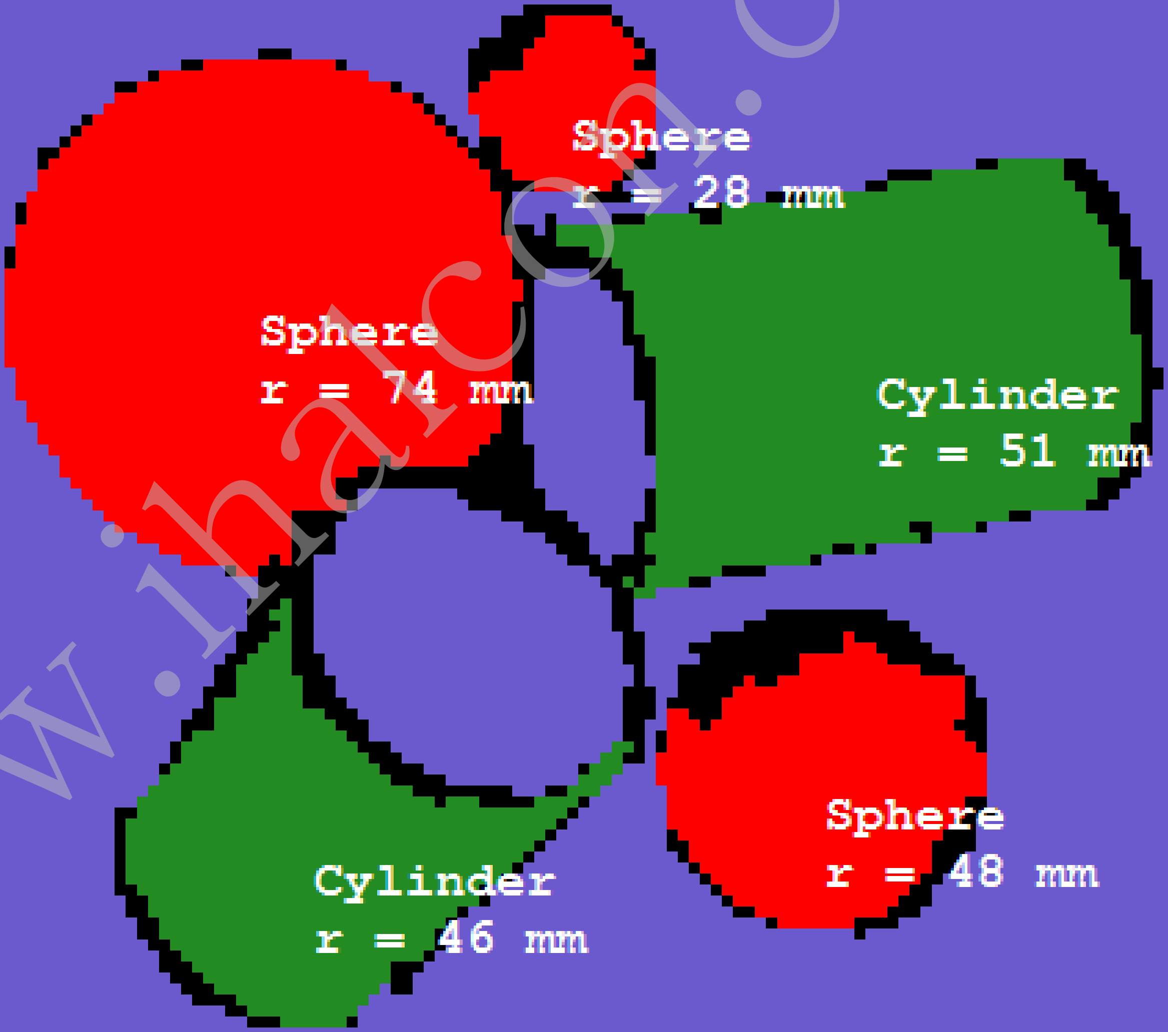
prepare_object_model_3d

segment_object_model_3d

fit_primitives_object_model_3d

Segment and fit primitives in 3D data

1. Segmentation
2. Fitting

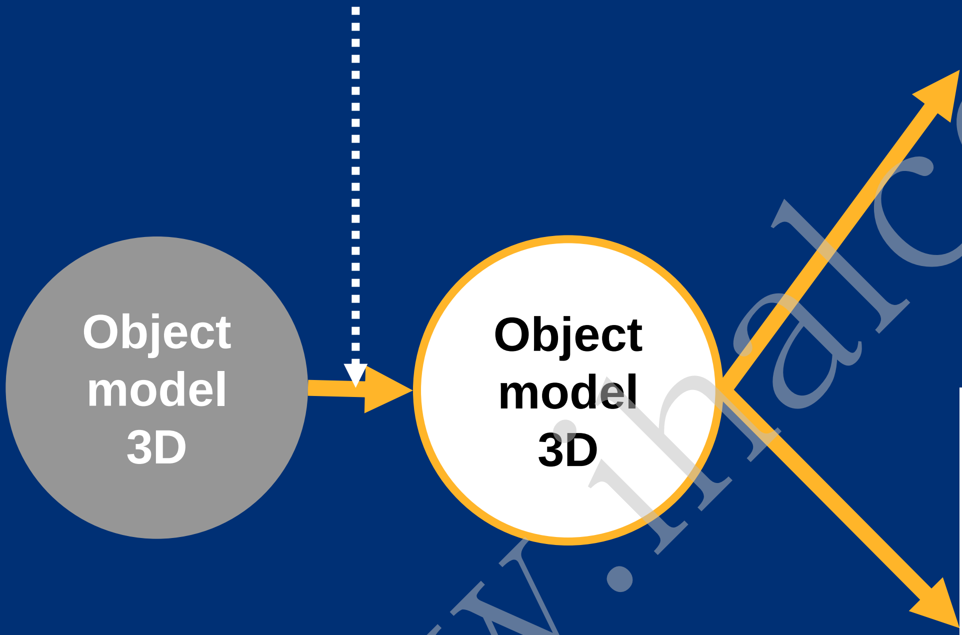


Object model 3D use cases

Object model 3D input	Method	Object model 3D output
	reconstruct_surface_stereo ('point_meshing')	Points + normals (Points + triangles)
Points + triangles (Points + XYZ-mapping)	segment_object_model_3d ('output_xyz_mapping')	Points (+ XYZ-mapping)
Points	fit_primitives_object_model_3d ('output_point_coord') ('output_xyz_mapping')	Primitive parameters (+ Points) (+ Points + XYZ-mapping)
Points + normals (Points + XYZ-mapping) (Points + triangles) (Points + polygons)	create/find_surface_model	
Points + polygons (Points + triangles)	create_shape_model_3d	
	get_sheet_of_light_result_object_model_3d	Points + XYZ-mapping
	xyz_to_object_model_3d	Points + XYZ-mapping

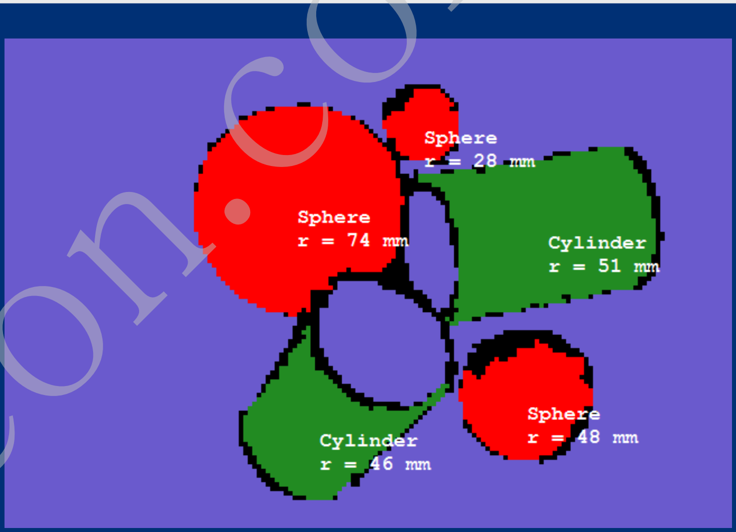
Prepare the 3D object model before use

prepare_object_model_3d

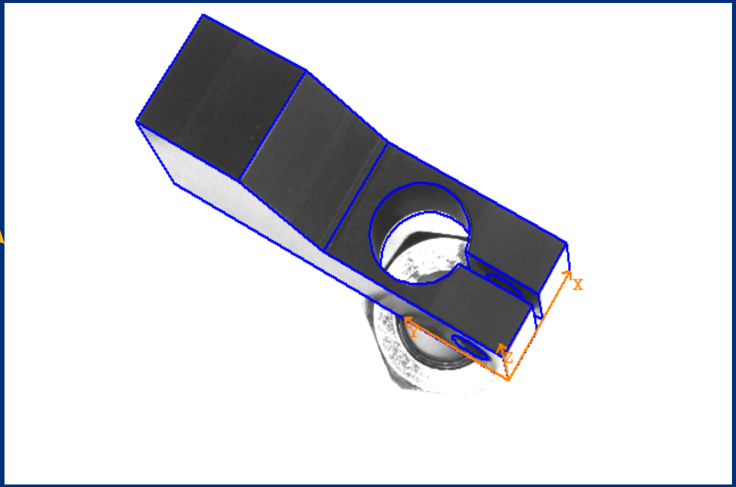


Object model 3D

Object model 3D

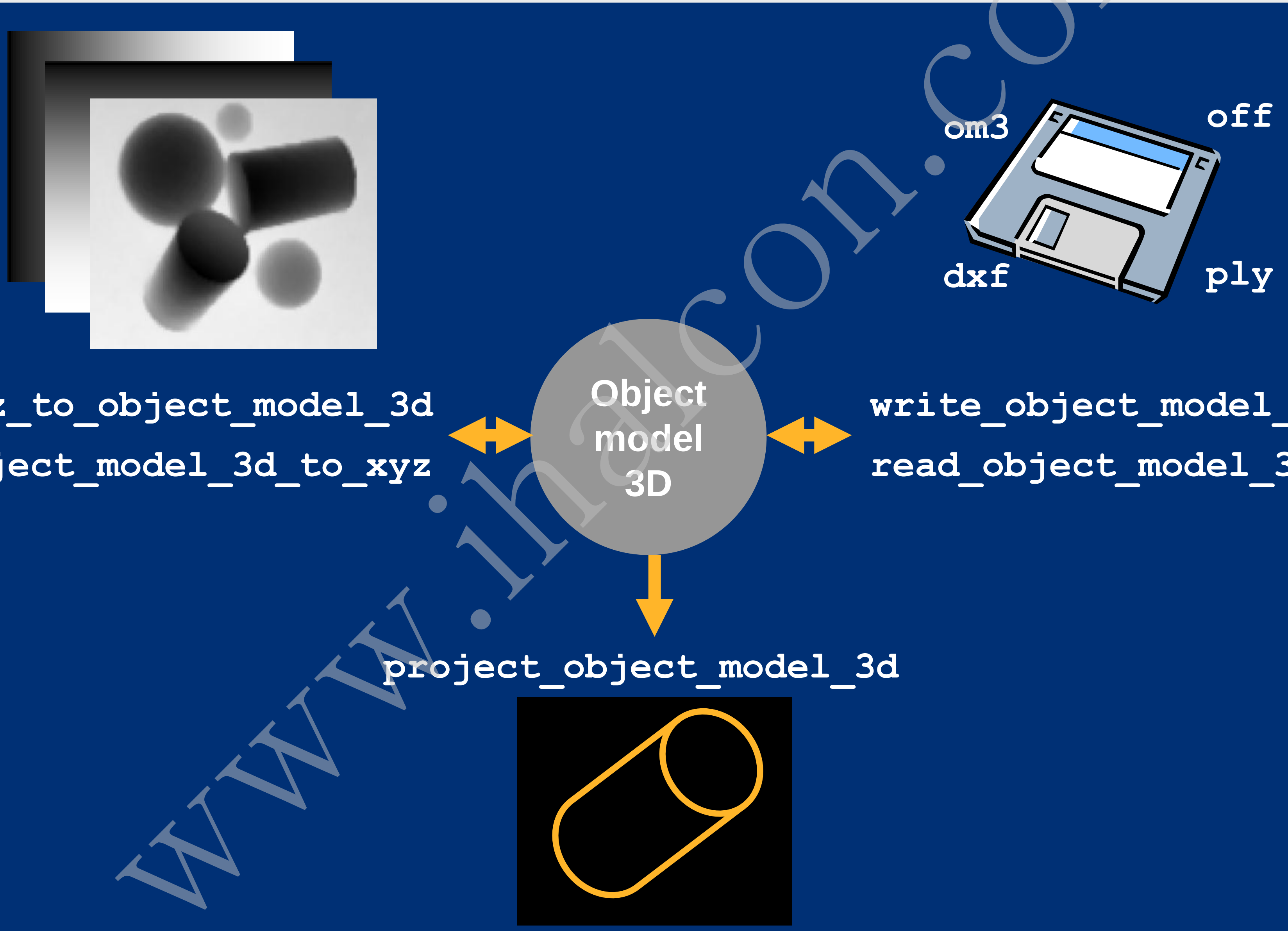


segment_object_model_3d



find_shape_model_3d

Convert 3D object model



xyz_to_object_model_3d

object_model_3d_to_xyz

Object model 3D

write_object_model_3d

read_object_model_3d

project_object_model_3d

HALCON

the Power of Machine Vision

Thank you!

MVTEC

IMAVISION

Contact Information

段德山

Telephone:

010-82828878-8078

Mobile Phone:

13810099305

Email:

duands@daheng-image.com