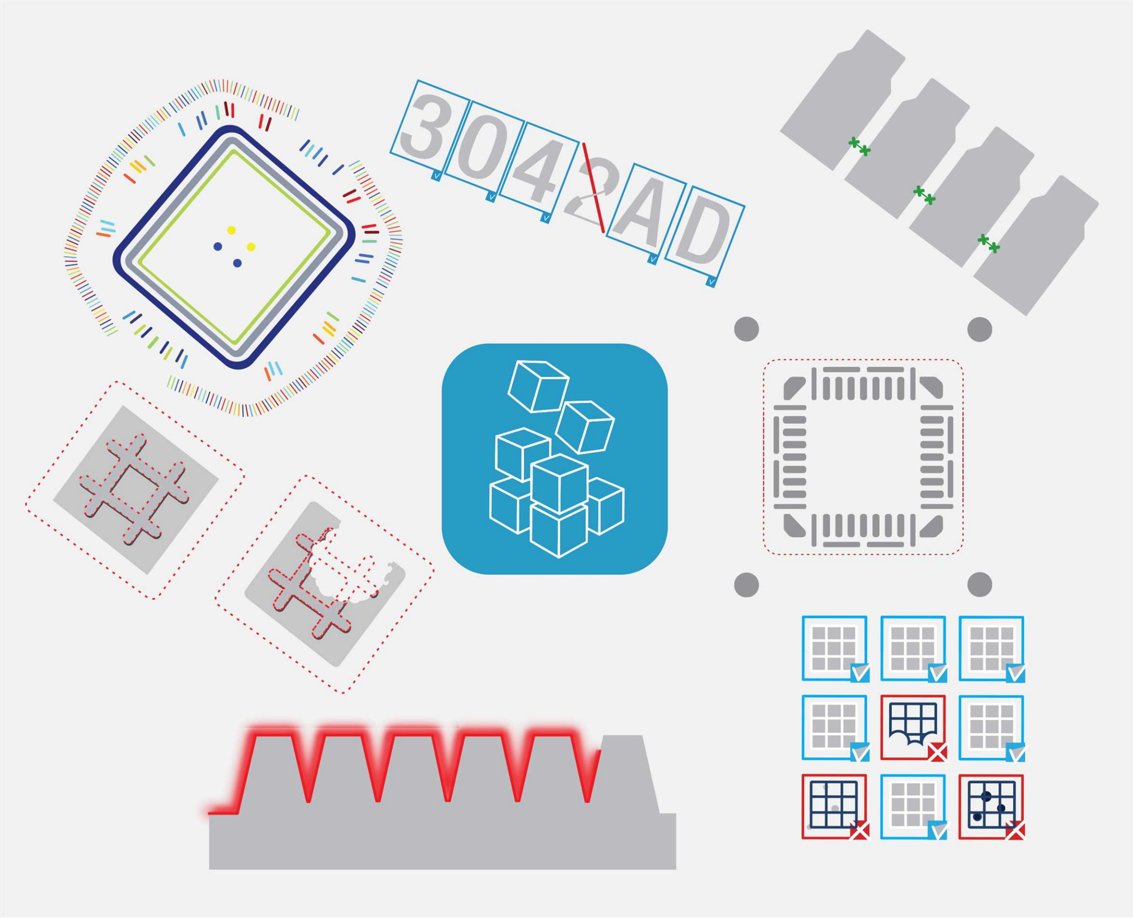
euresys
Empowering Computer Vision

USER GUIDE

# Open eVision

## Easy3D Compatibility with Lucid Helios 3D Sensors



## Terms of Use

EURESYS s.a. shall retain all property rights, title and interest of the documentation of the hardware and the software, and of the trademarks of EURESYS s.a.

All the names of companies and products mentioned in the documentation may be the trademarks of their respective owners.

The licensing, use, leasing, loaning, translation, reproduction, copying or modification of the hardware or the software, brands or documentation of EURESYS s.a. contained in this book, is not allowed without prior notice.

EURESYS s.a. may modify the product specification or change the information given in this documentation at any time, at its discretion, and without prior notice.

EURESYS s.a. shall not be liable for any loss of or damage to revenues, profits, goodwill, data, information systems or other special, incidental, indirect, consequential or punitive damages of any kind arising in connection with the use of the hardware or the software of EURESYS s.a. or resulting of omissions or errors in this documentation.

This documentation is provided with Open eVision 2.12.0 (doc build 1129).

# Easy3D Compatibility with LUCID Helios 3D Sensors

## Introduction

Helios is a 3D time of flight camera based on the **Sony DepthSense** sensor.

The specifications are available on the manufacturer website:
`https://thinklucid.com/helios-time-of-flight-tof-camera/`

- This document explains how to use the 3D data coming from these sensors with **Open eVision** 3D libraries and tools.
- A sample application distributed with source code demonstrates that integration.

### *Resources*

This document and the sample applications are based on the following resources:

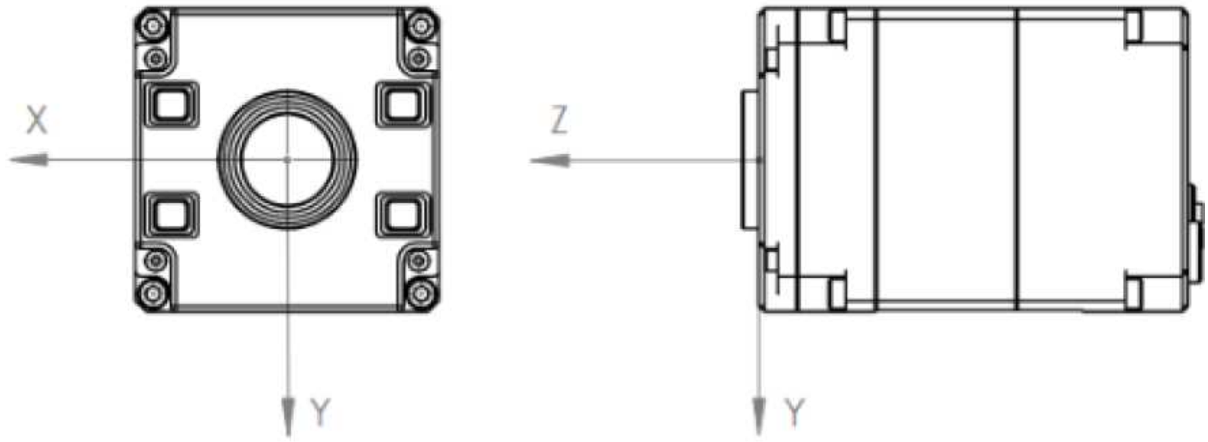
- □ **Helios** camera HLS003S-001, firmware version 1.13.0.0
- □ **Lucid Arena** SDK v1.0.20.4
- □ **Open eVision** 2.12
- □ Microsoft Visual Studio 2017

## Features

- The **Helios** camera is **Genicam** compliant and produces the range data in the following formats:

| Range data format | Description | Bits per pixel |
|---|---|---|
| `Coord3D_ABCY16` | 4-channel point cloud XYZ + intensity, 16-bits per channel | 64 |
| `Coord3D_ABC16` | 3-channel point cloud XYZ, 16-bits per channel | 48 |
| `Coord3D_C16` | Depth map Z plane | 16 |

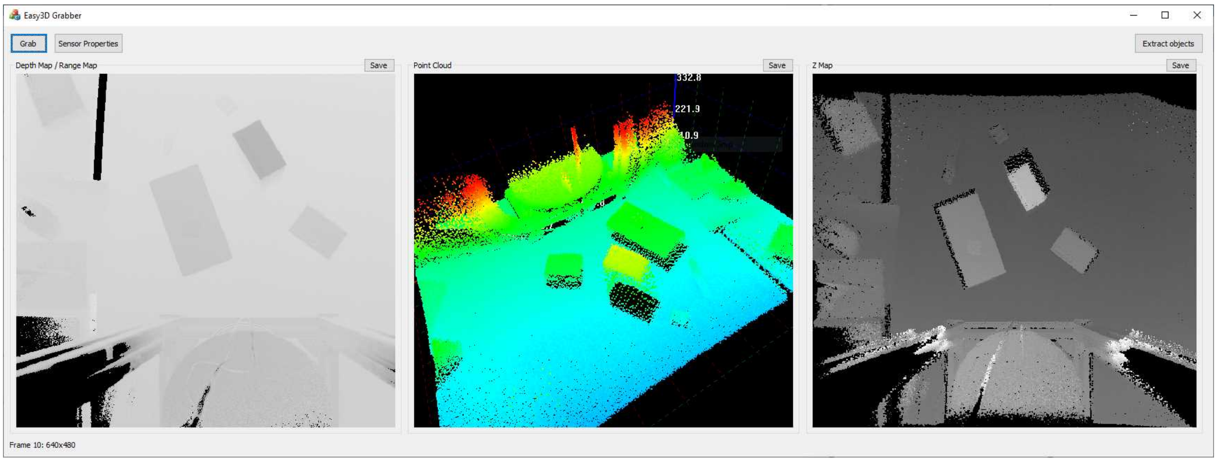
- The XYZ positions are expressed in a coordinate system centered on the camera:
  - □ To convert the 16 bits X, Y and Z values to millimeters, apply a scale factor (`Scan3dCoordinateScale` parameter).
  - □ The position is invalid if the value of one of the coordinates is 0x8000.



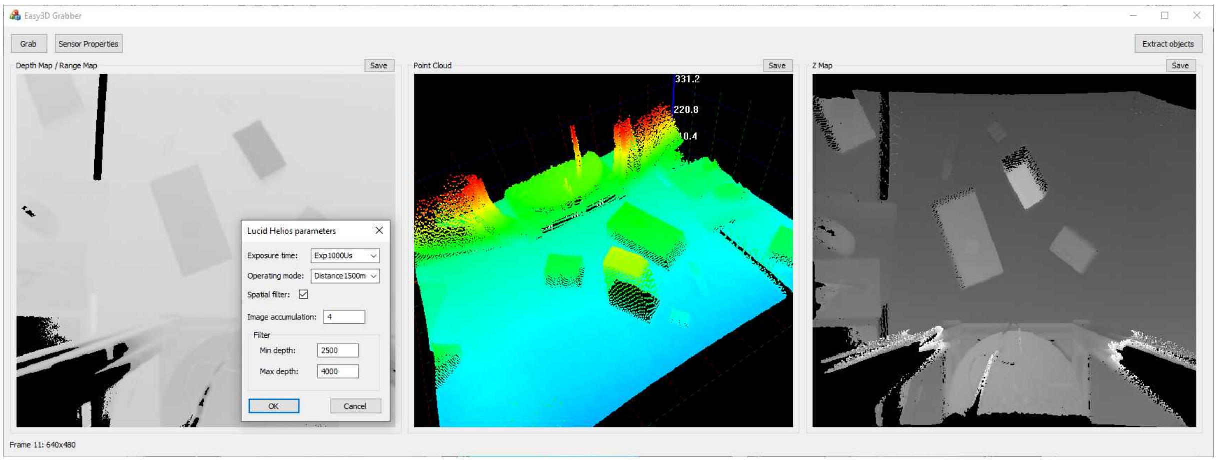
## Easy3DGrab sample application

**Easy3DGrab** is distributed with C++ source code as an **Open eVision** additional resource.

- □ It features the acquisition of **Helios** range data, the conversion to depth maps, point clouds and ZMaps.
- □ You can save these representations.
- □ Each time you click on the Grab button, a new image is captured.
- □ Some camera parameters are exposed in the Sensor Properties dialog.
- □ An optional object extraction function is exposed but only available if the **Easy3DObject** license is installed.



The Easy3DGrab application: EDepthMap (left), EPointCloud (center), EZMap (right)



Lucid Helios parameters

Exposure time: Exp1000Us

Operating mode: Distance1500m

Spatial filter: ☑

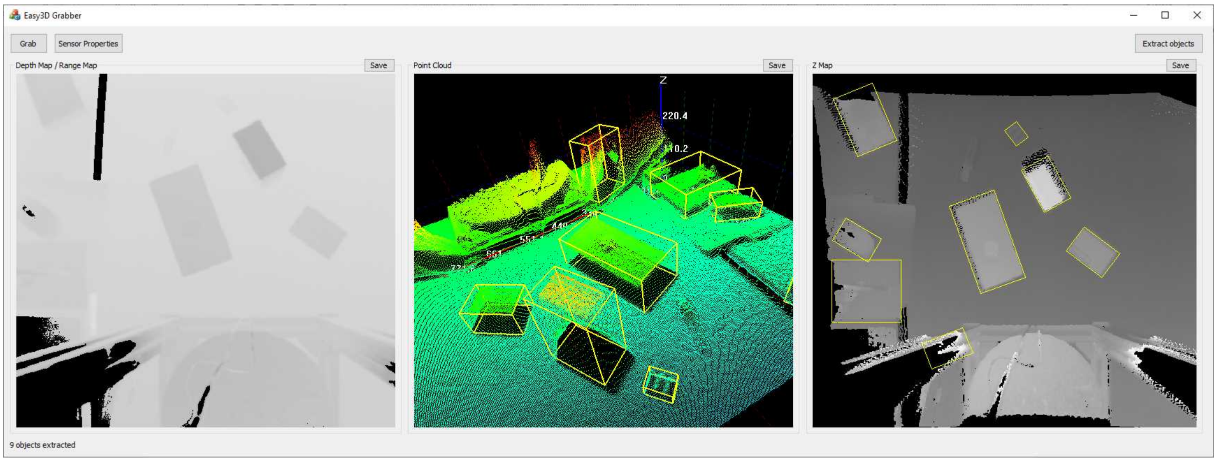
Image accumulation: 4

Filter

Min depth: 2500

Max depth: 4000

Tuning the Helios camera parameters to filter the range data



Automatic extraction of 3D objects with Easy3DObject library

# C++ code sample to convert Helios range data to Easy3D objects

## *Converting an Helios range map to a depth map*

Here is the code snippet to fill an `Easy3D::EDepthMap16(map16)` object from an `Arena::IImage(image)`:

```
int w = (int)image->GetWidth();
int h = (int)image->GetHeight();
uint64_t pf = image->GetPixelFormat();
int pix_size = (int)image->GetBitsPerPixel() / 16;

const uint16_t* data = (const uint16_t*)image->GetData();
if (pf == Coord3D_ABC16 || pf == Coord3D_ABCY16)
  data += 2;    // move data origin to the Coordinate_C start offset

// Initialize EDepthMap16
map16.SetSize(w, h);
uint16_t undef = map16.GetUndefinedValue().Value;

// Loop on lines and columns and copy valid Z positions to the depth map
for (int y = 0; y < h; ++y)
{
  const uint16_t* src = data + y * w * pix_size;
  uint16_t* dst = (uint16_t*)map16.GetBufferPtr(0, y);
  for (int x = 0; x < w; ++x)
  {
    if (*src != 0x8000)
      *dst = *src;
    else
      *dst = undef;

    src += pix_size;
    dst++;
  }
}
```

### Converting an Helios range map to a point cloud

Here is a code snippet to fill an `Easy3D::EPointCloud(pc)` from an `Arena::IImage(image)`.

**NOTE**

Only `Coord3D_ABC16` and `Coord3D_ABCY16` image formats are supported.

```
int w = (int)image->GetWidth();
int h = (int)image->GetHeight();
uint64_t pf = image->GetPixelFormat();
const int16_t* data = (const int16_t*)image->GetData();
int pix_size = (int)image->GetBitsPerPixel() / 16;
int npix = w * h;
// Initialize array for converted 3D points
std::vector<Easy3D::E3DPoint> pts;
pts.reserve(npix);
Easy3D::E3DPoint p;

for (int i = 0; i < npix; ++i, data += pix_size)
{
  if (src[0] != -32768)        // Test invalid position
  {
    p.X = coordA_scale * data[0];       // Coordinate_A
    p.Y = coordB_scale * data[1];       // Coordinate_B
    p.Z = coordC_scale * data[2];       // Coordinate_C
    pts.push_back(p);
  }
}
// Fill point cloud
pc.AddPoints(pts);
```

### ZMap

- You cannot generate a ZMap (a gray scale image encoding distance from a reference plane) directly from **Helios** camera data.
- Generate a ZMap from the point cloud with the `Easy3D::EPointCloudToZMapConverter` class.

**TIP**

The sample application **Easy3DGrab** implement the `EDepthMap16`, `EPointCloud` and `EZMap` conversions.