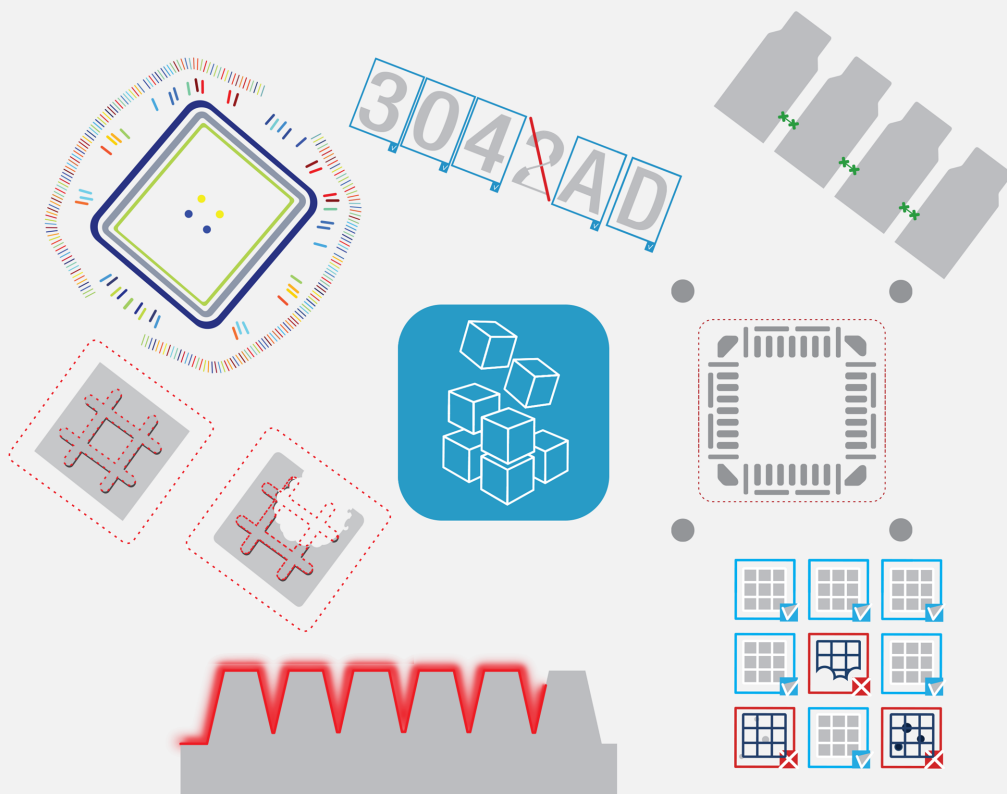


Open eVision

Easy3D Compatibility with Heliotis 3D Sensors



This documentation is provided with **Open eVision 23.04.3** (doc build **1184**).
www.euresys.com

This documentation is subject to the General Terms and Conditions stated on the website of **EURESYS S.A.** and available on the webpage <https://www.euresys.com/en/Menu-Legal/Terms-conditions>. The article 10 (Limitations of Liability and Disclaimers) and article 12 (Intellectual Property Rights) are more specifically applicable.

Easy3D Compatibility with Heliotis 3D Sensors

Introduction

The **Heliotis** 3D scanners are based on white light interferometry.

The specifications are available on the manufacturer website:

<https://www.heliotis.com/en/sensors>



- 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. This application is freely available in the **Easy3D Sensors Compatibility** additional resources package on **Euresys** website.

Resources

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

- A **helinspect H8** device with **WLI8 2x** optics
- The firmware version 22082401
- The **C4Utility** version 1.9.0 from **Heliotis**.
- **Mimic**, an open-source application used as a higher-level interface to **GenICam** and **GenTL** (MIT license)
- **CMake** version 3.23 (to build **Mimic**)
- **Open eVision** 22.12
- **Microsoft Visual Studio** 2017

Features

- The sensor gives access to the computed 3D-data as an image (ZMap), either composed of 16-bit unsigned integers or 32-bit floats.
- You can also retrieve the reflectance of the object (16-bit unsigned integers).
- Refer to the latest sections for code snippets showing how to transform these 3D-data to **Easy3D** objects.

Easy3DGrab sample application

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

It demonstrates the acquisition of **Heliotis** sensors data to **Easy3D** objects.

- It features the import of the intensity to an EZMap16 and the conversion of that EZMap16 to a point cloud.
- You can save these representations.
- Click on the **Grab** button to acquire a new image.
- Open the **Sensor Properties** dialog to change the main properties of the sensor.
- The **Object extraction** function is exposed but you can use it only with the **Easy3DObject** license.
- You can also perform a **Ground leveling** if the scene context is compatible.

Building Mimic

To use the **Easy3DGrab** sample application, you must first build **Mimic**:

1. Open **CMake** (the source code is in Easy3DGrab\Easy3DGrab\HeliotisGrabberApp\mimic).
2. Built it into Easy3DGrab\Easy3DGrab\HeliotisGrabberApp\mimic\build.
- ▶ **CMake** generates a **Visual Studio** solution.
3. Open the solution and build the **Mimic** target in Debug x64 and Release x64 mode.

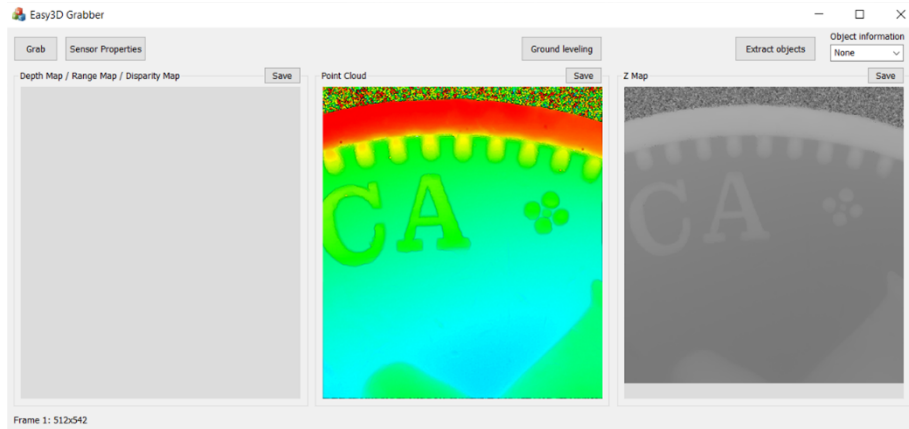
NOTE: The **Easy3DGrab** solution is configured to include the outputs of this build.

Connecting to the sensor

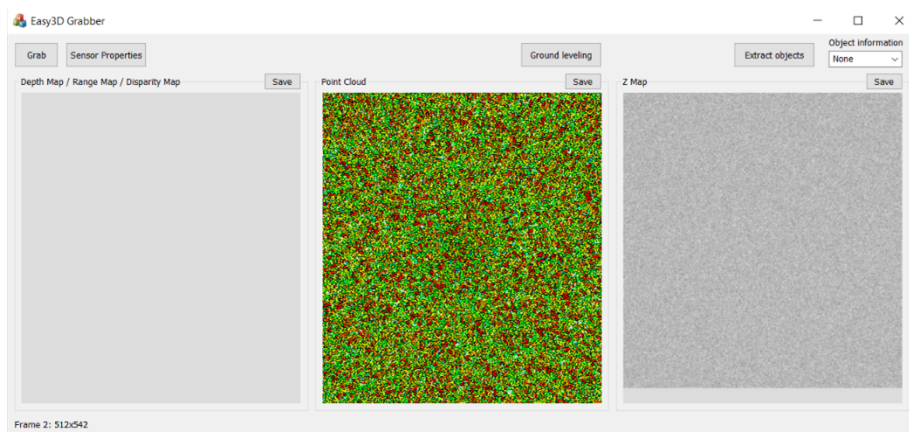
1. Connect your sensor to an Ethernet port with a fixed IP, compatible with the IP of the sensor.
 - For ex.: our sensor had IP 192.168.2.71 and we had to configure our IP to 192.168.2.100.
2. Use the **Heliotis** updateTool program to check if your device is detected.

Acquiring data

- **Heliotis** sensors are able to capture data with a very high resolution but they require the object to be at a specific distance from the sensor.
 - For example, for an **heliInspect H8** with 2x optics, that distance is about 4.5 cm.
- If the scan you acquire looks like a diffuse point cloud, try to modify the scanPosition setting in the sensor properties.



The Easy3DGrab application:
No depth map displayed (left), point cloud (center), ZMap (right)



The Easy3DGrab application:
example of a failed scan due to a bad scanPosition setting

[C++ code sample to convert Heliotis formats \(GentTL\) to Easy3D objects](#)

[Converting an EZMap to an EPointCloud](#)

As explained above, you can create an Easy3D::EPointCloud from an Easy3D::EZMap.

To retrieve the Easy3D::EZMap from the sensor, see the following sections.

```
Easy3D::EPointCloud pc;
Easy3D::EZMap16 zmap;
GetZMap(zmap);
Easy3D::EZMapToPointCloudConverter converter;
converter.Convert(zmap, pc);
```

Converting the “Intensity” GenTL buffer to an EZMap16

Here is the code snippet to fill an Easy3D::EZMap16 object from an acquired “Intensity” buffer:

```
// buffer requested must be "Intensity"
std::unique_ptr<mimic::Mimic_Buffer> buffer;
float zScale; // must contain zmap's scale
float zOffset; // must contain zmap's offset
float pixel_side_length; // must contain physical size of a pixel
Easy3D::EZMap16 zmap;

int64_t width = buffer->getPartWidth(0);
int64_t height = buffer->getPartHeight(0);

zmap.SetSize((int)width, (int)height);
zmap.SetZResolution(zScale);

zmap.SetXResolution(pixel_side_length);
zmap.SetYResolution(pixel_side_length);

Easy3D::E3DTransformMatrix mat = Easy3D::E3DTransformMatrix::CreateIdentityMatrix();
mat.SetValue(3, 2, zOffset);
zmap.SetMapToWorldMatrix(mat);

size_t xPadding = zmap.GetRowPitch() - (width * 2);
size_t bufferSize = zmap.GetRowPitch() * height;
buffer->getPartData((uint16_t*)zmap.GetBufferPtr(), &bufferSize, xPadding, int64_t(0));
```

Converting the “Range” GenTL buffer to an EZMap32f

Here is the code snippet to fill an Easy3D::EZMap32f object from an acquired “Range” buffer:

```
// buffer requested must be "Range"
std::unique_ptr<mimic::Mimic_Buffer> buffer;
float pixel_side_length; // must contain physical size of a pixel
Easy3D::EZMap32f zmap;

int64_t width = buffer->getPartWidth(0);
int64_t height = buffer->getPartHeight(0);

zmap.SetSize((int)width, (int)height);
zmap.SetZResolution(zScale);

zmap.SetXResolution(pixel_side_length);
zmap.SetYResolution(pixel_side_length);

Easy3D::E3DTransformMatrix mat = Easy3D::E3DTransformMatrix::CreateIdentityMatrix();
mat.SetValue(3, 2, zOffset);
zmap.SetMapToWorldMatrix(mat);

size_t xPadding = zmap.GetRowPitch() - (width * 4);
size_t bufferSize = zmap.GetRowPitch() * height;
buffer->getPartData((float*)zmap.GetBufferPtr(), &bufferSize, xPadding, int64_t(0));
```