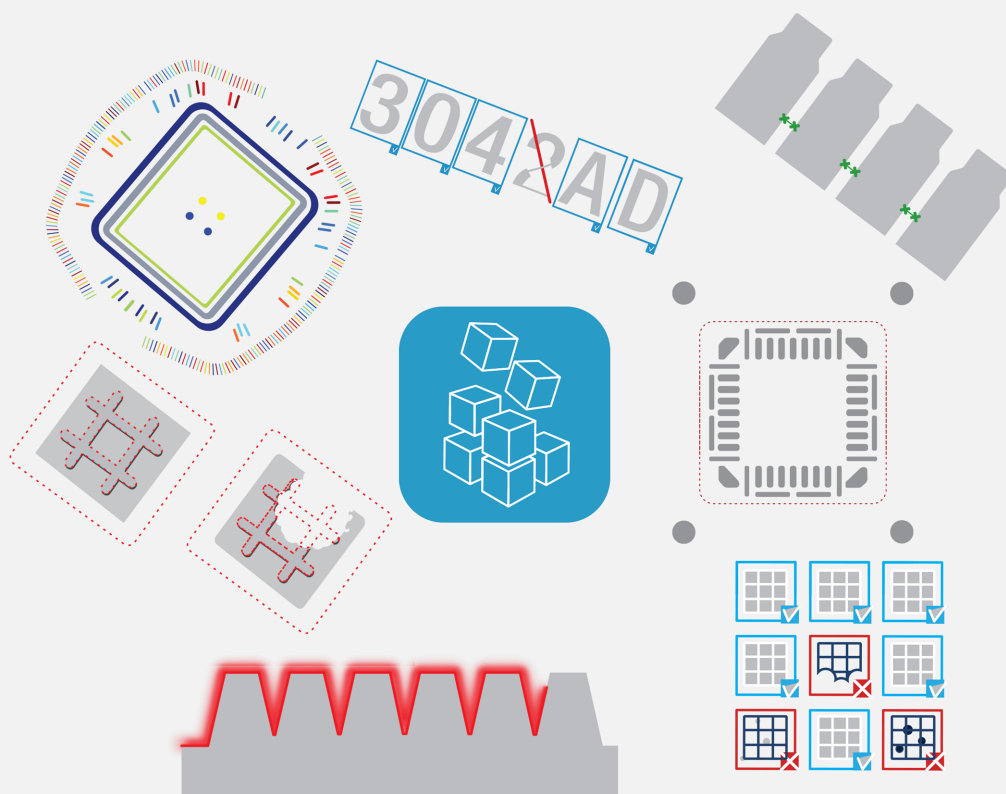


Open eVision

Easy3D Compatibility with Mech-Mind



This documentation is provided with **Open eVision 2.17.2** (doc build **1162**).
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 Mech-Mind

Introduction

The **Mech-Eye** 3D cameras from **Mech-Mind** are structured-light cameras for industrial applications.

The specifications are available on the manufacturer website:

<https://www.mech-mind.de/en/components/mech-eye>



- This document explains how to use the 3D data coming from these cameras 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:

- **Mech-Eye** Pro S Enhanced (it should also be compatible with all other **Mech-Eye** cameras).
- **Mech-Eye SDK**
- **Open eVision** 2.17
- Microsoft Visual Studio 2017

The **Mech-Eye SDK** is available at https://github.com/MechMindRobotics/mecheye_cpp_interface

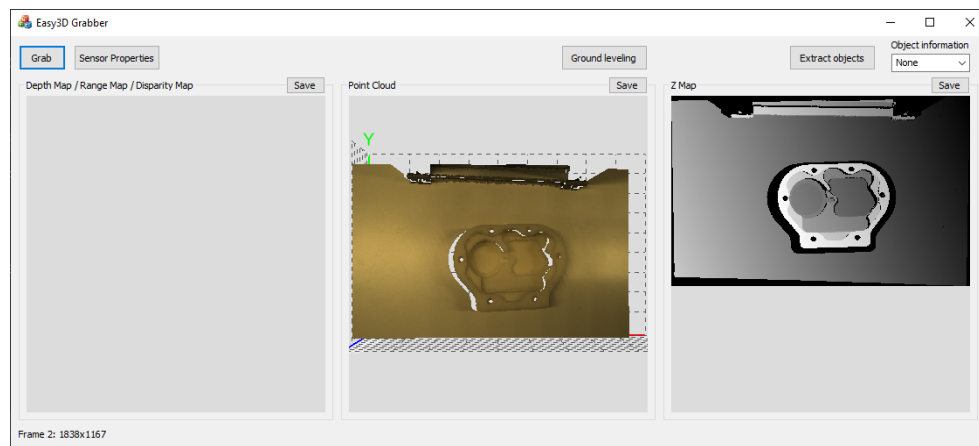
Features

- The **Mech-Eye SDK** exposes point clouds from PCL:
 - `pcl::PointCloud<pcl::PointXYZRGB>`
 - A `pcl::PointXYZRGB` corresponds to 128 bits:
 - 3×32 bits for the XYZ
 - a 32-bit float for RGB (with 8 bits not used)

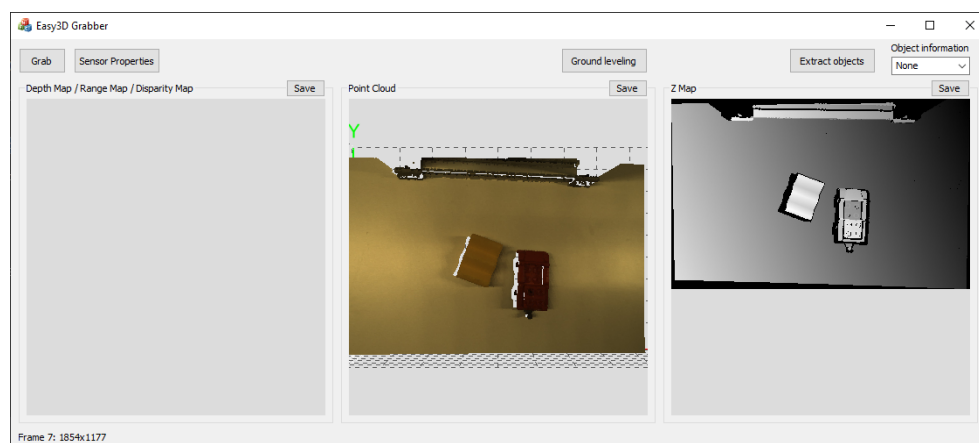
Easy3DGrab sample application

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

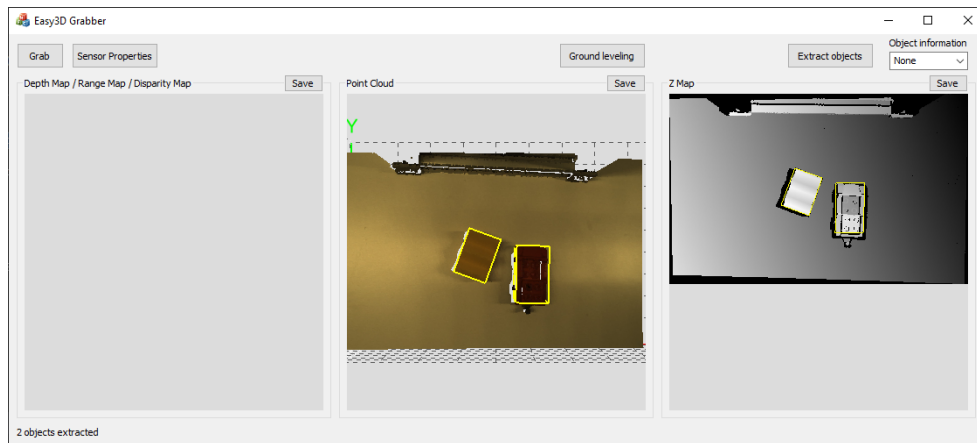
- It features the import of the `pcl::PointCloud<pcl::PointXYZRGB>` formats and the conversion to **Open eVision** formats (EPointCloud to EZMap).
- You can save these representations.
- Click on the **Grab** button to acquire a new image.
- Open the **Sensor Properties** dialog to:
 - ☐ Modify the exposure mode.
 - ☐ Modify the exposure time.
- The **Object extraction** function is exposed but you can use it only with the **Easy3DObject** license.
- You can also perform a **Ground leveling**.



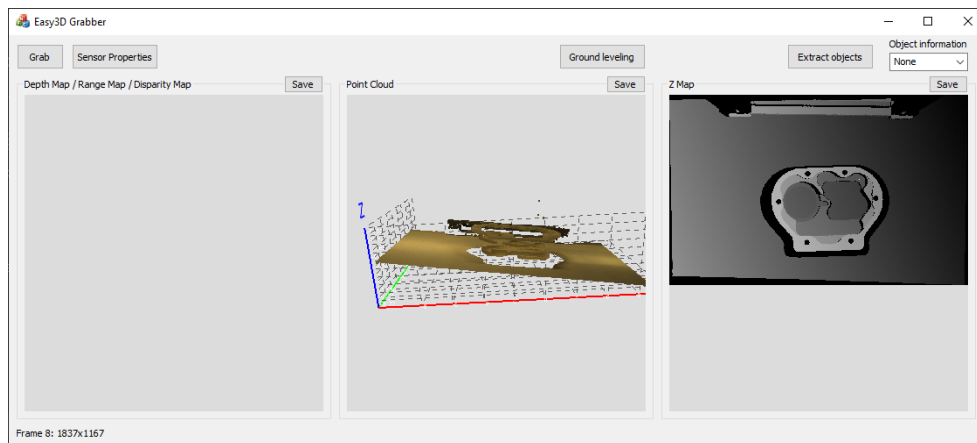
The Easy3DGrab application:
EDepthMap not available (left), EPointCloud (center), EZMap (right)



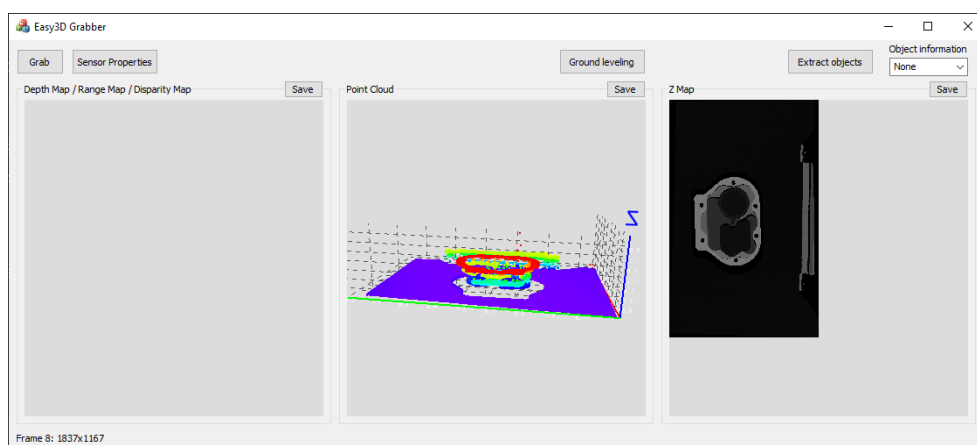
The Easy3DGrab application: an EPointCloud (center) retrieved with colors



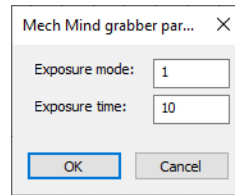
The Easy3DGrab application: extraction of objects



The Easy3DGrab application: before ground leveling



The Easy3DGrab application: after ground leveling



Setting the 3D sensor parameters

C++ code sample to convert **Mech-Mind** formats to **Easy3D** objects

Converting a `pcl::PointCloud<pcl::PointXYZRGB>` to an `EPointCloud`

Here is the code snippet to fill an `Easy3D::EPointCloud` object from a **Mech-Eye** `pcl::PointCloud<pcl::PointXYZRGB>`:

```
CameraClient camera;

// Connecting to camera
if (!camera.connect("192.168.1.118"))
{
    // Connection to camera failed
    throw(std::runtime_error("Connection to the camera failed"));
}

const pcl::PointCloud<pcl::PointXYZRGB> rgbCloud = camera.captureRgbPointCloud();

const size_t nbPoints = rgbCloud.size();

std::vector<Easy3D::E3DPoint> points;
points.reserve(nbPoints);
std::vector<EC24A> colors;
colors.reserve(nbPoints);

for (size_t i = 0; i < nbPoints; ++i)
{
    const pcl::PointXYZRGB& p = rgbCloud[i];
    if (p.x != 0 && p.y != 0 && p.z != 0)
    {
        points.emplace_back(-p.x, p.y, -p.z);
        float color = p.rgb;
        colors.emplace_back(p.r, p.g, p.b, 255);
    }
}

Easy3D::EPointCloud pointcloud;
pointcloud.AddPoints(points);
pointcloud.FillAttributeBuffer((int)Easy3D::E3DAttribute_Color, colors.data());
```

ZMap

- You cannot generate a ZMap (a gray scale image encoding distance from a reference plane, also called an orthographic projection of the point cloud) directly from the **Mech-Eye** 3D cameras.
- Generate a ZMap from the point cloud with the `Easy3D:EPointCloudToZMapConverter` class.

**TIP**

The sample application **Easy3DGrab** implements these conversions.