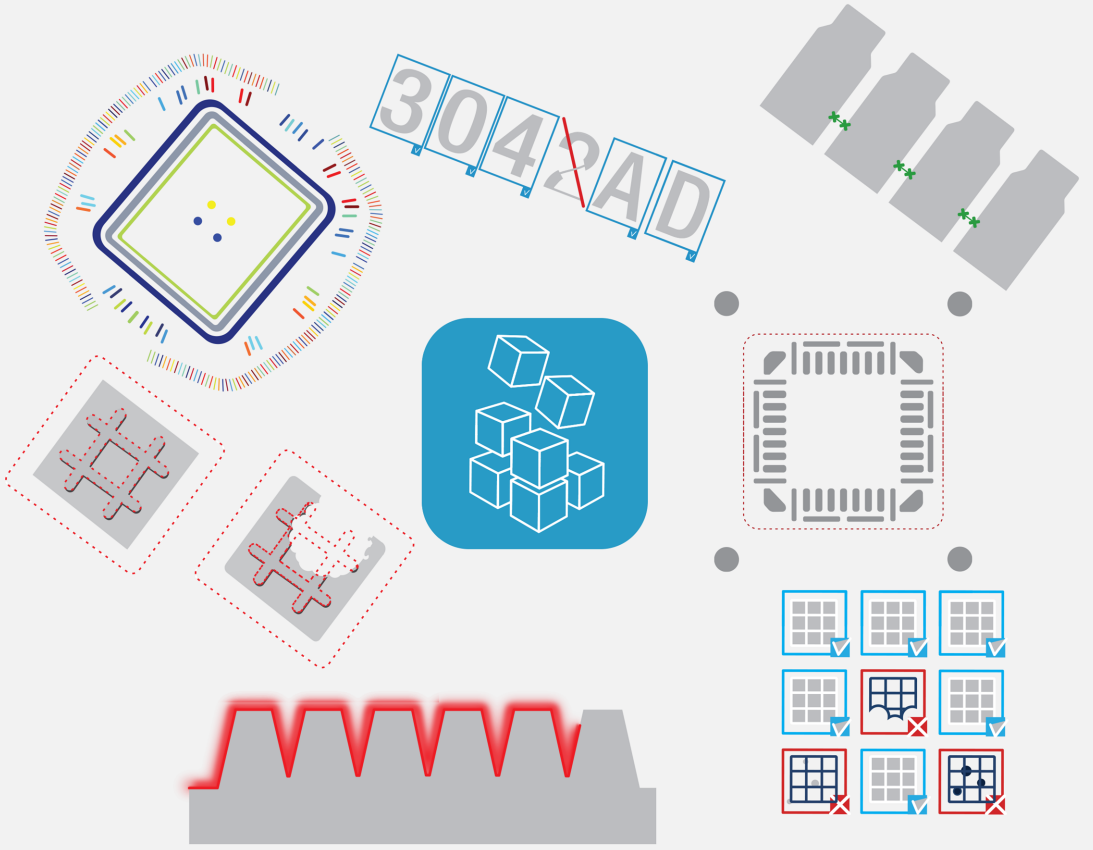
euresys
Empowering Computer Vision

USER GUIDE

Open eVision

Easy3D Compatibility with IDS Ensenso 3D Sensors



This documentation is provided with Open eVision 2.14.0 (doc build 1141).
www.euresys.com

Easy3D Compatibility with IDS Ensenso 3D Sensors

Introduction

The **IDS Ensenso** sensors are active stereo cameras for industrial applications.

The specifications are available on the manufacturer website:
https://en.ids-imaging.com/ensenso-stereo-3d-camera.html

- 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** web site.

Resources

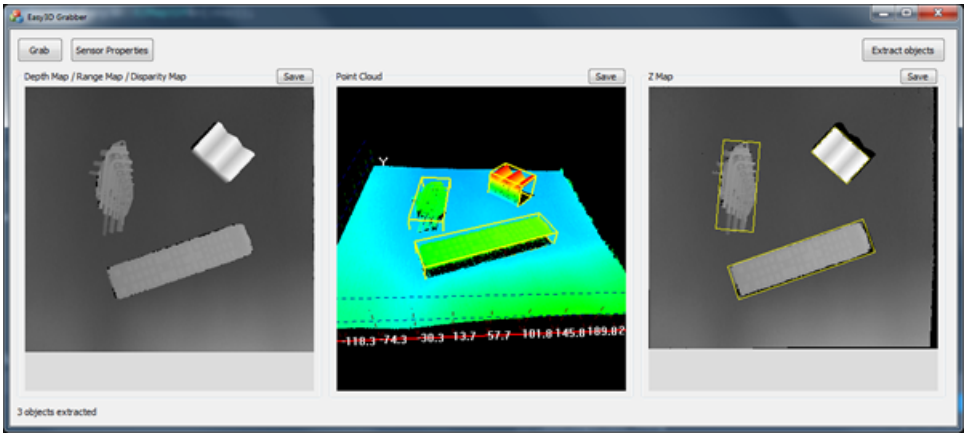
This document and the sample applications are based on the following resources:
- **Ensenso N35** device
- **Ensenso SDK** 3.0.275
- **Open eVision** 2.13
- Microsoft Visual Studio 2017

Features

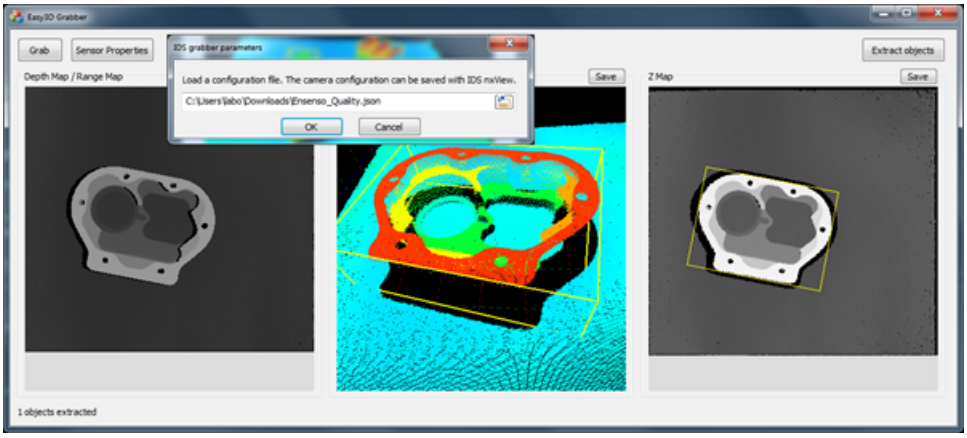
- The **Ensenso SDK** generates several depth outputs:
  - The raw and rectified images from the 2 cameras.
  - The disparity map as a 16-bit per pixel image.
  - The point map as three 32-bit float per pixel images.
  - The normal map as three 32-bit float per pixel images.
- The **Easy3DGrab** sample only uses and converts the disparity and the point maps.

Easy3DGrab sample application

Easy3DGrab is distributed with C++ source code as an **Open eVision** additional resource.
- It features the acquisition of **IDS Ensenso** range data, the conversion to depth maps, point clouds and ZMaps.
- You can save these representations.
- Click on the Grab button to acquire a new image.
- Open the Sensor Properties dialog to load a `JSON` configuration file generated by the **IDS nxView** application.
- The Object extraction function is exposed but you can use it only with the Easy3DObject license.



The Easy3DGrab application: EDepthMap (left), EPointCloud (center), EZMap (right) with automatic extraction of 3D objects with Easy3DObject library (yellow boxes)



The Sensor Properties button allows to load and apply a configuration file

C++ code sample to convert IDS Ensenso data to Easy3D objects

Converting a disparity map

You can generate `EDepthMap16` and `EPointCloud` objects from the disparity and point maps.

- The range of the disparity map is shifted to fit the unsigned short format of an `EDepthMap16`.
- The invalid pixels in the disparity map are marked with the value `0x8000` (`-32768`). You must translate them to `0`.

Here is the code snippet to fill an `Easy3D::EDepthMap16` object from a disparity map:

```
NxLibItem dispMap = camera[itmImages][itmDisparityMap];

int width, height;
dispMap.getBinaryDataInfo(&width, &height, 0, 0, 0, 0);

std::vector<int16_t> binaryData;
dispMap.getBinaryData(binaryData, 0);

// convert from signed short to unsigned short
// search for the minimum value
int d_min = 32767;
for (int i = 0; i < binaryData.size(); ++i)
{
  int16_t d = binaryData[i];
  if (d > -32768 && d < d_min)
    d_min = d; }

// Copy buffer to EDepthMap16
map.SetSize(width, height);

int i = 0;
for (int y = 0; y < height; ++y)
{
  uint16_t* dst = (uint16_t*)map.GetBufferPtr(0, y);

  // Copy disparity values to depth buffer z-values
  for (int x = 0; x < width; ++x, ++i)
  {
    int d = binaryData[i];

    if (d > -32768) // skip undefined pixel
      dst[x] = uint16_t(d - d_min);
    else
      dst[x] = 0;
  }
}
```

Converting a point map

A `PointMap` is a 3-channel image of 32-bit floats.

- Each pixel represents the 3 coordinates in millimeters with respect to the camera coordinate system (by default).
- The special float value `NaN` is used to mark invalid pixels.
- For consistency with other sensors, the Z and Y axis are reversed to refer the coordinates of the points to a "ground" origin, instead of a camera origin.

Here is the code snippet to fill an `Easy3D::EPointCloud` object from a `PointMap`:

```
// This converts the disparity map into XYZ data for each pixel
NxLibCommand(cmdComputePointMap).execute();

// Get info about the computed point map and copy it into a std::vector
int width, height;
camera[itmImages][itmPointMap].getBinaryDataInfo(&width, &height, 0, 0, 0, 0);

std::vector<float> pointMap;
camera[itmImages][itmPointMap].getBinaryData(pointMap, 0);

int size = width * height;

// Push valid 3D points to an EPointCloud
// with coordinate system flipping

std::vector<Easy3D::E3DPoint> pts;
pts.reserve(size);
float max = -1.0f;

Easy3D::E3DPoint p;
for (int i = 0; i < size; ++i)
{
  p.X = pointMap[3 * i];
  if (!std::isnan(p.X))
  {
    p.Y = height - pointMap[3 * i + 1];
    p.Z = pointMap[3 * i + 2];

    pts.push_back(p);
    if (p.Z > max)
      max = p.Z;
  }
}

for (int i = 0; i < pts.size(); ++i)
  pts[i].Z = max - pts[i].Z;

// Update the point cloud
point_cloud.Clear();
point_cloud.AddPoints(pts);
```

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 **IDS Ensenso SDK**.
- Generate a ZMap from the point cloud with the `Easy3D::EPointCloudToZMapConverter` class.

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