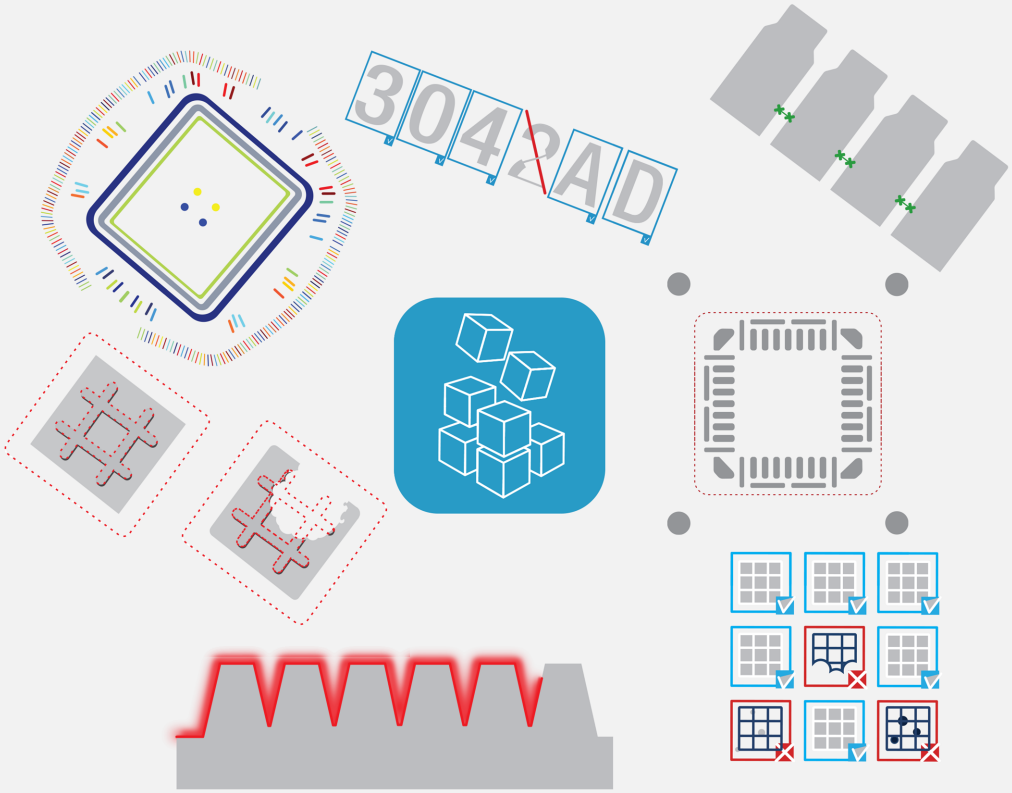
euresys
Empowering Computer Vision

USER GUIDE

Open eVision

Easy3D Compatibility with Benano 3D Scanners



© **EURESYS S.A. 2021** - Document D199EN-Easy3D Compatibility with Benano 3D Scanners-

This documentation is provided with **Open eVision 2.17.0** (doc build **1157**).
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 Benano 3D Scanners

Introduction

The **Benano** 3D sensors are structured-light cameras for industrial applications.

The specifications are available on the manufacturer website:
`http://www.benanos.com/product?id=23&lang=us`

- 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:

- **Benano** C2100 (it should also be compatible with all other **Benano** sensors, but some sensor options might not be configured through the sample).
- **Benano SDK** 1.24.7
- **Open eVision** 2.17
- Microsoft Visual Studio 2017

The **Benano SDK** is available from the manufacturer upon request.

Features

- The **Benano SDK** exposes different data types:

Format	Description	Bits per pixel
`PointCloudResult.PointCloud`	float32 array of 3D coordinate	96
`PointCloudResult.Normals`	float32 array of 3D coordinate	96
`PointCloudResult.Colors`	uint8 array of RGB or grayscale colors	24 or 8
`PointCloudResult.MeshVertices`	int32 array of mesh vertices index	96
`ZmapResult.ImageRaw`	float32 array of distances along the z-axis	32
`ZmapResult.Image2D`	uint8 array of RGB or grayscale colors	24 or 8

- The XYZ positions in the different data types are expressed in a coordinate system that has its origin on the center of the object with a z-axis pointing towards the camera.

Easy3DGrab sample application

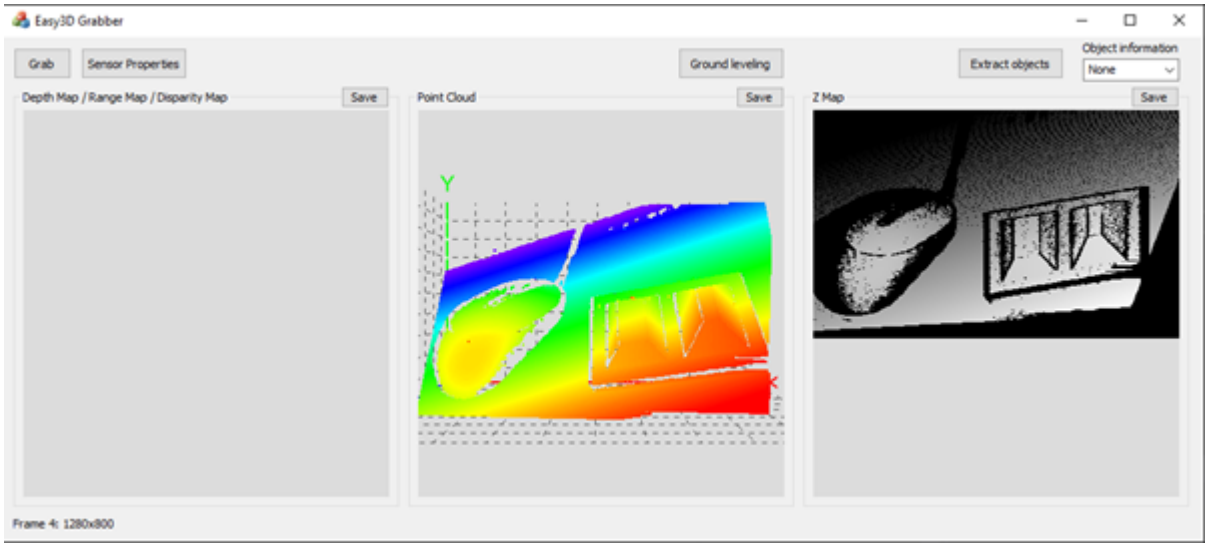
Easy3DGrab is distributed with C++ source code as an **Open eVision** additional resource.

- It features the import of the `PointCloudResult.PointCloud`, `PointCloudResult.Normals`, `PointCloudResult.Colors` and the `ZmapResult.ImageRaw` formats and the conversion to **Open eVision** formats (`EPointCloud` and `EZMap`).
- You can save these representations.
- Click on the `Grab` button to acquire a new image.
- Open the `Sensor Properties` dialog to:
  - Modify the brightness of the sensor.
  - Modify the gain of the sensor.
  - Modify the exposure mode of the sensor.
  - Choose to retrieve Normals.
  - Choose to retrieve Colors.

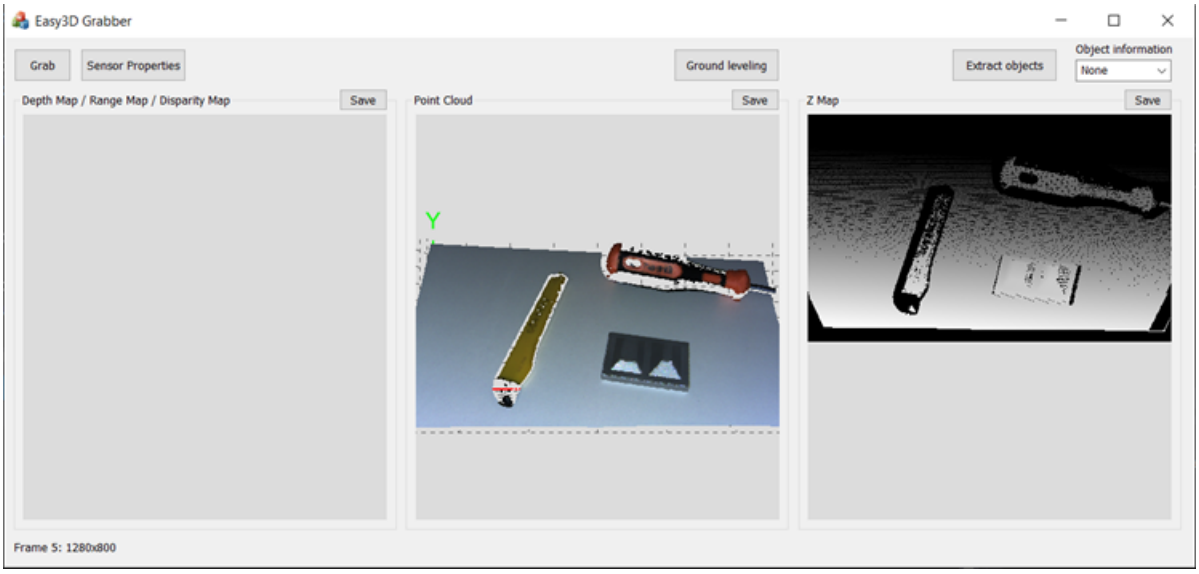
NOTE: When you modify whether to retrieve Normals / Colors or not, the configuration of the scanner takes a few seconds.

- The `Object extraction` function is exposed but you can use it only with the `Easy3DObject` license.
- You can also perform a `Ground leveling`.

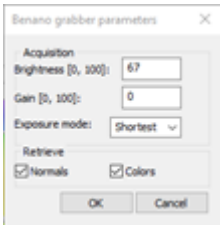
NOTE
When the application starts, it triggers an initial scan from the sensor. This takes a few seconds and lights the sensor.



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



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



Setting the 3D sensor parameters

C++ code sample to convert Benano formats to Easy3D objects

The samples presented below are a bit different than the **Easy3DGrab** application.

- The SDK of **Benano** uses a callback to retrieve the data.
- In **Easy3DGrab**, the callback gets a reference to the current object that will contain the `EPointCloud` / `EZmap32f`.
- In the samples, for simplicity, the callback receives a reference to the `EPointCloud` / `EZmap32f` directly.

Also for simplicity, the code samples do not handle the asynchronity of the callback.

Converting a PointCloudResult (with normals and colors) to an EPointCloud

Here is the code snippet to fill an `Easy3D::EPointCloud` object from a **Benano** `PointCloudResult`:

```
using namespace benano_scansdk;

// callback function
void OnViewDataResultCallback(void* cbOwner, CViewDataResult* viewResult)
{
  Easy3D::EPointCloud* pointCloud = (Easy3D::EPointCloud*)cbOwner;
  std::vector<Easy3D::E3DPoint> buffer;
  buffer.reserve(viewResult->PointCloudResult.PointsCount);
   const float* const data = viewResult->PointCloudResult.PointCloud;
   for (size_t i = 0; i < viewResult->PointCloudResult.PointsCount * 3; i += 3)
     buffer.emplace_back(data[i], data[i + ↵], data[i + 2]);
   pointCloud->AddPoints(buffer);

   // get normals
   const float* const normalsData = viewResult->PointCloudResult.Normals;
   if (normalsData != nullptr)
   {
     pointCloud->AllocateAttributeBuffer(Easy3D::E3DAttribute_Normal, Easy3D::E3DPoint(0.f, 0.f, 0.f));
     Easy3D::E3DPoint* const normals = (Easy3D::E3DPoint*) pointCloud->GetAttributeBuffer(Easy3D::E3DAttribute_
Normal);
     for (size_t i = 0; i < viewResult->PointCloudResult.PointsCount; i++)
       normals[i] = Easy3D::E3DPoint(normalsData[3 * i], normalsData[3 * i + ↵], normalsData[3 * i + 2]);
   }

   // get colors
   const UINT8* const colorsData = (const UINT8*)viewResult->PointCloudResult.Colors;
   if (colorsData != nullptr)
   {
     pointCloud->AllocateAttributeBuffer(Easy3D::E3DAttribute_Color, EC24A(0u, 0u, 0u, 0u));
     EC24A* const colors = (EC24A*)pointCloud->GetAttributeBuffer(Easy3D::E3DAttribute_Color);
     if (viewResult->PointCloudResult.ColorChannelCount == ↵)
     {
       for (size_t i = 0; i < viewResult->PointCloudResult.PointsCount; i++)
         colors[i] = EC24A(colorsData[i], colorsData[i], colorsData[i], 255u);
     }
     else  // ColorChannelCount == 3
     {
       for (size_t i = 0; i < viewResult->PointCloudResult.PointsCount; i++)
         colors[i] = EC24A(colorsData[3 * i], colorsData[3 * i + ↵], colorsData[3 * i + 2], 255u);
     }
   }
 }

// Configure device
CScanManager* scanManager = new CScanManager();
CInitialParam initParam;
```

```
initParam.MovableDeviceType = eSDKMovableDeviceType::SDK_NoDevice;
initParam.CalibrationFilePath = "C:\\Program Files\\Benano\\CalibrationFiles\\";
initParam.MaxSolverCnt = 1;
scanManager->InitialScanner(initParam);

// Enable normals and colors computation for point cloud
CParameterCtrl& paramController = scanManager->GetParameterController();
CScanParam* scanParam = paramController.GetScanParam();
scanParam->SetEnableNormalCalculation(true);
scanParam->SetEnablePointCloudColorOutput(true);
paramController.ApplyScanParam(scanParam);

// Capture pointcloud
CScanCtrl& scanCtrl = scanManager->GetScanController();
//1) Set scan result callback
CScanResultCallback rltCallback;
Easy3D::EPointCloud result;
rltCallback.SetViewDataResultCallback(&result, OnViewDataResultCallback);
//2) Create scan background thread
scanCtrl.StartScan(rltCallback);

// cleanup
scanManager->ReleaseScanner();
```

Converting a ZMapResult to an EZMap32f

Here is the code snippet to fill an `Easy3D::EZMap32f` object from a **Benano** `ZMapResult`:

```
using namespace benano_scansdk;

// callback function
void OnViewDataResultCallback(void* cbOwner, CViewDataResult* viewResult)
{
  Easy3D::EZMap32f* zmap = (Easy3D::EZMap32f*)cbOwner;
  const CZmapData* const benanoZmap = viewResult->ZmapResult;
  const float* const zmapData = benanoZmap->ImageRaw;
  int width = benanoZmap->ImageWidth;
  int height = benanoZmap->ImageHeight;
  zmap->SetSize(width, height);
  zmap->SetResolution(Easy3D::E3DPoint(float(benanoZmap->ImageResolutionX) / 1000.f,
    float(benanoZmap->ImageResolutionY) / 1000.f, 1.f));

  // benano's zmaps are centered on the center of the image
  // we are centered on the lower left corner of the image
  float biasX = float(width / 2) * zmap->GetXResolution();
  float biasY = float(height / 2) * zmap->GetYResolution();
  Easy3D::E3DTransformMatrix mat = Easy3D::E3DTransformMatrix::CreateTranslationMatrix(-biasX, -biasY, 0.f);
  zmap->SetMapToWorldMatrix(mat);

  const float undef = zmap->GetUndefinedValue().Value;
  for (int i = 0; i < height; i++)
  {
    float*const row = (float*)zmap->GetBufferPtr(0, i);
    for (int j = 0; j < width; j++)
    {
      row[j] = zmapData[width * i + j];
      if (row[j] == BENANO_UNDEF_FLOAT)
        row[j] = undef;
    }
  }
}

// Configure device
CScanManager* scanManager = new CScanManager();
CInitialParam initParam;
```

```
initParam.MovableDeviceType = eSDKMovableDeviceType::SDK_NoDevice;
initParam.CalibrationFilePath = "C:\\Program Files\\Benano\\CalibrationFiles\\";
initParam.MaxSolverCnt = ►;
scanManager->InitialScanner(initParam);

// Enable zmap computation
CParameterCtrl& paramController = scanManager->GetParameterController();
CPostProcessParam* postProcParam = paramController.GetPostProcessParam();
postProcParam->Data.ZMap_bEnableZMapGeneration = true;
paramController.ApplyPostProcessParam(postProcParam);

// Capture pointcloud
CScanCtrl& scanCtrl = scanManager->GetScanController();
//►) Set scan result callback
CScanResultCallback rltCallback;
Easy3D::EZMap32f result;
rltCallback.SetViewDataResultCallback(&result, OnViewDataResultCallback);
//2) Create scan background thread
scanCtrl.StartScan(rltCallback);

// cleanup
scanManager->ReleaseScanner();
```

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