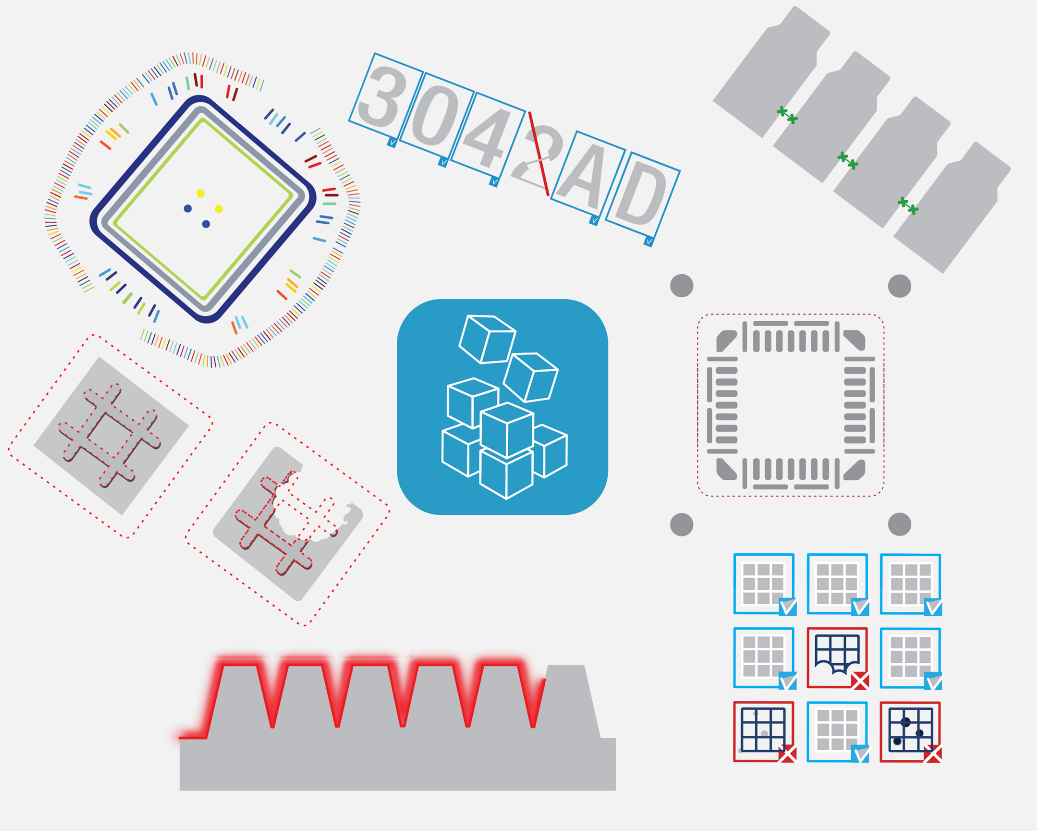
euresys
Empowering Computer Vision

USER GUIDE

Open eVision

Easy3D Compatibility with Azure Kinect 3D Sensors



This documentation is provided with Open eVision 2.15.0 (doc build 1147).
www.euresys.com

Easy3D Compatibility with Azure Kinect 3D Sensors

Introduction

The **Azure Kinect** depth camera implements the Amplitude Modulated Continuous Wave (AMCW) Time-of-Flight (ToF) principle. The camera casts modulated illumination in the near-IR (NIR) spectrum onto the scene. It then records an indirect measurement of the time that the light takes to travel from the camera to the scene and back.

The specifications are available on the manufacturer website:
https://docs.microsoft.com/en-us/azure/kinect-dk/depth-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. 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:

- □ **Microsoft Azure Kinect** DK
- □ **Azure Kinect SDK** v1.4.1
- □ **Open eVision** 2.15
- □ Microsoft Visual Studio 2017

The **Azure Kinect SDK** is available on the manufacturer website:
https://github.com/microsoft/Azure-Kinect-Sensor-SDK/blob/develop/docs/usage.md

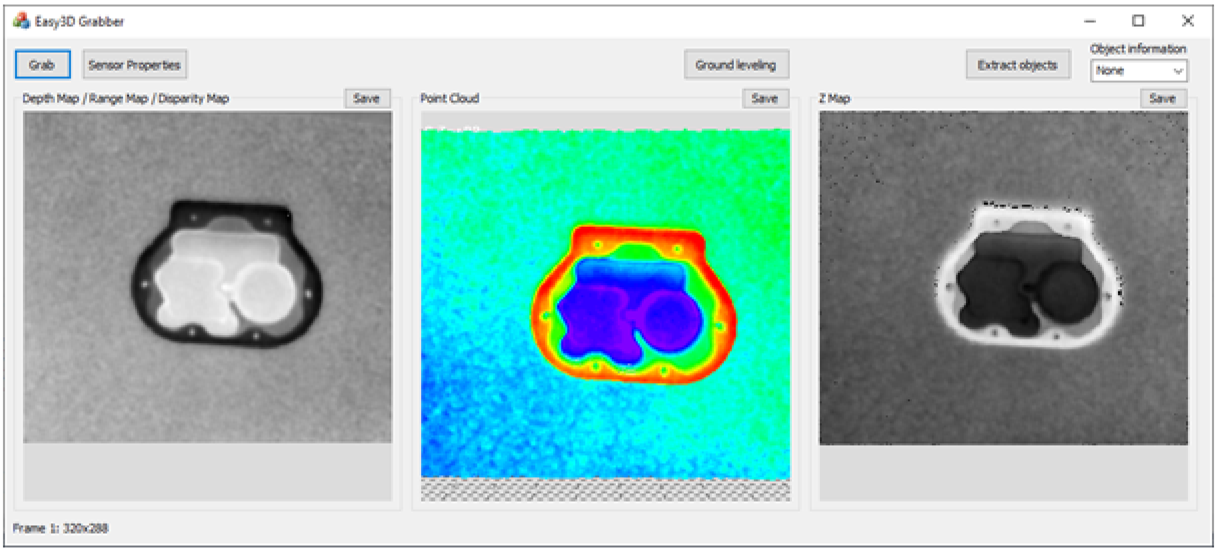
Features

- The **Azure Kinect SDK** provides a range map (depth image) as an array of unsigned int (16-bit per pixel). You can use a precomputed table to calibrate these data.

Easy3DGrab sample application

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

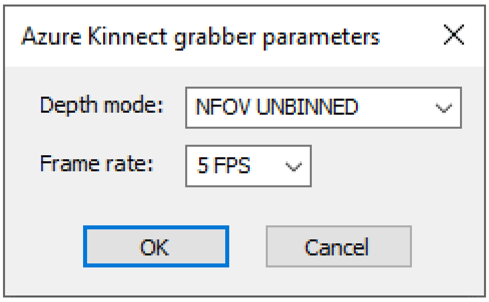
- It features the acquisition of **Azure Kinect** data and the conversion to depth maps, point clouds and ZMaps.
- You can save these representations.
- Click on the Grab button to acquire a new image.
- Open the Sensor Properties dialog to change the depth mode and the frame rate.
- The Object extraction function is exposed but you can use it only with the Easy3DObject license.
- You can also perform a Ground leveling.



Object information: None

Frame 1: 320x288

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



Azure Kinnect grabber parameters

Depth mode: NFOV UNBINNED

Frame rate: 5 FPS

The 3D sensor parameters: depth mode and frame rate

C++ code sample to convert the Azure Kinect data to Easy3D objects

Converting the Azure Kinect depth data to an EDepthMap

Here is the code snippet to fill an `Easy3D::EDepthMap16` object from the sensor depth data:

```
k4a_device_t device;

// Connect to device (Todo)

// Get a capture
k4a_capture_t capture;

if (k4a_device_get_capture(device, &capture, timeout_ms) != K4A_WAIT_
RESULT_SUCCEEDED)
{
  // Error
}

k4a_image_t depth_image = k4a_capture_get_depth_image(capture);

if (depth_image == 0)
{
  // Error
}

// Copy depth_image to map
Easy3D::EDepthMap16& dmap

int width = k4a_image_get_width_pixels(depth_image);
int height = k4a_image_get_height_pixels(depth_image);
dmap.SetSize(width, height);

uint16_t *depth_data = (uint16_t *)k4a_image_get_buffer(depth_image);

// Copy each row of the depth map
for (int y = 0; y < height; depth_data+=width, ++y)
{
  void* dst = dmap.GetBufferPtr(0, y);
  memcpy(dst, depth_data, 2 * width);
}
```

Converting the Azure Kinect depth data to an EPointCloud

Here is the code snippet to fill an `Easy3D::EPointCloud` from the sensor depth data:

- Before using the code snippet, configure the device as in this example: https://github.com/microsoft/Azure-Kinect-Sensor-SDK/blob/develop/examples/fastpointcloud/main.cpp

- Coordinate system:
  - □ The XYZ positions obtained from the **Azure Kinect** are expressed in a coordinate system centered on the camera with a Z axis toward the scene.
  - □ In **Open eVision** we usually use the Z axis in the opposite direction, so this code snippet reverts the axis.

```
// Configure the device as in the example "fastpointcloud" from
// Azure Kinect SDK and use the function create_xy_table

k4a_image_t depth_image;
// Get a capture (Todo)

// Retrieve the point cloud data
int width = k4a_image_get_width_pixels(depth_image);
int height = k4a_image_get_height_pixels(depth_image);

// Convert depth image to point cloud using precomputed table
uint16_t *depth_data = (uint16_t *)k4a_image_get_buffer(depth_image);
k4a_float2_t *xy_table_data = (k4a_float2_t *)k4a_image_get_buffer(xy_
table);

std::vector<Easy3D::E3DPoint> pts;
pts.reserve(width * height);
Easy3D::E3DPoint p;
float z_max = -1.f;
for (int i = 0; i < width * height; ++i)
{
  if (depth_data[i] != 0 && !isnan(xy_table_data[i].xy.x) &&
      !isnan(xy_table_data[i].xy.y))
  {
    p.X = xy_table_data[i].xy.x * (float)depth_data[i];
    p.Y = xy_table_data[i].xy.y * (float)depth_data[i];
    p.Z = (float)depth_data[i];

    if (p.Z > z_max)
      z_max = p.Z;

    pts.push_back(p);
  }
}

// Invert Y and Z axis to get the origin at the furthest point,
// going up to the camera
for (size_t i = 0; i < pts.size(); ++i)
{
  pts[i].Y = -pts[i].Y;
  pts[i].Z = z_max - pts[i].Z;
}

Easy3D::EPointCloud point_cloud;
point_cloud.AddPoints(pts);
```

ZMap

- You cannot generate a ZMap directly from the **Azure Kinect** 3D sensor.
- Generate a ZMap from the point cloud with the `Easy3D::EPointCloudToZMapConverter` class.

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