ADVACAM
Imaging the Unseen

Pixet SDK

Online version: https://wiki.advacam.cz/wiki/Pixet_SDK

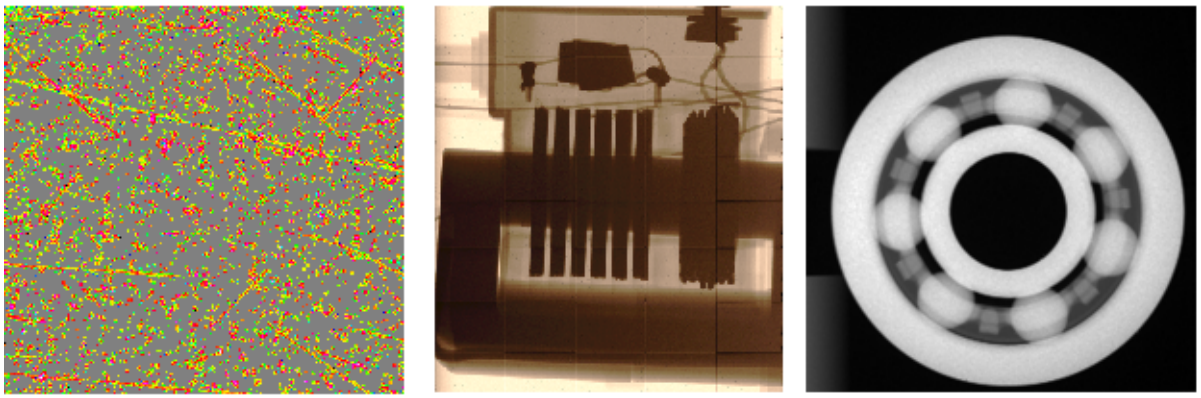
Contents

Overview

We provide SDK for our products. Currently it has binary and python API.
The SDK packages are available here: https://downloads.advacam.com/sdk.php
Each API contain 3 groups of using type:

Core/Basic measuring and settings

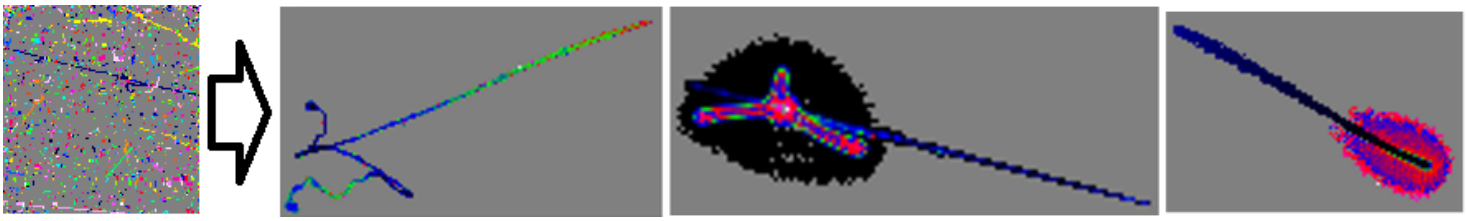
- Settings of the device
- Getting information about the device
- Single-frame measurement
- Multi-frame measurement
- Continuous frame measurement (endless repeats and without dead time on some devices, Mpx3 and Tpx2 for example)
- Data-driven (pixel mode) measurement (Tpx3 only)
- Synchronized measurement
- TDI imaging - Scanning linearly moving objects, for example on a conveyor belt.
- Basic image bad pixel correction
- Beam hardening correction (Xray imaging has a very non-linear relationship between the thickness of the material and the intensity of the transmitted radiation. First, low-energy radiation is captured, followed by components that are significantly more penetrating.)



Basic measuring example

Cluster processing

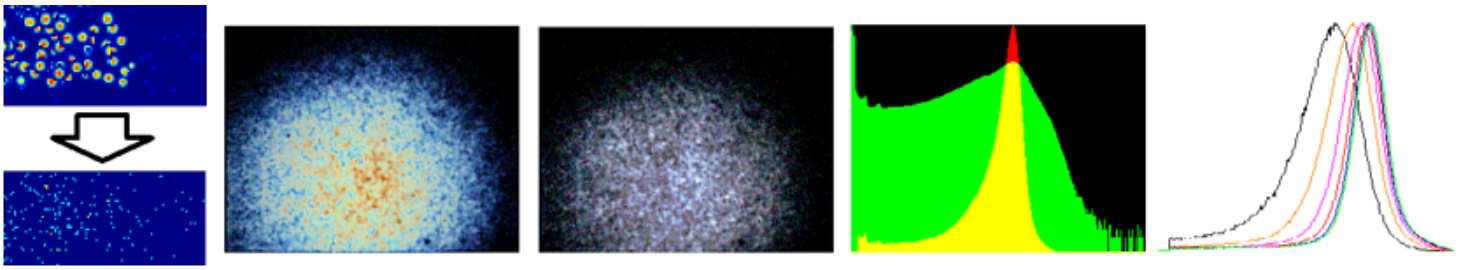
- Searching for clusters in saved data
- Measuring with online searching for clusters
- Getting informations about clusters - Time of arrival, energy, size, ...



Clustering example

Spectral imaging

- Cluster processing with convert each cluster to pixel and create images by spectrum settings
- Getting spectrum information from whole chip, selected rectangle area or from single pixels
- Generating image with subpixel resolution



Spectral Imaging examles

Binary (C/C++) APIs

This APIs contains binary libraries, DLLs for Windows and SOs for Linux.
It is intended to be easily used from C/C++, but it is also possible to use it from many other languages, e.g. C#, Visual Basic, kotlin, Xojo basic, can be used in Python via ctypes, if our Pyhon API did not suit you, ... and can be imported to the LabView.
This API was created as simplified, for ease of use, for example in industry, so it is not object-oriented.
Binary libraries are available in 32 and 64 bit versions for PCs and for ARMs and can be used on PC, RPi and some Android phones

The libraries are exportes in C style:

- Windows DLLs - extern "C" __declspec(dllexport)
- Linux SOs - extern "C" __attribute__ ((visibility("default")))

Download the AdvacamAPIexamples package
https://advacam.com/examples/AdvacamAPIexamples.rar
(MS Visual Studio 2017 Solution with C++ projects of Windows CLR programs)

The binary API has parts:

Core/basic API

With the pxcore library, allowing basic measurements and device settings.
Files:

- pxcapi.h API header file
- pxcore.dll or pxcore.so binary libraries for Windows or Linux
- pxcore.lib static linging file for easier using on Windows (compile time only)

See Binary core API

Clustering API

With the pxproc library, designed for searching and processing clusters. Processing is possible online during measurement, or offline by processing data from files.
Files:

- clusteringapi.h API header file
- pxproc.dll or pxproc.so binary libraries for Windows or Linux
- pxproc.lib static linging file for easier using on Windows (compile time only)

See Binary Clustering API

Spectral Imaging API

Also with the pxproc library, which, after cluster processing, additionally creates frames, where a single pixel is created from each cluster. And energy spectrum graphs can also be easilly generated. From the entire surface or a selected section.
Files:

- spectraimgapi.h API header file
- pxproc.dll or pxproc.so binary libraries for Windows or Linux
- pxproc.lib static linging file for easier using on Windows (compile time only)

See Binary Spectral Imaging API

Python API

The Python API was created primarily for scientific purposes, so it is quite extensive and fully object-oriented. It can be used either directly using Python installed on the computer or from a small IDE integrated in the Pixet program.
API files list:

- pypixet.pyd - Basic API file for all basic device functions like as settings, measuring and get status information.

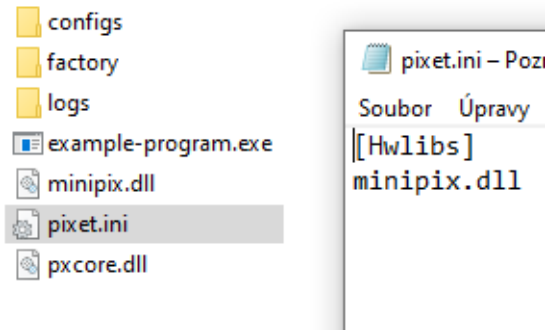
- pypxproc.pyd - Pixel processing, used by Clustering and Spectral imaging objects.
- pypixetgui.pyd - API for using the Pixet GUI. Programmer can create windows, dialogs, graphs, ... (Usable only if started from the IDE integrated in the Pixet program).

This API also using the pxcore and pxproc libraries.

See Python API

Auxilliary files

If you want using some our API, you need an auxilliary files:



Directory of minimalistic program using binary API and working on computer with Visual Studio installed

- The pixet.ini file located in the program starting directory.
- hwlib file(s) listed in the pixet.ini.
- Drivers for some devices properly installed on the computer.
- Additional files required for some devices. Typically FPGA firmwares. Located in the program starting directory.
- system libraries used by SDK libraries (pxcore.dll depends on: mfc110.dll, mfcloop.dll, msvcp120.dll, msvcp140.dll, msvcr110.dll, msvcr120.dll, vccorlib140.dll, vcruntime140.dll). This is not needed when the program is running from a development environment, or on a computer where the environment is installed. Don't forget to add the files when exporting for use on another computer.
- Device factory configuration XML file(s) for individual device. Located by pixet.ini settings, default is the "factory" subdirectory of the program starting directory.
- Location for saving the current configuration, with write enabled. Located by pixet.ini settings, default is the "configs" subdirectory of the program starting directory.
- Location for storing log files, with write enabled. Located by pixet.ini settings, default is the "logs" subdirectory of the program starting directory.

Notes:

1. Devices without proper configuration loaded can working, but it measuring strange things.
2. For simple test You can copy your executable file to the Pixet working directory. But it is better to create a separate directory (directories) for development, to avoid chaos in files.

See Files and directories: Main directory of the API-using programs