Embedded Vision Kit

The evolution of the classical machine vision towards embedded vision is rapidly evolving. These compact systems, in which the cost factor plays a major role, consume less energy by increasing performance. But developing an embedded vision device can be very time consuming and cost-intensive. The limitations of these highly specialized devices regarding data interfaces, performance, storage space and user interfaces make hardware handling and software development very difficult compared to a desktop workstation with standard components. Especially with proprietary developments (hardware platform, firmware and software) you may lose a lot of time until the first results are available.

But especially for the pre-development phase there are now a number of suitable embedded standard components that allow out-of-the-box testing. In combination with qualified software solutions, the first insights for vision applications can be derived very quickly.

Our TechTip shows in a few simple steps how to implement a simple embedded vision application with a uEye camera and a Raspberry Pi 3.

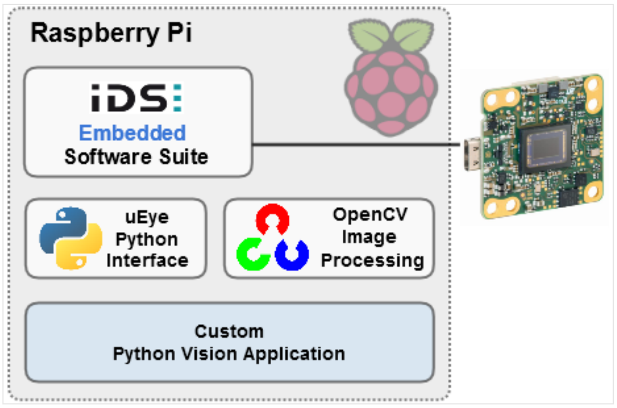
Background

For fast image processing results, we use the OpenCV (Open Computer Vision) OpenSource library. In addition to an algorithms collection, it also provides sample code for various aspects of machine vision. With the BSD license, OpenCV is free for private as well as commercial projects and is pre-installed with the Raspbian OS.

OpenCV has a Python interface for a quick start and easy development. So you will benefit from the many advantages of Python, such as the interactive application programming. This allows you to write and test short code snippets without the complex setup of a complete development environment.

With the new "PyuEye" interface you can now use all uEye cameras with the object-oriented programming language Python. In combination with the OpenCV Python wrapper, it is an easy way to develop prototypes on an embedded system including the Raspberry Pi.

Once the PyuEye interface is installed, you can import a "uEye" module into your Python application to access all uEye functions and types of the installed uEye SDK. The calling syntax of the functions and their parameters is completely oriented on the uEye manual.



Embedded Vision application with uEye Python interface and OpenCV

Approach

As hardware platform for our demo project we use a Raspberry Pi 3 with Raspbian OS version "Jessie" and a uEye USB camera.

To keep the demo project as simple as possible, we only use software components that can be obtained from the package sources of Raspbian Jessie and the Python Package Index (PyPI).

The following software components have to be additionally installed on the Raspberry Pi:

- latest embedded uEye camera driver
- new uEye Python interface "PyuEye"
- OpenCV incl. Python interface

Step 1: Prepare hardware

Set up a Raspberry Pi 3 with the Raspbian OS and update the system to the latest software version.

```
pi@raspberrypi:~ $ sudo apt-get update && apt-get upgrade
```

Instructions on setting up a Raspberry Pi can be found on the Internet. Basically, you can use any other ARMv7 compatible embedded board (e. g. Odroid XU4) for the demo. However, the Raspberry Pi3 with its quad-core CPU has enough power for simple image processing tasks and the Raspbian OS comes with many pre-installed components. Everything else can be easily installed via the package sources.

Connect a uEye USB camera to an USB port of the Raspberry Pi.

Step 2: Installing the Camera Driver and Interface

Install the latest embedded uEye camera driver. After successful driver installation you can use the uEye USB camera with the included uEye demo application.

Information about "PyuEye" can be found on the Python uEye interface website. Pyueye is hosted as OpenSource project in the Python Package Index (https://pypi.org/project/pyueye/). You can either download it as a package or install it directly via the Python package management program "PIP". All necessary requirements are pre-installed with Raspbian.

```
pi@raspberrypi:~ $ sudo pip install pyueye
```

This installs the uEye Python interface to be used with Python 2.7. Necessary module dependencies are installed automatically by PIP. To check the proper installation, use the Python interpreter and import the uEye module.

```
pi@raspberrypi:~ $ python
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
>>> from pyueye import ueye
>>>
```

If no error message is returned, the installation was successful.

Step 3: Install OpenCV

The OpenCV development libraries can be easily installed from the Raspbian package sources. Although they have an older version (2.4.9.1), they are sufficient for our demo. The Python 2.7 bindings for the OpenCV libraries can also be installed from the package sources. If you want to use Python 3, you have to compile them yourself from the code sources for the embedded platform. You can also find simple instructions online.

```
pi@raspberrypi:~ $ sudo apt-get install libopencv-dev python-opencv
```

You can also check this installation with the Python interpreter by importing the OpenCV module "cv2".

Step 4: Download and run the PyuEye sample application

Start your own image processing applications with uEye and the Python interface by downloading the source code example, which is linked within the TechTipp website and extract it into a directory on your Raspberry Pi.

PyuEye example application (7.8 KiB)

The source code example is completely made with Python. Therefore, you do not need to cross-compile it for the system architecture (ARMv7 A) of the Raspberry Pi. This makes it platform-independent and you can execute it directly. In other words, you can also run this source code example on a Windows or Linux desktop system if the requirements (uEye drivers, PyuEye interface, Python 2.7) are installed on these systems. The PyuEye source code example consists of four Python files that provide classes and functionality for different parts of the example program:

1) pyueye_example_camera.py

Provides a "camera" class with frequently used camera functions.

2) pyueye_example_gui.py

With the classes "PyuEyeQtView" and "PyuEyeQtApp" you can create a simple Qt widget GUI application. This module is based on Qt4 and uses the Python Qt4 bindings (PyGt4). Qt4 is already integrated in Raspbian Jessie. So you can install the Python bindings from the package sources:

```
pi@raspberrypi:~ $ sudo apt-get install python-qt4 python-qt4-doc
```

3) pyueye_example_utils.py

This module provides important convenience functions and classes that are very helpful when developing a camera application. From exception handling to the management of camera data and image storage, there are a number of useful options.

4) pyueye_example_main.py

The main module creates a simple Qt application framework, opens and initializes the connected camera and provides you with an image processing callback function, in which a simple image processing with OpenCV is implemented. The running demo shows a live image of the connected camera and overlays the processing results.

```
pi@raspberrypi:~/example $ python pyueye_example_main.py
```

OpenCV image processing

Our simple image processing task with OpenCV searches for circles in the image and highlights them.

A few lines of code in the main module are sufficient to do this, since OpenCV contains a complete implementation with the *cv2.HoughCircles()* function for this task. To work with OpenCV, "cv2" and "numpy" have been imported:

```
from pyueye_example_camera import Camera
from pyueye_example_utils import FrameThread
from pyueye_example_gui import PyuEyeQtApp, PyuEyeQtView
from PyQt4 import QtGui

from pyueye import ueye

import cv2
import numpy as np
```

The functional principle of *cv2.HoughCircles()* and its call parameters are explained very detailed in the OpenCV documentation. That's why we go directly to the application source code.

The image data must be available as a one-dimensional 8-bit data array. The function "*as_1d_array()*" from the ImageData class in the pyueye_example_utils.py module and the OpenCV function "*cvtColor()*" will handle this task. If *cv2. HoughCircles()* has detected circles in the image, they are highlighted with OpenCV drawing functions. The Sample Qt application is responsible for displaying the image data.

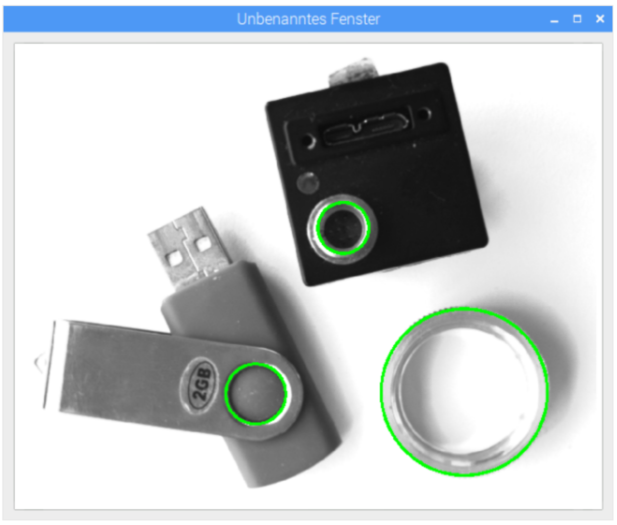
```
def process_image(self, image_data):

    # reshape the image data as 1dimensional array
    image = image_data.as_1d_image()
    # make a gray image
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    #image = cv2.medianBlur(image,5)
    # find circles in the image
    circles = cv2.HoughCircles(image, cv2.cv.CV_HOUGH_GRADIENT, 1.2, 100)
    # make a color image again to mark the circles in green
    image = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)

    if circles is not None:
           # convert the (x, y) coordinates and radius of the circles to integers
           circles = np.round(circles[0, :]).astype("int")
           # loop over the (x, y) coordinates and radius of the circles
           for (x, y, r) in circles:
```

```
                    # draw the circle in the output image, then draw a rectangle
                    # corresponding to the center of the circle
                    cv2.circle(image, (x, y), r, (0, 255, 0), 6)

      # show the image with Qt
      return QtGui.QImage(image_data,
                          image_data.mem_info.width,
                          image_data.mem_info.height,
                          QtGui.QImage.Format_RGB888)
```



OpenCV finds and highlights circles in the image data.

You can easily modify the application and test other processing tasks with OpenCV. Typical for Python, you can directly execute the modified application.

Summary

In addition to our embedded uEye camera driver, we introduce the new PyuEye 3rd party interface as another component to realize embedded vision projects quickly and easily. With the Python base you can use our powerful uEye camera SDK in a platform-independent application. Develop Python program code on a Windows desktop PC and run the application on a Raspberry Pi without thinking about setting up different development environments. In addition, Python is one of the most popular programming languages today, which results in the high availability of frameworks of almost all important areas. Web applications, user interfaces, data analysis and statistics and not to forget image processing (e. g. OpenCV). Python has a large community that has already discovered the embedded vision. The uEye Python interface gives you therefore access to a huge Embedded Vision kit.

Manual

All information at a glance - you can either view the manual online here or download it for offline use.

TO THE MANUAL

ReadMe

IDS peak 1.2.1 for Windows

README

Release Notes

These release notes describe the changes of IDS peak 1.2, which supports Python as an additional programming language besides .NET (incl. C#) and C/C++.

ZU DEN RELEASE NOTES