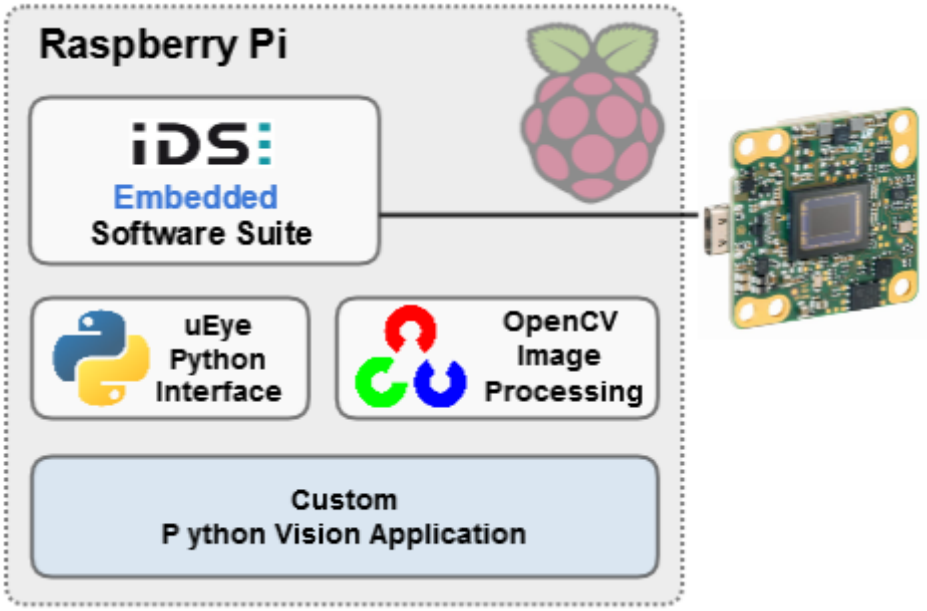
Desarrollo de prototipos con la uEye Python Interface y Open CV

La visión artificial clásica evoluciona rápidamente hacia una **visión embebida**. Los sistemas compactos, en los que el factor coste es importante, ofrecen un menor consumo de energía y al mismo tiempo unas prestaciones más elevadas. Sin embargo, desarrollar un dispositivo de visión embebida puede suponer una gran inversión de tiempo y de dinero. Las limitaciones de estos sistemas altamente especializados en cuanto a interfaces, rendimiento, memoria y posibilidades de visualización y de introducción de datos dificultan mucho el manejo del hardware y el desarrollo del software en comparación con una estación de trabajo de escritorio con componentes estándar. Y precisamente con desarrollos propios (plataforma de hardware, firmware y software) se puede tardar mucho tiempo hasta obtener los primeros resultados.

Sin embargo, en el mercado existe una serie de componentes de visión embebida estándar para la fase inicial de desarrollo que permiten realizar pruebas preliminares "**nada más salir de la caja**". En combinación con los programas de software adecuados, permiten realizar pruebas y obtener los primeros resultados muy rápidamente.

Nuestro Consejo técnico le explica cómo crear **en muy pocos pasos** una aplicación sencilla de visión embebida con una cámara uEye y una Raspberry Pi 3.



Visión embebida
Aplicación con la uEye Python Interface y Open CV

Contexto

Para obtener resultados rápidamente en el procesamiento de imágenes se puede utilizar la biblioteca OpenSource **OpenCV** (Open Computer Vision). Además de un catálogo de algoritmos, esta biblioteca también ofrece códigos de ejemplo para distintas tareas de la visión artificial. Con la licencia BSD, OpenCV es gratis para proyectos particulares y comerciales y ya está preinstalada en el SO Raspbian.

Existe una **Python Interface** para OpenCV que permite un inicio rápido y un desarrollo sencillo. De ese modo puede aprovechar las numerosas ventajas que ofrece Python, como por ejemplo la programación interactiva de una aplicación. Le permitirá escribir y probar fragmentos de código sin tener que crear complejos entornos de desarrollo.

Y con la nueva "**PyuEye**" Interface podrá utilizar todas las **cámaras uEye** con el lenguaje de programación orientado a objetos Python. En combinación con el wrapper de Phython para OpenCV es ideal para el desarrollo de prototipos en un sistema embebido como Raspberry Pi.

Si se ha instalado la PyuEye Interface podrá importar un módulo "uEye" a una aplicación Python mediante el que tendrá acceso a todas las funciones y tipos uEye del SDK uEye instalado. La sintaxis de activación de las funciones y de los parámetros funcionales se basa por completo en el manual de uEye.

Procedimiento

Como plataforma de hardware para nuestro proyecto demo utilizamos una **Raspberry Pi 3** con la versión "Jessie" del sistema operativo Raspbian y una cámara **uEye USB.**

Para simplificar al máximo el proyecto demo utilizamos solo componentes de software disponibles en los paquetes fuente de Raspbian Jessie y del Python Paket Index (PyPI).

En la Raspberry Pi se tienen que instalar además los siguientes componentes de software:

- un controlador de la cámara uEye embebida actualizado
- la nueva uEye Python Interface "PyuEye"
- OpenCV con Python Interface

Paso 1: Preparar el hardware

Instale el SO Raspbian en una Raspberry Pi 3 y actualice el sistema con la última versión de software.

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

En internet encontrará las instrucciones para configurar una Raspberry Pi. Teóricamente para la demo puede utilizar cualquier otra tarjeta embebida compatible con ARMv7 (por ejemplo Odroid XU4). Sin embargo, con su CPU Quadcore, Raspberry Pi3 posee suficiente potencia para realizar pruebas sencillas de procesamiento de imágenes y el SO Raspbian ya tiene preinstalados muchos de los componentes necesarios. Todo lo demás se puede instalar después cómodamente mediante los paquetes fuente.

Conecte una cámara USB uEye a un puerto USB de la Raspberry Pi.

Paso 2: Instalar el controlador de la cámara y la interfaz

Instale la versión actualizada del controlador de la cámara uEye embebida. Encontrará información sobre el controlador adecuado y la instalación en la página de descargas de uEye Software Suite. Una vez instalado el controlador correctamente ya podrá utilizar la cámara uEye USB con la aplicación uEye Demo.

Puede obtener información sobre la nueva uEye Python Interface en la sección de Interfaces del área de descarga de esta página web. "PyuEye" está disponible como proyecto OpenSource en el Python Package Index (PyPI) (https://pypi.org/project/pyueye/). La puede descargar desde allí como paquete

o directamente a través del programa de gestión de paquetes "PIP". Todos los requisitos necesarios ya están preinstalados con Raspbian.

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

De ese modo se instala la uEye Python Interface para el uso con Python 2.7 en el sistema. Las dependencias de módulo necesarias se instalan directamente con PIP. Para comprobar que la instalación es correcta inicie el Python-Interpreter e importe el módulo uEye.

```
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
>>>
```

Si no aparece ningún mensaje de error significa que la instalación es correcta.

Paso 3: Instalar OpenCV

Las **bibliotecas de desarrollo OpenCV** se pueden instalar muy fácilmente a partir de los paquetes fuente de Raspbian. Se trata de una versión más antigua (2.4.9.1), pero es más que suficiente para nuestra demo. La **conexión Python** para las bibliotecas OpenCV para Python 2.7 también se encuentra en los paquetes fuente. Para el uso con Python 3 debe compilarla a partir de los códigos fuente para la plataforma embebida. En este caso también encontrará instrucciones sencillas en la red.

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

Esta instalación también la puede comprobar en Python-Interpreter con la importación del módulo "cv2" de OpenCV.

Paso 4: Descargar y probar la aplicación de ejemplo PyuEye

Como punto de partida para sus propias aplicaciones de procesamiento de imágenes con uEye y la Python Interface descárguese el ejemplo de código fuente enlazado en este Consejo técnico y descomprímalo en el directorio que desee de la Raspberry Pi.

Ejemplo de aplicación PyuEye

El ejemplo de código fuente se ha generado por completo en Python. Por ello no tiene que hacer una compilación cruzada para la arquitectura del sistema (ARMv7 A) de la Raspberry Pi. Puede ejecutarlo directamente a través del Python Interpreter. De ese modo será compatible con cualquier plataforma. Esto significa que también puede ejecutar el ejemplo de código fuente en un sistema Windows o Linux Desktop siempre que en estos sistemas estén instalados los requisitos necesarios (controlador uEye, PyuEye Interface, Python 2.7).

El ejemplo de código fuente PyuEye está dividido en cuatro archivos Python que ofrecen clases y funcionalidades a distintas partes del programa de ejemplo:

1) pyueye_example_camera.py

Proporciona la clase "Camera" con funciones que se necesitan con frecuencia para trabajar con la cámara.

2) pyueye_example_gui.py

Con las clases "PyuEyeQtView" y "PyuEyeQtApp" puede generar un programa Qt Widget sencillo y con ello el marco para una aplicación GUI. Este módulo se basa en Qt4 y por consiguiente utiliza bindings de Qt4 para Python (PyQt4). Qt4 ya está integrado en Raspbian Jessie. Puede instalar los bindings para Python mediante los paquetes fuente:

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

3) pyueye_example_utils.py

Este módulo aporta importantes funciones y clases de conveniencia que son muy útiles para trabajar con una aplicación de cámara. Entre muchas otras, se encuentran el manejo de excepciones, de datos de la cámara y de las memorias gráficas.

4) pyueye_example_main.py

El módulo "main" crea una estructura de aplicación sencilla con Qt, abre e inicializa la cámara conectada y pone a su disposición una función callback en la que se ya se ha realizado un procesamiento de imágenes sencillo con OpenCV. Si ejecuta la demo visualizará la imagen en directo de la cámara conectada y los resultados del procesamiento de imágenes.

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

Procesamiento de imágenes OpenCV

Nuestro sencillo método de procesamiento de imágenes con OpenCV **busca círculos en la imagen** y los marca.

Para ello basta con unas pocas líneas de código en el módulo "main", puesto que para esta tarea OpenCV contiene una implementación completa con la función *cv2.HoughCircles* que vamos a utilizar. Y para poder trabajar con OpenCV se han importado "cv2" y "numpy":

```
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
```

El funcionamiento de la función*cv2.HoughCircles()* y de sus parámetros de activación están muy bien explicados en la documentación de OpenCV. Por ello pasaremos directamente al código fuente de la aplicación.

Los datos de imagen deben estar disponibles como array de datos unidimensional de 8 bits. La función "*as_1d_array()*" de la clase ImageData en pyueye_example_utils.py-Modul y la función OpenCV "*cvtColor()"* se encargan de realizar esta tarea. Cuando la función cv2.HoughCircles() encuentra círculos en la imagen, estos se marcan con las funciones de dibujo de OpenCV. De la visualización de los datos de imagen marcados se encarga de nuevo la aplicación de muestra Qt.

```
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)
```

Como es habitual en Python, puede ejecutar directamente la aplicación modificada. Puede seguir modificando la aplicación sin demasiado esfuerzo y probar otras tareas de procesamiento con OpenCV.

OpenCV encuentra y marca círculos en los datos de imagen.

Conclusión

Con la nueva PyuEye 3rd Party Interface, además de nuestro controlador de la cámara uEye embebida, le ofrecemos otra solución para realizar proyectos de visión embebida de forma rápida y sencilla. Utilizando Python como base, puede usar nuestro potente SDK de la cámara uEye en una aplicación compatible con cualquier plataforma. Desarrolle códigos de programa Python en un PC de sobremesa Windows y ejecute la aplicación en una Raspberry Pi sin preocuparse sobre la configuración de los distintos entornos de desarrollo. Python es uno de los lenguajes de programación más conocidos actualmente, algo evidente en vista de la gran disponibilidad de frameworks en prácticamente todos los ámbitos importantes. Aplicaciones web, interfaces de usuario, análisis de datos y estadísticas y, claro está, el procesamiento de imágenes (por ejemplo OpenCV). Python conecta a una gran comunidad que ya ha descubierto la visión embebida. Por consiguiente, la uEye Python Interface le permite acceder a un gran módulo de visión embebida.

Puede obtener más información sobre la uEye Python Interface en el siguiente enlace de nuestra página web:
https://es.ids-imaging.com/ueye-interface-python.html

Autor:
Heiko Seitz. Redacción técnica

Contacto:
IDS Imaging Development Systems GmbH
Dimbacher Straße 6-8
74182 Obersulm
Alemania

Tel.: +49 7134 96196-0
E-mail: marketing@ids-imaging.de
Sitio web: www.ids-imaging.de

Más consejos técnicos y campos de aplicación en nuestra página web.