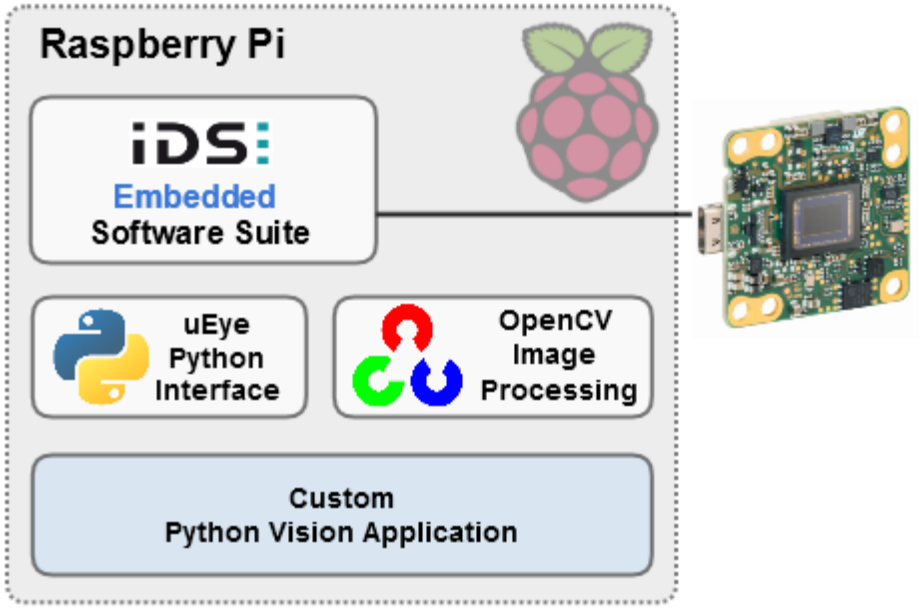
Développement de prototypes avec l'interface PyuEye et OpenCV

La vision industrielle classique continue d'évoluer très rapidement vers la **vision embarquée**. Les systèmes compacts, pour lesquels le facteur coût joue un rôle essentiel, consomment moins d'énergie et leurs capacités ne cessent d'augmenter. Cependant, le développement d'un appareil de vision embarquée peut être très gourmand en temps et en argent. Les limites de ces appareils hautement spécialisés résident cependant dans leurs interfaces, leurs performances, l'espace mémoire ainsi que leurs options d'affichage et de saisie, qui compliquent nettement la manipulation matérielle et le développement logiciel par comparaison à une station de travail de bureau dotée de composants standard. Rien que dans le cadre de développements internes (plateforme matérielle, firmware et logiciel), il faut consacrer beaucoup de temps, entre autres choses, avant de pouvoir constater les premiers résultats.

Pourtant, dès la phase de pré-développement, il existe aujourd'hui une série de composants standard intégrés et adaptés qui permet de réaliser **immédiatement** les premiers tests. Ces composants associés à des solutions logicielles adéquates permettent d'arriver très rapidement aux premiers résultats de la future application de vision.

Notre astuce technique montre **en quelques étapes claires** comment réaliser une application de vision embarquée simple avec une caméra uEye et un Raspberry Pi 3.



Application de vision embarquée avec interface uEye Python et OpenCV

Contexte

L'utilisation de la bibliothèque OpenSource **OpenCV** (Open Computer Vision) permet d'obtenir rapidement des résultats en matière de traitement d'image. Outre une collection d'algorithmes, cette bibliothèque met également à disposition des exemples de code pour différentes tâches de la vision par ordinateur. Avec la licence BSD, la bibliothèque OpenCV est gratuite aussi bien pour les projets privés que commerciaux et est pré-installée sur le système d'exploitation Raspbian.

Pour un démarrage rapide et un développement simple, OpenCV dispose d'une **interface Python**. Profitez des nombreux avantages de Python, comme la programmation interactive d'une application. Vous pouvez ainsi écrire et tester de courts extraits de code sans perdre de temps à créer un environnement de développement complet.

Avec la nouvelle interface « **PyuEye »**, vous pouvez désormais utiliser toutes les **caméras uEye** avec le langage de programmation Python orienté objet. L'association avec Python Wrapper pour OpenCV est idéale pour développer des prototypes sur un système intégré comme le Raspberry Pi.

Si l'interface PyuEye est installée, vous pouvez importer un module « uEye » dans une application Python, qui vous donnera accès à toutes les fonctions uEye et à tous les types de kits de développement logiciel uEye installés. La syntaxe d'appel des fonctions et des paramètres de fonction suit complètement le Manuel uEye.

Procédure

Pour notre projet de démonstration, nous utilisons comme plateforme matérielle un **Raspberry Pi 3** avec système d'exploitation Raspbian, version « Jessie »,et une **caméra USB uEye.**

Pour que le projet de démonstration reste le plus simple possible, nous utilisons uniquement des composants logiciels fournis via les paquets source Raspbian Jessie et le Python Package Index (PyPI).

Les composants logiciels suivants doivent être installés sur le Raspberry Pi :

- un pilote de caméra uEye intégrée actuel,
- la nouvelle interface uEye Python « PyuEye »,
- OpenCV avec l'interface Python.

Étape 1 : préparation du matériel

Installez un Raspberry Pi 3 avec le système d'exploitation Raspbian et actualisez le système avec la dernière version logicielle.

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

Vous trouverez sur Internet des instructions pour installer un Raspberry Pi. Théoriquement, vous pouvez utiliser pour la démonstration n'importe quelle autre carte intégrée compatible ARMv7 (p. ex. Odroid XU4). Cependant, le Raspberry Pi3, avec son processeur quadricœur, a suffisamment de puissance pour réaliser des tests de traitement d'image simples, et le système d'exploitation Raspbian n'est pas en reste avec les composants requis pré-installés. L'ensemble se réinstalle très aisément avec les paquets source.

Raccordez une caméra USB uEye à un port USB du Raspberry Pi.

Étape 2 : installation du pilote de la caméra et de l'interface

Installez le pilote actuel de la caméra uEye intégrée. Vous trouverez également des informations sur le choix du pilote et l'installation sur la page de téléchargement de la suite logicielle uEye. Une fois le pilote installé, vous pouvez utiliser la caméra USB uEye avec l'application de démonstration uEye jointe.

Dans la rubrique Interface de la page Web d'IDS, vous trouverez des informations sur la nouvelle interface uEye Python. L'interface « PyuEye » est hébergée en tant que projet OpenSource dans le Python Package Index (PyPI) (https://pypi.org/project/pyueye/). À cet emplacement, vous pouvez soit

télécharger le paquet, soit procéder directement à l'installation via le gestionnaire de paquets Python « PIP ». Toutes les configurations requises sont pré-installées avec Raspbian.

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

L'interface uEye Python est alors installée sur le système et peut être utilisée avec Python 2.7. Les dépendances de module nécessaires sont directement installées en même temps par le PIP. Pour vérifier si l'installation est correcte, démarrez l'interpréteur Python et importez le module uEye.

```
pi@raspberrypi:~ $ python
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Tapez "help", "copyright", "credits" ou "license" pour plus d'informations.
>>> from pyueye import ueye
>>>
```

Si aucun message d'erreur ne s'affiche, l'installation a réussi.

Étape 3 : installation d'OpenCV

Les **bibliothèques de développement OpenCV** s'installent très simplement à partir des paquets source Raspbian. Il s'agit en fait d'une version antérieure (2.4.9.1), mais qui suffit amplement pour notre démonstration. La **liaison Python** des bibliothèques OpenCV pour Python 2.7 se trouve également dans les paquets source. Pour une utilisation avec Python 3, vous devez les compiler vous-même à partir du code source pour la plateforme intégrée. Vous trouverez aussi sur Internet des instructions simples à ce sujet.

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

Vous pouvez également contrôler cette installation dans l'interpréteur Python en important le module OpenCV « cv2 ».

Étape 4 : téléchargement d'un exemple d'application PyuEye et test

Comme point de départ de vos propres applications de traitement d'image avec uEye et l'interface Python, téléchargez l'exemple de code source sous forme de lien sur la page Web Astuce technique et dézippez-le dans un répertoire quelconque de votre Raspberry Pi.

Exemple d'application PyuEye

L'exemple de code source a été entièrement créé dans Python. Par conséquent, vous n'avez pas besoin d'effectuer de compilation croisée pour l'architecture système (ARMv7 A) du Raspberry Pi. Vous pouvez l'exécuter directement grâce à l'interpréteur Python. Il ne dépend pas de la plateforme. Cela signifie que vous pouvez aussi exécuter cet exemple de code source sur un système informatique Windows ou Linux si les configurations préalables requises (pilote uEye, interface PyuEye, Python 2.7) sont présentes sur ce système.

L'exemple de code source PyuEye est réparti dans quatre fichiers Python, qui fournissent les classes et fonctionnalités aux différentes parties du programme de démonstration :

1) pyueye_example_camera.py

Met à disposition la classe « Camera » avec les fonctions fréquemment utilisées pour manipuler la caméra.

2) pyueye_example_gui.py

Avec les classes « PyuEyeQtView » et « PyuEyeQtApp », vous pouvez créer un widget Qt simple, et donc, le cadre d'une application GUI. Ce module repose sur Qt4 et utilise en conséquence les bindings Qt4 Python (PyGt4). Qt4 est déjà intégré à la version Jessie de Raspbian. Vous pouvez installer les bindings Python via les paquets source :

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

3) pyueye_example_utils.py

Ce module fournit des fonctions et des classes Convenience importantes, qui sont très utiles pour traiter une application de caméra. Tout est là : de la gestion des exceptions (Exception-Handling) au traitement des données de caméra et des mémoires d'images.

4) pyueye_example_main.py

Le module principal crée une structure d'application Qt simple, active et initialise la caméra raccordée et met à votre disposition une fonction de rappel de traitement d'image (Image Processing Callback), qui permet de traiter aisément les images avec OpenCV. Si vous exécutez la démonstration, vous verrez l'image en direct de la caméra raccordée et les résultats du traitement des images.

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

Traitement des images OpenCV

Notre traitement simple des images avec OpenCV **recherche des cercles dans l'image** et les repère.

Quelques lignes de code dans le module principal suffisent, car OpenCV comporte pour cette tâche une implémentation complète avec la fonction *cv2.HoughCircles*, que nous utilisons à cette fin. Pour pouvoir travailler avec OpenCV, « cv2 » et « numpy » ont été importés :

```
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
```

Le principe de la fonction *cv2.HoughCircles()* et ses paramètres de requête sont très bien expliqués dans la documentation OpenCV. Pour cette raison, nous allons passer directement au code source de l'application.

Les données graphiques doivent être représentées sous la forme d'un tableau de données 8 bits unidimensionnel. La fonction « *as_1d_array()* » de la classe ImageData dans le module pyueye_example_utils.py et la fonction OpenCV « *cvtColor()* » remplissent cette mission. Lorsque la fonction cv2.HoughCircles() trouve des cercles dans l'image, ils sont repérés dans l'image avec les fonctions de dessin OpenCV. L'échantillon de l'application Qt affiche de nouveau les données graphiques repérées.

```
def process_image(self, image_data):

    # remodèle les données graphiques en tableau unidimensionnel
    image = image_data.as_1d_image()
    # réalise une image grise
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    #image = cv2.medianBlur(image,5)
    # recherche des cercles dans l'image
    circles = cv2.HoughCircles(image, cv2.cv.CV_HOUGH_GRADIENT, 1.2, 100)
    # réalise une nouvelle image en couleur pour repérer les cercles en vert
    image = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)

    if circles is not None:
            # convertit les coordonnées (x, y) et le rayon des cercles en entiers
            circles = np.round(circles[0, :]).astype("int")
            # insère une boucle sur les coordonnées (x, y) et le rayon des
cercles
            for (x, y, r) in circles:
                    # dessine le cercle sur l'image de sortie, puis trace un rec-
tangle
                    # correspondant au centre du cercle
                    cv2.circle(image, (x, y), r, (0, 255, 0), 6)

    # affiche l'image avec Qt
    return QtGui.QImage(image.data,
                        image_data.mem_info.width,
                        image_data.mem_info.height,
                        QtGui.QImage.Format_RGB888)
```

Typiquement avec Python, vous pouvez exécuter directement l'application modifiée. Vous pouvez apporter d'autres modifications à l'application sans trop d'efforts et tester d'autres tâches de traitement avec OpenCV.

OpenCV trouve et repère des cercles dans les données graphiques.

Résumé

Avec la nouvelle interface tierce PyuEye, parallèlement à notre pilote de caméra uEye embarquée, nous proposons maintenant un autre module pour réaliser rapidement et simplement des projets de vision embarquée. En nous appuyant sur Python, nous utilisons notre habituel et performant kit de développement logiciel pour caméra uEye dans une application indépendante de la plateforme. Développez du code de programmation Python sur un PC de bureau Windows et exécutez l'application sur un Raspberry Pi sans avoir à réfléchir à l'installation de différents environnements de développement. De plus, Python figure aujourd'hui parmi les langages de programmation les plus connus, ce qui, dans la profusion de structures, le distingue dans presque tous les domaines importants : applications Web, interfaces utilisateur, analyse de données et statistiques, sans oublier le traitement des images (p. ex. OpenCV). Python met en relation une communauté importante qui a déjà découvert la vision embarquée. Avec l'interface uEye Python, vous avez donc désormais accès à un module conséquent de vision embarquée.

Vous trouverez d'autres informations sur l'interface uEye Python sur notre page Web, à l'adresse : https://fr.ids-imaging.com/ueye-interface-python.html

Auteur :
Heiko Seitz, rédacteur technique

Contact :
IDS Imaging Development Systems GmbH
Dimbacher Straße 6-8
74182 Obersulm
Allemagne

Tél. : +49 7134 96196-0
E-mail : marketing@ids-imaging.de
Web : www.ids-imaging.com

D'autres astuces techniques et rapports d'application sont disponibles sur le site Web.