Acquisition et affichage temps réel d'images RGB24 depuis une caméra GenICam

4 février 2025

Table des matières

1 Introduction

Ce document présente une description détaillée d'un script Python qui réalise les opérations suivantes :

- Connexion automatique à un appareil GenICam.
- Configuration du flux vidéo (dimensions, format de pixel).
- Pré-calcul d'une Look-Up Table (LUT) pour appliquer une correction gamma sur des valeurs d'intensité codées sur 24 bits.
- Conversion de données brutes au format **RGB24** (24 bits par canal, soit 72 bits par pixel) en une image **RGB8** (8 bits par canal, soit 24 bits par pixel) avec correction gamma.
- Affichage en temps réel du flux et mesure des performances (temps de conversion, FPS).
- Sauvegarde d'une ou plusieurs images sous format JPEG.

2 Bibliothèques Utilisées

— **time**
Fournit des fonctions pour gérer le temps et mesurer les durées (calcul des FPS, temporisation lors des essais de connexion, mesures de performances).
— **numpy**
Utilisée pour la manipulation de tableaux multidimensionnels et les opérations arithmétiques sur les données (conversion, redimensionnement et application de la LUT de correction gamma).
— **cv2 (OpenCV)**
Bibliothèque de traitement d'images et d'affichage. Elle sert à convertir les espaces colorimétriques (RGB vers BGR pour OpenCV), afficher l'image et sauvegarder certaines trames en JPEG.
— **ctypes**
Permet l'accès à la mémoire bas niveau afin de convertir le buffer provenant du dispositif en un tableau NumPy exploitable.
— **arena_api.system**
Fournit une interface pour détecter, connecter et configurer les appareils de capture Arena.

3 Description des Fonctions

3.1 `create_device_automatically()`

Objectif : Se connecter automatiquement au premier appareil Arena détecté.

Fonctionnement :
— La fonction effectue jusqu'à 6 tentatives pour détecter un appareil en appelant `system.create_device()`.
— Si aucun appareil n'est trouvé, elle attend 10 secondes (affichant un compte à rebours) avant d'effectuer une nouvelle tentative.
— Dès qu'un appareil est détecté, le premier de la liste est sélectionné et retourné.
— En l'absence d'appareil après 6 tentatives, une exception est levée pour informer l'utilisateur.

3.2 `setup(device)`

Objectif : Configurer les paramètres du flux de capture.

Fonctionnement :
— Récupération de la *nodemap* du dispositif pour accéder aux paramètres de configuration.

— Les paramètres `OffsetX` et `OffsetY` sont mis à zéro.
— La largeur (`Width`) et la hauteur (`Height`) sont définies (ici, 2880 et 1860 respectivement).
— Le format des pixels est configuré sur `RGB24`. Dans ce cas particulier, le flux brut est codé avec 24 bits par canal (soit 72 bits par pixel), d'où le calcul de la taille attendue du buffer par `width * height * 9` (9 octets par pixel).

3.3 `precalculate_gamma_lut_24bit(gamma, max_24bit_value=16777215, output_max_value=255)`

Objectif : Pré-calculer une Look-Up Table (LUT) permettant d'appliquer une correction gamma sur des valeurs d'intensité codées sur 24 bits.

Fonctionnement :
— Pour chaque valeur possible (de 0 à `max_24bit_value`, soit 16 777 215), la fonction calcule la valeur corrigée selon la formule :

$$\text{valeur corrigée} = \left(\frac{i}{\texttt{max\_24bit\_value}}\right)^{\frac{1}{\gamma}} \times \texttt{output\_max\_value}$$

— La LUT ainsi obtenue est un tableau NumPy de type `uint8` qui mappe chaque intensité codée sur 24 bits à une intensité 8 bits corrigée.
— Dans le code, la LUT est calculée une seule fois avec $\gamma = 4$.

3.4 `convert_to_image_data(buffer, width, height, expected_size, gamma_lut)`

Objectif : Convertir les données brutes (au format **RGB24**, c'est-à-dire 24 bits par canal, soit 72 bits par pixel) en une image **RGB8** (8 bits par canal, soit 24 bits par pixel) en appliquant la correction gamma.

Fonctionnement :
— **Accès au buffer :**
Le buffer brut est converti en tableau NumPy via `ctypes`. On crée un tableau de `expected_size` octets et on le transforme ensuite en un tableau NumPy.
— **Reshape et extraction des canaux :**
Le tableau est réorganisé en une matrice à 2 dimensions de taille $(-1, 9)$ (chaque ligne correspondant à un pixel représenté par 9 octets). Les 9 octets sont regroupés par canaux de la manière suivante :
  — Les 3 premiers octets (indices 0, 1, 2) représentent le canal rouge.
  — Les 3 octets suivants (indices 3, 4, 5) représentent le canal vert.
  — Les 3 derniers octets (indices 6, 7, 8) représentent le canal bleu.

Pour chaque canal, on recompose une valeur 24 bits en effectuant des décalages (bit-shift) et des opérations `OR`.

— **Application de la LUT gamma :**
Pour chaque canal (R, G, B), la LUT pré-calculée est utilisée pour transformer la valeur 24 bits en une valeur 8 bits corrigée.

— **Recomposition de l'image :**
Les trois canaux corrigés sont empilés pour former une image finale au format **RGB8** de dimensions `(height, width, 3)`.

— **Mesure du temps :**
Le temps total de conversion est mesuré et affiché.

3.5 `example_entry_point()`

Objectif : Démontrer le flux complet de capture et de conversion des données en image, l'affichage en temps réel, la mesure des performances et la sauvegarde de quelques trames.

Fonctionnement :

— **Connexion et configuration :**
La fonction se connecte automatiquement à un appareil via `create_device_automatically()` et configure les paramètres du flux avec `setup()`.

— **Démarrage du flux :**
Dans un bloc `with device.start_stream():`, le script entre dans une boucle de capture continue qui se termine lorsque l'utilisateur appuie sur la touche `Échap`.

— **Traitement de chaque trame :**
Pour chaque itération :
- Le buffer est récupéré et le temps nécessaire à cette opération est mesuré.
- Le buffer est converti en image via `convert_to_image_data()`, et le temps de conversion est affiché.
- L'image obtenue (au format RGB) est convertie en format BGR (pour être compatible avec OpenCV) et affichée dans une fenêtre.
- Les deux premières trames sont sauvegardées au format JPEG.
- Le buffer est réassigné via `device.requeue_buffer(buffer)`.
- Le script calcule et affiche les FPS instantanés ainsi que le temps consacré à chaque étape (obtention du buffer, conversion et réassignation).
- Une moyenne des FPS est calculée toutes les 5 trames.

— **Nettoyage :**
À la sortie de la boucle (lorsque l'utilisateur appuie sur Échap), le flux est arrêté, la fenêtre OpenCV est fermée et tous les appareils sont détruits via `system.destroy_device()`.