- Rapport de Projet: LegoTracker
  - Méthode Mise en Œuvre
  - Explication du Code
  - Fichier main.cpp
  - Fichier image_processing.cpp
  - Fichier lego_detection.cpp
  - Fichier utils.cpp
  - Fichier auto.py
  - Fichier xmltoDict.py
- Présentation et Commentaires des Résultats Obtenus
- Conclusion
- Annexes
  - DataSets

Rapport de Projet: LegoTracker

Ce projet a été réaliser par **Nathan MOREL**, **Alexandre BÉRAUD** et **Tom BEAUPUIS**

Le projet LegoTracker a pour objectif de reconnaître et de compter le nombre de pièces de Lego dans une image. Le système doit être capable de traiter des images en utilisant des techniques de traitement d'image pour identifier et compter les Legos. Les principales exigences du projet sont les suivantes :

1. **Reconnaissance des couleurs** : Utiliser la décomposition HSV pour détecter les couleurs principales des Legos.
2. **Détection des contours** : Utiliser l'algorithme de Canny pour détecter les contours des Legos.
3. **Segmentation de région** : Utiliser la méthode de croissance de région pour segmenter l'image en différentes régions.
4. **Affichage des résultats** : Afficher les résultats de la détection et du nombre de Legos détectés.

Méthode Mise en Œuvre

Pour répondre aux exigences du projet, nous avons mis en œuvre les étapes suivantes :

1. **Prétraitement de l'image** : Conversion de l'image en niveaux de gris, application d'un filtre gaussien pour réduire le bruit et binarisation de l'image.
2. **Segmentation de région** : Utilisation de la méthode de croissance de région pour segmenter l'image en différentes régions.
3. **Détection des contours** : Utilisation de l'algorithme de Canny pour détecter les contours des objets dans l'image.
4. **Identification des Legos** : Filtrage des petites régions et approximation polygonale des contours pour identifier les Legos.
5. **Affichage des résultats** : Affichage des résultats de la détection et du nombre de Legos détectés.

Explication du Code

Fichier main.cpp

Le fichier main.cpp contient le point d'entrée principal du programme. Il lit les arguments de seuil et de seuil de l'utilisateur, charge l'image, détecte les Legos et affiche les résultats.

Fichier image_processing.cpp

Le fichier image_processing.cpp contient les fonctions de traitement d'image, y compris la conversion en niveaux de gris, l'application d'un filtre gaussien et la binarisation de l'image.

Fichier lego_detection.cpp

Le fichier lego_detection.cpp contient la fonction `detectLegos` qui détecte et compte les Legos dans une image en utilisant la segmentation de région et la détection des contours.

Fichier utils.cpp

Le fichier utils.cpp contient des fonctions utilitaires pour charger, afficher et sauvegarder des images.

Fichier auto.py

Le fichier auto.py exécute le programme avec différentes valeurs de seuil et de seuil, et enregistre les résultats dans un fichier JSON.

Il sert egalement à automatiser la comparaison entre les résultats obtenus et les résultats attendus (contenu du data)

Fichier xmltoDict.py

Le fichier xmltoDict.py convertit les fichiers XML dans le dossier results en fichiers JSON et les enregistre dans le même dossier. Nous avions pris une base de donnée mais malheuresement les données etaient en XML et nous avons donc du les convertir en JSON pour pouvoir les comparer avec nos résultats.

Présentation et Commentaires des Résultats Obtenus

Les résultats obtenus montrent que le système est capable de détecter et de compter les Legos dans une image avec une précision raisonnable. Les résultats sont affichés sous forme de rectangles autour des Legos détectés, et le nombre total de Legos est affiché dans la console.

Conclusion

Le projet LegoTracker a atteint ses objectifs en fournissant un système capable de reconnaître et de compter les Legos dans une image. Les techniques de traitement d'image utilisées ont permis d'obtenir des résultats satisfaisants. Des améliorations futures pourraient inclure l'intégration de modèles d'apprentissage automatique pour améliorer la précision et l'adaptation du système pour la détection en temps réel via une webcam.

Annexes

- MyGithubDetection
- Computer Vision: Lego Blob Detection using OpenCV [Python]
- Github, Ur5 Lego Vision
- Github, Lego Detection
- LegoCraftAI: Lego Detection and Shape Advisor
- Open CV installation
- How to use open CV
- Apprendre les bases du traitement d'image
- Procédure pas à pas : création d'un réseau de traitement d'image

DataSets

- DataSet LegoBricks OnebyOne
- DataSet LegoBricks OnebyOne 3D
- DataSet LegoBricks OnebyOne reel
- DataSet LegoBricks Vrack reel