université de BORDEAUX

Rapport de Projet technologique

Etudiants :
Marc CERUTTI

6 février 2019

Table des matières

Première partie

Présentation du sujet

1 Déroulement

L'objectif de ce projet de L3 informatique est à terme de créer un robot suiveur qui suivrai son utilisateur à distance de 2 mètres.

On passera cependant par plusieurs étapes.

D'abord la création de nos fonctions d'analyse et une interface graphique à partir de la bibliothèque OpenCv et Qt respectivement. OpenCv implémente notamment des algorithmes pour faire les filtres, les cartes de disparité et les cartes de profondeur nécessaires à l'analyse d'images pour l'intelligence artificielle, et Qt un système d'interface complet pour pouvoir créer une interface permettant de tester nos fonctions d'analyse d'images.

Ensuite la création sous Unity d'une simulation avec les différents paramètres permettant de créer les comportements par défaut du robot et testant les différents algorithmes. Elle devra prévoir les spécificités du matériel et les tests pour différents environnements et situations.

Ces deux étapes se feront respectivement au semestre 5 et 6 de Licence informatique 2018-2019.

2 Analyse de l'existant

Les consignes sont de s'appuyer sur la bibliothèque d'OpenCv 3.2.0 et Qt supérieur à 4.8 afin d'assurer une compatibilité avec les ordinateurs du Cremi. Du fait de la communauté de OpenCv et de Qt, de nombreux tutoriels sont disponibles en ligne afin de nous permettre d'assurer les fonctionnalités demandé, à savoir le traitement et l'analyse d'image et la création d'une interface pour tester le résultat.

Nous avons aussi comme matériel un robot avec une caméra stéréoscopique mis à la disposition des élèves avec bien sur des contraintes de disponibilité.

3 Besoins

Comme notre programme de traitement d'image sera utilisé dans un système embarqué, la performance de notre code est critique, ainsi que la mémoire utilisé.
Il faudra aussi pour la portabilité sur le robot exactement savoir les parties de bibliothèque OpenCv à importer.

Pour analyser l'environnement du Robot, il faut assurer la gestion de ces caméra stéréoscopique et le traitement des images qui seront transmis à l'unité de contrôle.

Il faut aussi une interface qui permettra de tester, quantifier les performances, et s'assurer qualitativement du bon déroulement de l'implémentation des fonctionnalités.
Cela amène en plus à des fonctions de conversions d'images pour passer du format de OpenCv à Qt.

Deuxième partie

Architecture

4 Vue Global et conventions

La convention de nommage adopté est la suivante :

Attribut	*_name_complete*
Variables locales	*name_complete*
Méthodes	*nameComplete*
Classes	*NameClass*
Fichiers	*nameclass.**

La hiérarchie des fichiers est en annexe.
Le dossier "tools" rassemble les classes concernant le débuguage, la conversion, et le traitement d'images. Le dossier "subWindows" contient les classes de sous fenêtres à la fenêtre principal de l'interface.

5 Interface Qt

6 Bibliothèque d'analyse

6.1 Filtre d'images

6.2 Carte de disparité

6.3 Calibration Caméra

6.4 Carte de profondeur

Troisième partie
Bilan

7 Semestre 5

7.1 Difficultés rencontrés

Pas assez de bras

7.2 Critique et améliorations potentielles

Améliorer l'interface et les tests.

Quatrième partie

Annexes

Table des figures

A Hiérarchie du projet

```
l3-rvc
.
|-- 3DRV.pro
|-- cmdHierachie
|-- main.cpp
|-- maintests.cpp
|-- mainwindow.cpp
|-- mainwindow.h
|-- mainwindow.ui
|-- PRIORITES.txt
|-- Rapport
|   |-- Brouillons
|   |   |-- 14_12_2018.txt
|   |   |-- Calibration.txt
|   |   |-- Procédure Matériel
|   |   |-- rapport_05_11_2018.txt
|   |   `-- rapport_05_12_2018.txt
|   |-- hierarchie.txt
|   |-- img
|   |   `-- ubx-logo.png
|   |-- rapport.pdf
|   `-- rapport.tex
|-- README.md
|-- subWindows
|   |-- bmparamdialog.cpp
|   |-- bmparamdialog.h
|   |-- bmparamdialog.ui
|   |-- cameraparamdialog.cpp
|   |-- cameraparamdialog.h
|   |-- cameraparamdialog.ui
|   |-- sgbmparamdialog.cpp
|   |-- sgbmparamdialog.h
|   `-- sgbmparamdialog.ui
`-- tools
    |-- cameracalibration.cpp
    |-- cameracalibration.h
    |-- cvqtinterface.cpp
    |-- cvqtinterface.h
    |-- imagefilter.cpp
    |-- imagefilter.h
    |-- projectfiles.cpp
    |-- projectfiles.h
    |-- projectutilities.cpp
```

```
    |-- projectutilities.h
    |-- stereoanalyser.cpp
    |-- stereoanalyser.h
    |-- videoanalyser.cpp
    `-- videoanalyser.h

5 directories, 41 files
```

B Rapports

B.1 Rapport Initial

Scission du groupe initial du fait de problèmes internes concernant des choix d'implémentation de l'interface graphique.

Fonctionnalités de base déjà présentes à ce moment :
-Fonctions de conversion basique d'images de OpenCv vers Qt.
-Mise en place d'une bibliothèque graphique d'analyse adaptant les fonctions d'OpenCv.
-Ouverture de dialogues particulier pour les cartes de disparité.
-Filtres de flou Gaussien, de Laplacien, et de séparation d'image pour les cartes de disparité implémenté.
-Fonction de débuguage d'image pour l'affichage OpenCv crée.

B.2 Rapport 05 novembre 2018

Rapport du groupe CERUTTI
Le 05 novembre 2018

Interface graphique :
-Rajout d'une check box pour remettre à l'image d'origine automatiquement dans la fenêtre principal.
-Rajout de l'affichage de l'approximation de l'efficacité des fonctions.
-Mise a jour du code des boutons, des menus, et des fenêtres pour implémenter ces nouvelles mécaniques.
-Rajout d'une fenêtre pour les paramètres de StereoBM.
-Rajout des boutons de Sobel et Flou gaussien.

Analyse d'images :
-Amélioration des algorithmes pour prendre en charge les images en GrayScale, permet d'enlever les conversions redondantes, et l'utilisation des fonctions de manière successives facilité.
-Rajout de showMatrice(cv : :Mat mat) dans ImageAnalyser pour le débuguage.
-Rajout de computeEfficiency(double time, func, args) pour avoir une approximation de l'efficacité des algorithmes.
-Rajout des fonctions Sobel et Flou gaussien.

Mise en place d'un exécutable pour les tests d'analyse d'images.

Pistes d'améliorations :
-Implémentation des tests
-Recherche des meilleurs paramètres pour algorithmes stereo
-Factorisation du code pour les fenêtres BM et SGBM

-Recherche sur la possible utilisation de threads
-SteroVar non essayé openCv
-Recherche analyse de flux vidéo
-Recherche object tracking et template

Remarques professeur :
Pas d'image dans Git
Signaler fuites mémoire, même bibliothèque
Sobel demandé n'est pas juste l'utilisation la fonction de OpenCv, l'objectif était de faire un gradient
Voire problème de carte de disparité
Trop de découpage de code
Laisser le code cvtColor dans les fonctions nécessaires sinon cela ne facilite pas la lecture de code
Mettre des références à la place de la recopie de matrice

Prochain rapport :
Simplifier le code
Renommer les variables avec convention (code et fichiers) et commenter le code
Voire carte disparité problème
Prévoir quoi faire avec le matériel
Calibration

B.3 Rapport 5 Décembre 2018

Rapport du groupe CERUTTI Le 05 Décembre 2018
Correction depuis dernier rendu :
-Adoption d'une convention de nommage :

Attribut	*_name_complete*
Variables locales	*name_complete*
Méthodes	*nameComplete*
Classes	*NameClass*
Fichiers	*nameclass.**

-Changement de la fonction Sobel en Gradient, et implémentation en vrai Gradient.
-Correction des cartes de disparité inversé
-Amélioration des performances en enlevant la recopie des return et en passant des références en paramètre.

Calibration :
-Implémentation de findOneCalibration. Pour une image particulier il trouve les paramètres de calibration, et les mets dans un fichier. Il envoie ensuite dans la

matrice de sortie l'image non distordue.
-Test valgrind, 32 erreurs externes (supposé lié à la bibliothèque Qt)

Interface graphique :
-Rajout d'une option TestCameraCalibrate, dans les menus prend l'image courant d'interface et applique findOneCalibration.
Remarque, certaines images sont distordus comme la set1/10_20_43_159.jpg après calibration. Possibilité que le ChessBoard soit trop loin, couplé au fait qu'il n'y a qu'une seule valuation de calibration.

Pistes d'améliorations :
-Filtre pour les cartes de disparité (si cela n'impacte pas les performances)
-Créer un système de sauvegarde général pour calibration et performances.
-Améliorer le système de calibration par multiples valeurs et prise en charge automatique de calibration par dossier d'images.
-Améliorer l'interface pour prendre en compte la calibration de différentes caméras.

Remarques professeur :
-Sobel erreur
-Synthèses à améliorer
-enlever qtdesigner.txt
-mettre algos
-résultats
-problème
-pas de code dans rapport
-Erreur récurentes (?)
-Rigueur sur le code, moins sur la théorie
-Rapport pas un bilan d'activité

Prochain rapport :
Faire un deuxième système de calibration par ChAruco.
Calculer cartes de profondeur.
Rendu du rapport de semestre 5
Implémenter des tests unitaires et de performances.
Améliorer l'interface pour prendre en compte l'affichage de calibration.
(Filtre pour les cartes de disparité)