TESTOUD Romain
AVRIL-TERRONES Thibaud
Groupe APP5 Info 5A

Traitement d'image -- TP1

Reconstruction d'une image et évaluation de la précision

Binôme : Romain TESTOUD et Thibaud AVRIL-TERRONES

APP5 Informatique – Polytech' Paris-Sud

Date de rendu : 27/10/2019

TESTOUD Romain
AVRIL-TERRONES Thibaud
Groupe APP5 Info 5A

Sommaire :

Date de rendu : 27/10/2019

TESTOUD Romain
AVRIL-TERRONES Thibaud
Groupe APP5 Info 5A

1 – Objectif du Projet

Ce projet nous confronte à un problème que l'on peut souvent rencontrer dans la vie réelle. Il met en situation une image (ici une fresque) qui a été séparée en de nombreux morceaux que l'on va appeler « fragments ». Le but de ce projet est donc de réaliser un programme permettant de récupérer tous ces fragments et de les assembler afin de reconstituer au mieux l'image de base et par la suite, estimer la qualité de cette reconstitution.

Dans un premier temps, nous allons donc devoir réaliser et implémenter un programme qui devra récupérer tous les fragments que nous avons à disposition, et les replacer au bon endroit, avec le bon angle dans notre nouvelle fresque afin qu'elle ressemble le plus possible à l'originale. Dans un second temps, nous devons réaliser et implémenter un autre programme qui servira à calculer la qualité de la reconstitution d'un fichier comparé à la « vérité terrain ».

2 – Etat des lieux

Binôme :

Notre binôme est constitué de Romain TESTOUD et de Thibaud AVRIL-TERRONES. Nous avons régulièrement eu l'occasion de travailler ensemble sur divers projets, ce qui est un grand avantage car chacun d'entre nous sait comment l'autre fonctionne, ce qui simplifiera les échanges et les méthodes de travail.

Projet :

Une base de code nous est déjà fourni avec le sujet, ainsi que l'image à reconstruire, tous les fragments nécessaires à la reconstruction de la fresque.

Cette base de code contient de fichiers :

- Main.cpp qui nous servira de squelette pour notre projet
- CMakeLists.txt qui nous permettra de compiler et de lancer notre projet

Connaissances techniques :

Le langage de programmation utilisé pour réaliser ce projet est le C++. Nous sommes familiers (mais loin d'être experts) avec ce langage, au vue de nos cours suivis en APP3 et APP4.

Au niveau de OpenCV, nous n'avions pas de compétences avant le début du module « Traitement d'Images ». Nous avons donc dû nous former sur les grands axes de cette librairie avant de commencer le projet.

TESTOUD Romain
AVRIL-TERRONES Thibaud
Groupe APP5 Info 5A

3 – Technologies utilisées et Solutions

Technologies utilisées :

- C++
- OpenCV
    - Version 4.1.1make
- Git

  Afin de garder une trace de l'avancée de notre projet et par mesure de sécurité, nous avons décidé d'uploader notre projet sur GitHub.

  Lien Git : https://github.com/RomsKo9/TI.git

- CMake

  CMake nous permet de construire un projet à partir d'un descriptif de projet (CMakeLists.txt). Il nous évite la gestion manuelle de toutes les dépendances et librairies utilisées dans notre projet, ce qui peut s'avouer être une tâche source de multiples erreurs.

- Xming

  Xming est un système de fenêtrage X des systèmes Linux conçu pour Windows. Nous l'utilisons afin de pouvoir ouvrir des interfaces graphiques, en l'occurrence de pouvoir ouvrir nos matrices images.

- Bash Linux pour Windows

TESTOUD Romain
AVRIL-TERRONES Thibaud
Groupe APP5 Info 5A

Solutions :

Nous allons diviser cette partie en deux sous parties, correspondant chacune a une mission : la reconstruction et l'estimation de la qualité.

- **Reconstruction**

Pour la reconstruction de notre image, la première étape était de modéliser les fragments sous formes d'objet afin de rendre plus simple leur manipulation. Nous avons donc créé une structure nommé Fragment comportant les attributs décrits dans le fichier texte fragments.txt, soit id, posX (position horizontale), posY (position verticale) et angle (angle d'inclinaison du fragment). Nous avons ensuite parcouru ce fichier texte et créé en tant qu'objet tous les fragments renseignés dans le fichier fragments.txt, que nous stockons dans une liste.

Il a fallu ensuite initialiser une image (matrice) vierge de la même taille que l'image à reconstruire, afin qu'elle puisse par la suite accueillir les fragments.

```
cv::Mat res(775,1707,CV_8UC4, cv::Scalar(255,255,255,255));
```

*Création d'une matrice de taille 1707*775, de couleur blanche*

Pour tous ce qui suit, on parcourt chaque fragment de notre liste créée précédemment, et pour chaque fragment, on respecte les étapes suivantes :

- Transformation de nos objets Fragments en matrice

On utilise la propriété imread de opencv afin de transformer une image .png en matrice. On pourra récupérer la bonne image correspondant au bon fragment grâce à l'attribut id de notre structure Fragment.

```
cv::Mat matFrag = cv::imread(pathFragment+std::to_string(frag.id)+".png", cv::IMREAD_UNCHANGED);
```

- Ajuster nos matrice Fragment avec le bon angle

Tout d'abord, il faut changer le centre de rotation de notre matrice qui est par défaut le coin supérieur gauche. Nous avons décidé que le centre de rotation de nos matrices serait le point central. On l'obtient de la façon suivante :

```
cv::Point2f rotPoint((matFrag.size().width)/2, (matFrag.size().height)/2);
```

Date de rendu : 27/10/2019

Par la suite, on utilise la méthode getRotationMatrix2D de la librairie opencv afin de récupérer une matrice correctement orienté grâce à la propriété angle de notre structure Fragment.

- Recopie de notre matrice orientée sur la matrice vierge

La dernière étape a été de recopier les fragments d'images sur l'image vierge créée au début. Nous avons opté pour une recopie pixel par pixel, car après de nombreux essaies, nous n'avons pas réussi à utiliser les méthodes (copyTo par exemple) que nous offre opencv.

Pour finir, on affiche notre matrice finale grâce à la propriété imshow de opencv.

TESTOUD Romain
AVRIL-TERRONES Thibaud
Groupe APP5 Info 5A

- **Estimation de la qualité**

Pour pouvoir calculer une estimation de la qualité de notre reconstruction, il y a plusieurs paramètres à prendre en compte :

- La taille des fragments
- S'ils sont bien placés
- S'ils sont mal placés
- La taille totale des fragments de la fresque

Comme pour la partie reconstruction, nous disposons d'une structure Fragment composée d'un id, d'une position horizontale, d'une position verticale, d'un angle mais aussi d'un attribut size qui représente la taille de la matrice du fragment.

Ici, nous disposons d'un fichier solution.txt possédant la même structure de données que fragments.txt, et qui sera comparé à celui-ci pour établir une estimation de la qualité.

- Créer des deux listes de fragments

La première étape consiste a créé deux listes contenant des fragments, une correspondant au fichier « vérité terrain » (fragments.txt) et une au solution proposées (solution.txt).

Pendant la création de tous les fragments « vérité terrain », nous en avons profiter pour établir la taille totale (fragSizeTotaleFresque) des fragments de la fresque (comme c'est le fichier de référence).

- Vérification des fragments bien placés

On parcourt tous les fragments contenus dans la liste des solutions afin de les comparer à chaque fragment « vérité terrain ». Lorsque toutes les conditions sont respectées (avec la mesure d'incertitude), on incrémente un entier (fragSizeInFresque) représentant la taille totale des fragments bien placés avec l'attribut size de la structure Fragment.

- Vérification des fragments mal placés

On parcourt tous les fragments « vérité terrain » afin de les comparer à chaque fragment contenu dans solution. On initialise un booléen à chaque nouveaux fragment Solution visité à false. En parcourant les deux boucles, si un fragment solution correspond à un fragment vérité terrain, le booléen passe à true, sinon, on incrémente un entier (fragSizeOutFresque) représentant la taille totale des fragments mal placés dans la fresque.

- Calcul de la précision

Grâce aux étapes précédentes, nous possédons maintenant tous le nécessaire afin de calculer la précision. On la calcule comme suit, et le résultat sera exprimé en %.

```
precision = (float)(fragSizeInFresque-fragSizeOutFresque)/(float)fragSizeTotalFresque;
```

4 – Problème rencontrés

- Problème au niveau de l'environnement de développement

Tout d'abord, nous avons rencontrés des problèmes au niveau de l'environnement de développement. En effet, nous avons aborder le projet sur les machines de Polytech, qui sont des machines Linux et où tout est déjà configuré. Or, nous possédons tous les deux des machines Windows, et comme vous vous en doutez, où rien est configuré et installé pour le projet. Nous avons donc passé beaucoup de temps à avoir un environnement de développement opérationnel.

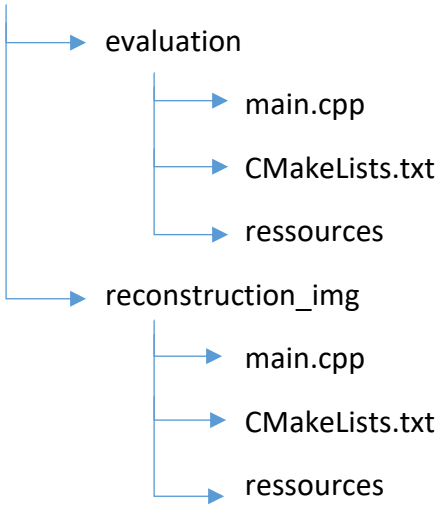
- Connaissance d'OpenCV

Aucun de nous n'avait pratiquer opencv auparavant. Nous avons donc dû nous former « sur le tas » afin de pouvoir répondre aux problématiques du projet.

5 – Compilation et lancement du projet

Structure du projet :

TI_project_1



TESTOUD Romain
AVRIL-TERRONES Thibaud
Groupe APP5 Info 5A

Compilation et lancement :

Tout d'abord, il faut lancer Xming afin de pouvoir ouvrir des fenêtres. Sur windows, Xming peut être téléchargé sur le lien suivant : https://sourceforge.net/projects/xming/

Une fois Xming lancé, vous devriez voir apparaître l'image ci-dessous dans les tâches en cours d'exécution.

De plus, comme nous développons sous Windows (c'est une erreur ;)), nous avons installé un bash Linux.

- Reconstruction de la fresque / Estimation de la qualité

Pour compiler et lancer la partie reconstruction de la fresque, il faut se rendre dans le dossier *reconstruction_img/evaluation* puis exécuter les commandes suivantes dans le bash Linux *:*

- cmake CMakeLists.txt

```
romsko@DESKTOP-1LGECV2:/mnt/c/Users/testo/Desktop/Polytech/5A/TI/TI_project_1/reconstruction_img$ cmake CMakeLists.txt
-- The C compiler identification is GNU 7.4.0
-- The CXX compiler identification is GNU 7.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found OpenCV: /usr/local (found version "4.1.1")
-- Found OpenMP_C: -fopenmp (found version "4.5")
-- Found OpenMP_CXX: -fopenmp (found version "4.5")
-- Found OpenMP: TRUE (found version "4.5")
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/c/Users/testo/Desktop/Polytech/5A/TI/TI_project_1/reconstruction_img
```

- make (pour compiler)

```
romsko@DESKTOP-1LGECV2:/mnt/c/Users/testo/Desktop/Polytech/5A/TI/TI_project_1/reconstruction_img$ make
Scanning dependencies of target lectureImage
[ 50%] Building CXX object CMakeFiles/lectureImage.dir/main.cpp.o
[100%] Linking CXX executable bin/lectureImage
[100%] Built target lectureImage
```

- ./bin/lectureImage ./ressources/Michelangelo_ThecreationofAdam_1707x775.jpg (pour lancer la reconstruction)

Ensuite, une fenêtre Xming contenant la reconstruction va s'ouvrir pour la reconstruction, et un score sera affiché dans le terminal pour l'évaluation.

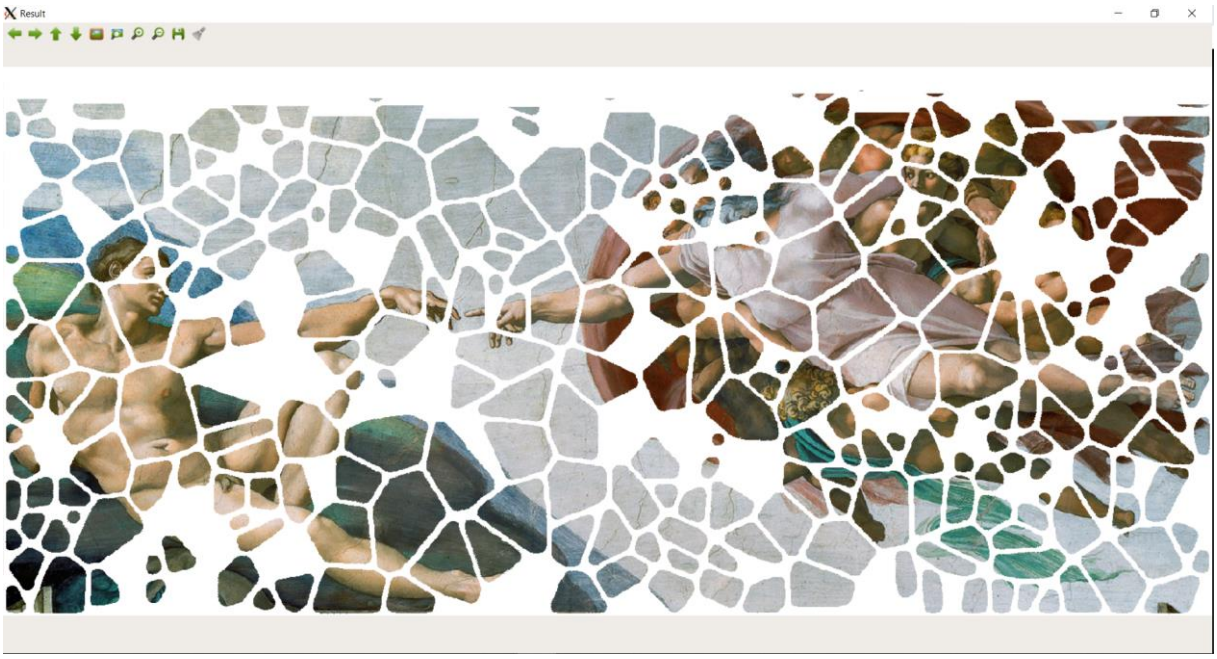
NOTE : Pour lancer reconstruction, il faut être dans le dossier reconstruction_img, dans le dossier evaluation pour le score

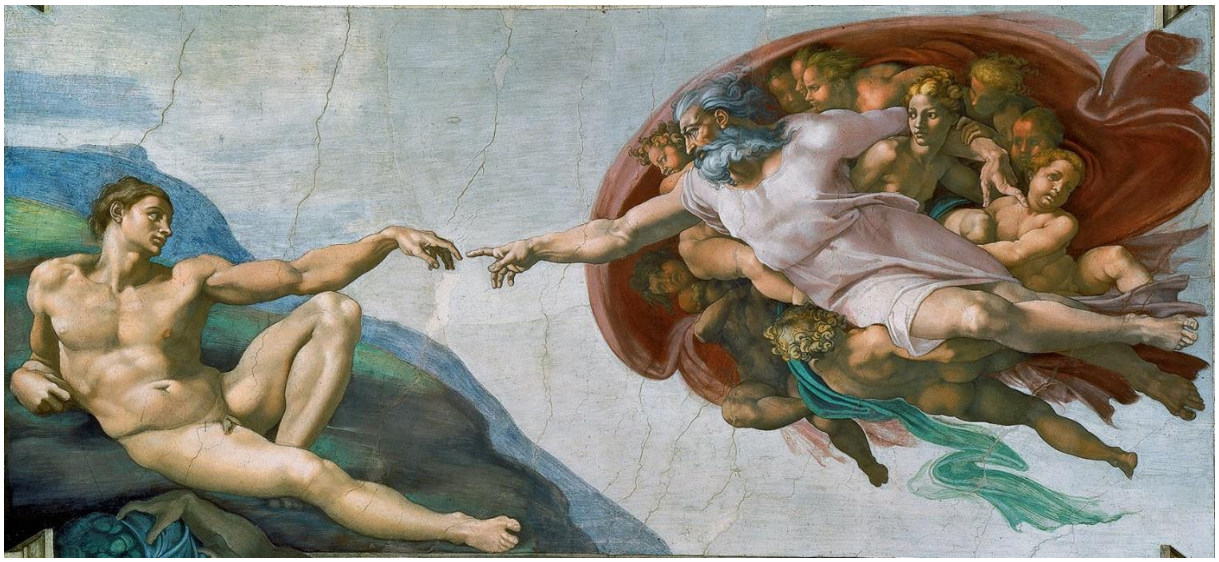
Date de rendu : 27/10/2019

TESTOUD Romain
AVRIL-TERRONES Thibaud
Groupe APP5 Info 5A

6 – Résultats

Reconstruction de la fresque :





Date de rendu : 27/10/2019

TESTOUD Romain
AVRIL-TERRONES Thibaud
Groupe APP5 Info 5A

Estimation de la qualité :

```
romsko@DESKTOP-1LGECV2:/mnt/c/Users/testo/Desktop/Polytech/
Fragments Correctement placés : 153 -- Taille : 2736090
Fragments Mal Placés : 2 -- Taille : 23328
Fragments Manquants : 144
Taille total des fragments de la Fresque :  3737953
Précision : 72.5735%
```

On obtient un résultat nous indiquant le nombre de fragments correctement placés, le nombre de fragment mal placés, le nombre de fragments manquants, ainsi que la précision de la reconstruction du fichier *solution.txt*. Le résultat tient bien évidemment compte des consignes données dans l'énoncé, notamment sur la taille des fragments.

Date de rendu : 27/10/2019