Extraction et analyse d'image

Projet d'étude et de Développement

Manson Tomy
Mestreau Nicolas
Ridel Fabien
Wen Jun

Client :
Jérémy Laviole

Table des matières

1 Introduction

1.1 Presentation

L'objectif de notre projet d'étude et de développement est de produire une bibliothèque d'extraction et d'analyse d'images à partir d'un flux vidéo. Cette bibliothèque sera utilisée pour des applications de vision par ordinateur.

L'extraction d'images consiste à récupérer un espace de dessin dans la vidéo. cet espace de dessin peut être une feuille blanche, entourée par une planche de tags. Au début du projet, une bibliothèque d'extraction (`ARToolkit`) nous a été fournie. Nous devions donc nous concentrer sur les analyses des images obtenues.

Les analyses demandées par ce projet sont de deux types :
- Une analyse rapide, sur une image de faible qualité. Le but de cette analyse est de retrouver des formes simples (voir les templates ci-dessous) dans le dessin.
- Une analyse sur des images de haute qualité. Cette analyse a pour but de détecter les nouveautés dans le dessin.

Des applications utilisant la bibliothèque sont également à développer, notamment une application de détection de nouveautés dans un dessin, et une application de reconnaissance de formes simples.

Exemples de templates

1.2 Contexte

contexte et existant sift surf etc... Detection nouveauté article

2 Objectifs

2.1 Cahier des charges

Dans le but de développer la bibliothèque, nous avons établi un cahier des charges présentant et définissant les trois ensembles de fonctionnalités qui la constitue :

- Pré-traitement des images
- Détection des évolutions d'un dessin
- Reconnaissance des formes dessinées

2.1.1 Pré-traitement

Tout traitement sur une image nécessite d'effectuer un certain nombre de pré-traitements afin d'améliorer la qualité de cette image. Cette étape a pour but de favoriser l'obtention de bons résultats lors de la phase de traitement.

La bibliothèque devra permettre d'appliquer aisément un certain nombre de filtres, notamment :

Filtre morphologique Optimisation des methode de template matching ;

Binarisation Permettre de trouver les différences qui existent entre deux images ;

Correction de couleur Balance des blancs, permet d'annuler la coloration globale de l'image, dépendante de l'éclairage de la scène (lumière du jour, lumière artificielle, etc ...) ;

Filtrage par couleur Récupérer uniquement les formes dessinées avec une couleur précise.

La bibliothèque devra permettre de developper de nouveau filtres. Tous les filtres devrons pouvoir être utilisés de la même manière afin d'en simplifier l'usage, ils devrons avoir une interface commune.

2.1.2 Détection des évolutions d'un dessin

La première fonctionnalité de la bibliothèque est de permettre la récupération de plus haute qualité possible d'un dessin à différents moments. Les moments de prise de vue du dessin seront considérés comme idéaux, pas de mains, ni d'objets dans le champs de prise de vue.

On cherche ici à permettre le développement d'applications permettant d'observer les moments clés de la création d'un dessin sans perturber le travail de l'artiste, c'est à dire, éviter de scanner la feuille.

Performance

La bibliothèque devra permettre l'analyse d'image à une cadence de 2 à 3 images par seconde. Le but ici n'est pas la performance mais bien la qualité de l'analyse. Les images à traiter seront aquises en haute definition, un minimum de 1080 lignes de 1920 pixels, ce qui correspond à la resolution d'un flux vidéo HDTV 1080p.

2.1.3 Reconnaissance des formes dessinées

La deuxième fonctionalité de la bibliothèque est de reconnaitre, dans un flux video, des dessins ou pictogrammes préenregistrés.

Différentes méthodes pourrons être développées selon que les formes à reconnaitre dans l'image sont simples (p.ex. : un carré, un cercle) ou complexe(p.ex. : un visage).

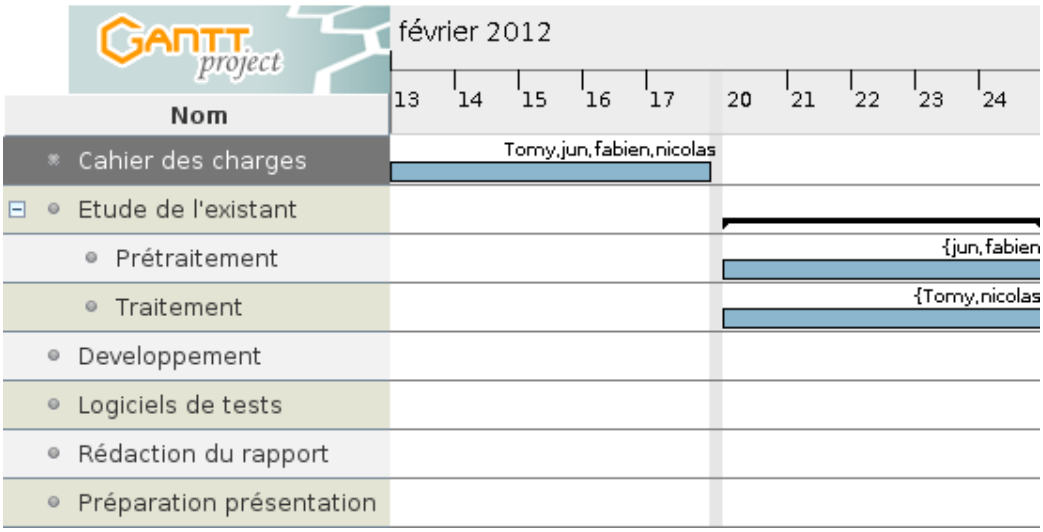
On utilisera par exemple :

- Template Matching : Mise en correspondance d'une image avec une image de référence pour la detection de formes simple.
- Feature detection : Utilisation de caractéristique invariante dans l'image pour la reconnaisance de formes complexe (par exemple : descripteur SURF).

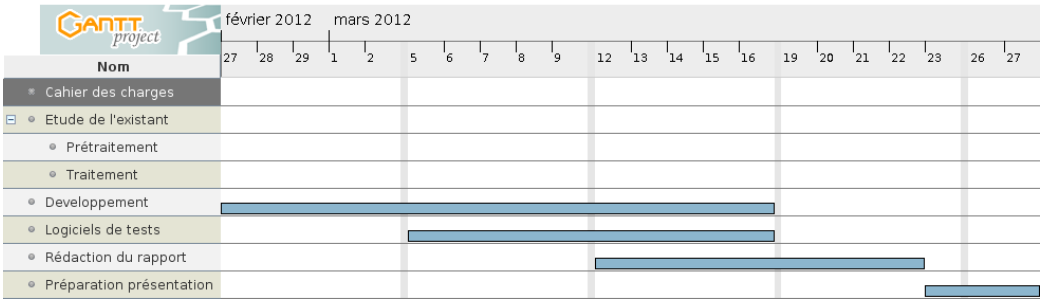
Performance

La bibliothèque devra permettre l'analyse d'image à une cadence de 30 à 60 images par seconde. Le but ici est d'effectuer de l'analyse sur un flux vidéo temps réel, de basse qualité. Les images à traiter seront aquises en basse qulaité, on prendra comme base des vidéos au format vidéo VGA 480 lignes de 640 pixels, ce qui correspond au format VGA.

2.2 Planning



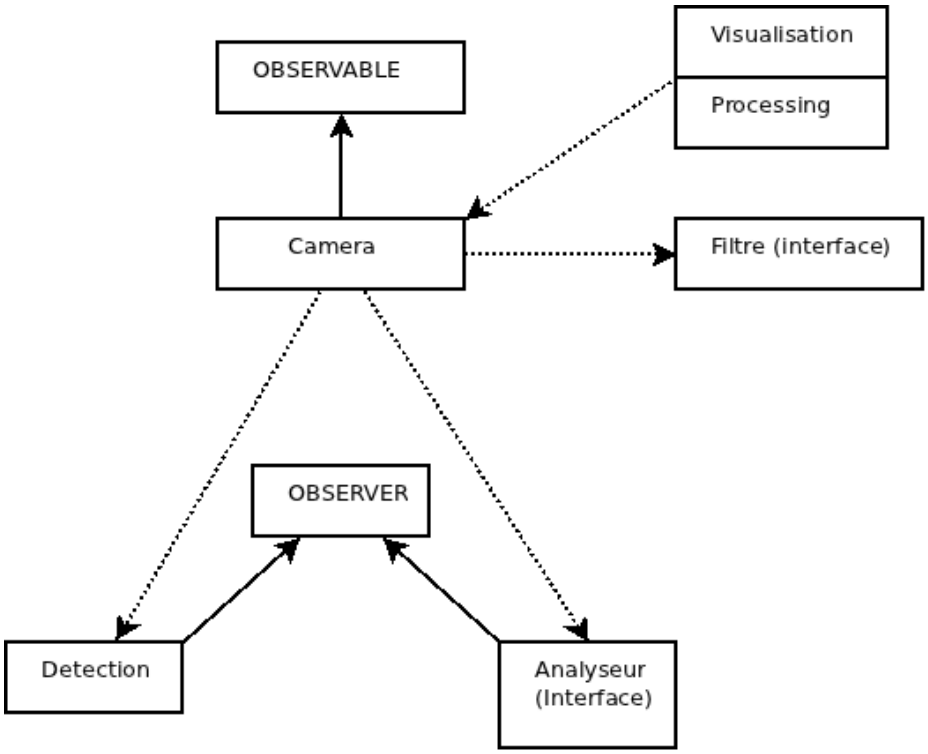
Planning taches préparatoire



Planning de dévelopement et de fin de projet

3 Travail accompli

3.1 Architecture logicielle



3.2 Pré-traitement

3.3 Traitement

3.3.1 Reconnaissance de formes dessinées

Afin de reconnaître des formes simples dans une vidéo, nous avons mis en place un système de détecteur.

Nous avons implémenté deux détecteurs différents. Le premier s'appuie sur du template matching pour reconnaître une forme. Le second utilise les descripteurs SURF.

Pour faire ainsi, nous avons mis en place une classe abstraite `Detector` qui présente les fonctions de base d'un détecteur, à savoir ajouter, récupérer ou supprimer un template, et définit la fonction permettant de lancer la détection. Chacun de nos détecteurs (`MatchingDetector et SurfDetector`

doit donc implémenter cette fonction de détection.

Template matching

Notre premier détecteur est le `MatchingDetector`, qui utilise l'algorithme de template matching.

L'algorithme de template matching consiste à faire glisser un template sur la surface de l'image, et de calculer un coefficient de "corrélation" entre le template et la zone qu'il recouvre.

Pour effectuer cette procédure, nous parcourons la liste des templates du détecteur. Pour chacun de ces templates, nous avons utilisé la fonction `cvMatchTemplate` d'OpenCV pour calculer une image de corrélation entre le template et l'image tirée de la vidéo, qui montre les endroits où le template a le plus correspondu à l'image. Après normalisation, nous parcourons cette image pour récupérer le minimum (ou maximum suivant la méthode passée en paramètre à `cvMatchTemplate`), ce minimum/maximum est l'endroit de meilleur matching. Nous stockons alors le quadruplet (x, y, w, h) avec :

x,y Les coordonnées du point de meilleur matching correspondant au coin supérieur gauche de la zone rectangulaire représentant l'objet détecté.

w,h Les dimensions du template, correspondant aux dimensions de la zone rectangulaire représentant l'objet détecté.

Une fois la position du meilleur matching récupérée, nous retirons de l'image source la zone rectangulaire décrite par le quadruplet (x, y, w, h). Cette étape permet d'éviter que deux templates trop proches détectent le même objet alors que deux objets sont valides (exemple : Sur un dessin, deux bonhommes sont dessinés. Avec deux templates de bonhomme, il est possible que les deux matchings trouvent comme meilleur résultat le premier bonhomme et ne détectent pas le second. Notre étape de suppression permet de faire en sorte qu'une fois que le premier matching a détecté le premier bonhomme, celui-ci n'est plus pris en compte pour les détections des autres templates. On augmente ainsi les chances de détecter tous les objets de l'image.)

Une fois tous les templates parcourus, nous retournons l'ensemble des quadruplets (x, y, w, h) correspondant aux détections.

Un des défauts de notre `MatchingDetector` est que le retour de `cvMatchTemplate` ne nous permet pas de calculer un seuil de confiance pour accepter ou rejeter le résultat du matching. On peut récupérer l'endroit où le template a le plus correspondu à l'image, mais on ne sait pas si il a matché à 99% où à 10%.

Ce défaut, couplé à la normalisation du résultat de `cvMatchTemplate` (première tentative, infructueuse, de déterminer un seuil de confiance) entraîne un faux positif lorsque le template ne matche pas un des objets de l'image.
Le problème est double : pour le moment, nous avons un template = un résultat. On a donc forcément des faux positifs lorsque le nombre de templates d'un type d'objet dépasse le nombre d'objets de ce type du dessin, ou quand le template ne matche pas à un des objets du dessin. Nous avons également des faux négatifs, donc des objets qui ne sont pas détectés, lorsque le nombre de templates d'un type d'objet est inférieur au nombre d'objets de ce type dans le dessin.

Une solution au problème serait d'arriver à calculer un indice de confiance du matching. Si cet indice de confiance ne dépasse pas un seuil à définir, on ne prend pas en compte le résultat du matching pour ce template.

Cette solution permet de boucler sur un template jusqu'à ce que l'indice de confiance calculé ne passe plus le seuil. Cela permet d'avoir un template par type d'objet, et non un template = un résultat positif comme à présent. Avec peu de templates de bonhomme, on pourrait détecter un grand nombre de bonshommes dans le dessin. On réduirait ainsi le problème de faux négatifs lorsque le nombre de templates d'un type d'objet est inférieur au nombre d'objets de ce type. Et comme on ne prend pas en compte les matchings qui ne passent pas le seuil, on réduit également le nombre de faux positifs.

SURF : Speeded Up Robust Features

L'algorithme SURF consiste à calculer des descripteurs sur les templates ainsi que sur l'image. Une fois ces descripteurs calculés, on en fait quelque chose.

"SURF[11] : Speeded Up Robust Features" is a high-performance scale and rotation-invariant interest point detector / descriptor claimed to approximate or even outperform previously proposed schemes with respect to repeatability, distinctiveness, and robustness. SURF relies on integral images for

image convolutions to reduce computation time, builds on the strengths of the leading existing detectors and descriptors (using a fast Hessian matrix-based measure for the detector and a distribution-based descriptor). It describes a distribution of Haar wavelet responses within the interest point neighbourhood. Integral images are used for speed and only 64 dimensions are used reducing the time for feature computation and matching. The indexing step is based on the sign of the Laplacian, which increases the matching speed and the robustness of the descriptor.

filtre -¿ patron de conception "patron de méthode" permet d'unifier les comprtement des filtres et factoriser du code "factory"

4 Tests et résultats

4.1 Presentation des logiciels de test

4.1.1 Reconnaissance de formes simples dans un dessin

Le premier logiciel que nous avons développé est un logiciel de reconnaissance de formes simples dans un dessin. Ce logiciel commence par lire une vidéo. L'utilisateur peut à tout moment mettre en pause la vidéo, et prendre des captures de la vidéo.

Lorsqu'une capture est prise, notre logiciel effectue des pré-traitements sur l'image obtenue, puis lance l'analyse de template matching. Le logiciel affiche ensuite le résultat de l'analyse, à savoir la capture prise dans la vidéo enrichie de `Bounding Box` autour des objets détectés.

Il est ensuite possible de naviguer entre les différentes captures.

4.1.2 Détection des évolutions d'un dessin

Le second logiciel développé est un logiciel de détection des évolutions d'un dessin. Ce logiciel propose de comparer deux images d'un dessin. Il affiche ensuite les différences entre ces deux images.

5 Conclusion

Bilan
Evolution possible applications

Références