ENSICAEN
ÉCOLE NATIONALE SUPÉRIEURE D'INGÉNIEURS DE CAEN
& CENTRE DE RECHERCHE

ENSICAEN - Projet 2A Informatique
Reconnaissance faciale : anti-fake

VIMONT Ludovic & KOTULSKI Guillaume
Promo 2016

Une grande école pour réussir

ENSICAEN
6, boulevard Maréchal Juin - F. 14050 Caen
Tél. +33 (0)2 31 45 27 50
Fax +33 (0)2 31 45 27 60

Remerciements

Nous tenons à adresser nos remerciements aux personnes qui ont permis que ce projet se déroule dans les meilleures conditions possibles.

Nous remercions M. Jean-Jacques Schwartzmann pour le temps et les conseilles qu'il nous a accordé, mais surtout pour nous avoir permis de travailler sur un projet innovant et intéressant.

Nous remercions également Mme. Estelle Cherrier pour son travail dans l'organisation des différents projets.

Enfin, nous remercions M. Ndiaga Faye pour son aide.

Sommaire

1 Introduction

1.1 Contexte

Dans le cadre des cours de l'ENSICAEN, nous devions réaliser un projet de deuxième année. Nous avons choisi de prendre le projet reconnaissance faciale anti-fake proposé par M. Schwartzmann Jean-Jacques.

1.2 Objectifs

L'objectif du projet est la réalisation d'un moyen d'authentification par reconnaissance facile grâce à l'utilisation d'une capture vidéo. En effet, actuellement ce genre d'applications sont facilement attaquables. Prenons l'exemple d'un smartphone qui se déverrouille grâce à ce moyen. Il suffit pour un attaquant de disposer de la photo de la victime pour contourner la protection. Ce projet consiste donc à utiliser une capture vidéo et être capable de différencier un humain d'une photo grâce à cette enregistrement. Afin de réaliser cette différenciation, nous devions baser notre travail sur une recherche du MIT. Enfin, nous devons intégrer notre résultat dans une application Android de reconnaissance faciale fournie par le GREYC.

1.3 Contraintes

La seule contrainte dont nous disposions était le temps de reconnaissance de la personne, en effet pour l'application Android au-dessus de 4 secondes cela était beaucoup trop long, d'un point de vue utilisateur.

2 Développement

2.1 Outils mis en place

— Site web en ligne (CMS Wordpress) : http://www.ecole.ensicaen.fr/~lvimont/

FIGURE 2.1 – Capture d'écran du site web.

— Gestionnaire de versions (Git)
  - Application Android : https://github.com/F4r3n/Vamp
  - Logiciel de traitement d'images : https://github.com/F4r3n/ImageProject.git

— Gestionnaire de projet (Trello)

2.2 Technologies utilisées

- QT :Nous avons utilisé la librairie Qt pour pouvoir créer notre application qui permettait de tester les algorithmes.
- Android :La programmation sous Android s'est faite en utilisant Java et de l'XML.
- OpenCV :Utilisation de la webcam et application utilisant la caméra gérée par OpenCV
- Pyhton : Traitement des données pour création d'un graphique et permet une meilleure vérification des résultats

2.3 Algorithme mis en place

L'œil humain est limité, en effet il est incapable de voir les subtils changements temporels. Alors que justement en effectuant des traitements sur une vidéo. On est capable de révéler ces changements comme par exemple, la circulation du sang. On peut même grâce à ça réussir à connaître le pouls d'une personne.
De plus lorsque le sang monte jusqu'à la tête puis descend, il fait que notre tête bouge imperceptiblement, ces différents changements peuvent capturés avec un téléphone portable.
Mais comment obtenir ces changements ? Comment savoir où se situe le visage ?

Dans un premier temps nous devons détecter si quelque chose ayant un visage humain se tenait devant l'appareil, ceci se fait très bien par l'api de Google sur Android et même par OpenCV. Après détection de la personne nous pouvons obtenir des coordonnées nous disant où se situe le visage.
Maintenant que nous savons où se situe le visage, nous nous demandons comment reconnaître une fréquence ?
Nous avons choisi l'espace de couleur RGB car c'est dans cette espace de couleur que nous détectons le plus de variations, les autres modes tel que HSL ou HUV nous donnaient des valeurs plus faibles. Nous avons donc fait une moyenne du canal vert (c'est le canal vert qui varie le plus) pour chaque pixel se situant dans un rectangle (en l'occurrence le rectangle délimitant le visage). Et ceci pour chaque frame que l'on capturait, nous obtenions une certaine valeur. Après obtention de ces valeurs nous pouvions les traiter. La variation des valeurs étant très grande, nous avons dans un premier temps fait une moyenne glissante. Puis pour amplifier les variations nous avons avons fait une dérivée de Taylor. Nos valeurs étant prêtes à être analyser, nous les avons analyser de deux façons différentes. La première, la plus simple fut le Trigger, qui nous permettait d'avoir une fourchette de pulsation.
La deuxième plus longue,une FFT, nécessitait des étapes de calcul supplémentaire tel que un fenêtrage de Hamming et un padding, mais donnait des résultats plus précis.
Ces deux analyses nous permettait d'avoir le pouls de notre utilisateur (dans le cas idéal).

Nous avons remarqué que la collecte des données n'accentuait pas assez les variations, nous avons donc créé une deuxième méthode qui en restant basé sur le même principe que la première technique, nous découpions notre image par zone de petits carrés de pixels. Par exemple une zone serait d'une taille de 5*5. On réalise ainsi une moyenne de chaque zone, puis par la suite une moyenne de ces moyennes. Enfin on rapplique les mêmes fonctions citées précédemment.

2.4 Solutions mis en place

Durant le projet nous avons développés plusieurs outils pour répondre au besoin. Une application Android qui permettrait d'utiliser le moyen d'authentification désiré et ainsi déverrouiller le smartphone d'une personne. Un logiciel C++ utilisant la librairie QT a également été mis en place, afin de pouvoir tester plus facilement les algorithmes que nous voulions développer. Dans la suite du projet, il a également permis d'utiliser la webcam.

2.4.1 Développement Android

Notre application Android est capable d'utiliser la caméra frontale de n'importe quel smartphone. L'api d'Android propose une classe appelée FaceDetectionListener qui va permettre de capter, les visages présent devant une caméra, grâce à ce système il est possible de dessiner un Rect (c'est à dire un rectangle) autour du visage de la personne. C'est notre classe DisplayedFace qui se charge de ce travail. Une fois la reconnaissance mis en place, nous avons utiliser la fonction onPreviewFrame. Cette dernière permet d'enregistrer en temps réel, les données. On lance un compteur quand la première reconnaissance d'un visage a lieu, ce dernier va durer 5 secondes.

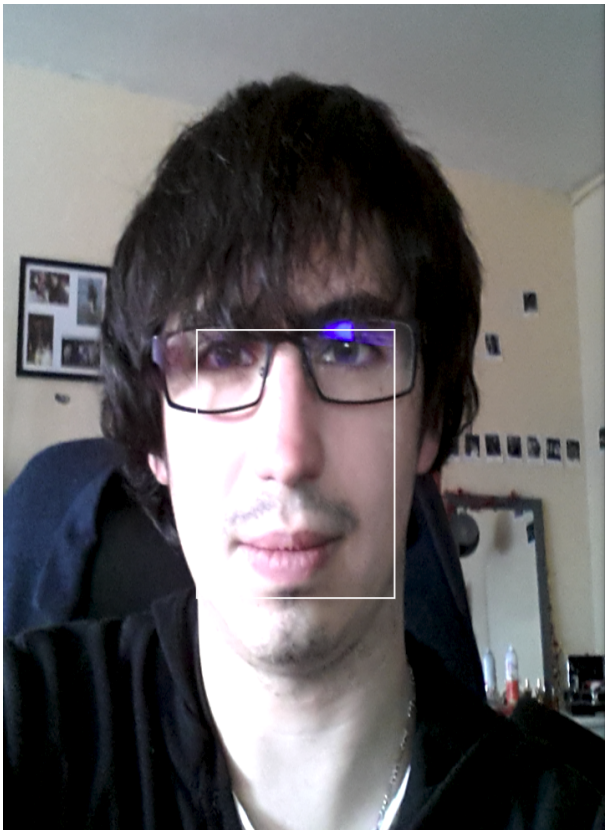

FIGURE 2.2 – Aperçu de l'application Android

Les données ainsi récoltées sont au format de l'espace de couleur YUV. Nous devons donc convertir ces données au format RGB qui se révèle plus évocateur au niveau des variations. A la fin du compteur, on réalise une dérivée, puis une amplification des valeurs obtenues. On observe alors les variations qu'on obtient et on conclu alors sur la présence ou non d'un humain.

L'api de Google est très rapide pour détecter un visage et pour nous fournir ces coordonnées en faisant très peu de calcul. Mais celle-ci ne suit pas les petits changements de position, par exemple lorsque l'utilisateur prend une vidéo il bouge légèrement ce qui peut changer le résultat final.

Pour résoudre ce manque nous avons codé une application fonctionnant avec OpenCV, celle-ci suit parfaitement le visage lorsqu'il bouge. Mais le temps de traitement d'une image par OpenCV est beaucoup plus lent que l'api de Google. Ce qui fait que nous tournons à environ 7 FPS en back-caméra et 2 FPS en front-caméra ce qui n'est pas idéal.

L'application sous OpenCV permet de faire une FFT et un système de Trigger, en croisant les résultats donnés par ces deux systèmes nous obtenons un meilleur résultat que si il n'y avait seulement le Trigger.

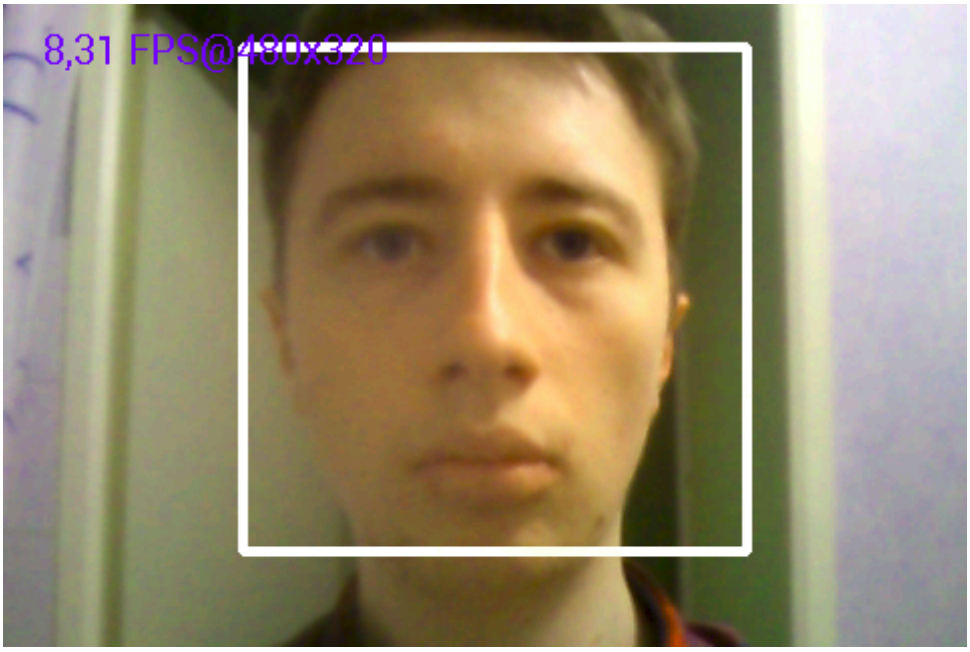

FIGURE 2.3 – Aperçu de l'application OpenCV

2.4.2 Développement C++

Nous avons créé un logiciel C++ nous permettant de tester nos algorithmes avant de les implémenter sous Android. Ce logiciel a de nombreuses fonctionnalités, notamment le découpage d'une vidéo en images pour permettre une analyse plus poussée. Nous utilisons avconv qui permet de découper une vidéo de n'importe quelle taille en une multitude d'images (15 images * taille de la vidéo) Après avoir découpé la vidéo en images nous faisons une analyse sur ces images. Cette analyse se découpe en plusieurs étapes. Dans un premier temps nous dessinons un rectangle autour d'un visage (si la fonction du rectangle automatique n'est pas enclenchée) Puis nous choisissons le type d'analyse a effectué, il y en a deux types :
-L'une fait la moyenne de tous les pixels qui se situe dans le carré
-La seconde découpe le rectangle en plusieurs zones de 5*5 puis amplifie les variations et fait une moyenne des valeurs Après avoir lancé l'analyse, nous affichons les courbes ainsi obtenues par notre application.

Nous pouvons voir la FFT, l'amplification, la dérivée ou une combinaison de plusieurs méthodes. L'amplification utilise la dérivée de Taylor du premier degré. La FFT utilise un filtre passe-bas combiné avec une fenêtre de Hamming et un padding. La FFT permet de connaître le rythme cardiaque d'une personne de manière précise si l'image est assez stable. Nous avons créé un Trigger qui permet aussi de connaître la fréquence cardiaque d'une personne mais de manière moins précise, le Trigger nous donne une fourchette de valeurs.

Si l'espace de couleur que nous avons choisi ne donne pas des résultats corrects nous pouvons en choisir un autre. Nous avons différents espaces de couleur disponible dont RGB, HSL, HUV et nous pouvons lancer une analyse avec un espace de couleur différent de RGB.

Les analyses avec des vidéos classiques n'étant pas suffisantes pour nos tests, nous avons dû faire une analyse avec des vidéos de la webcam. La webcam est configurée et lancée via la librairie OpenCV et nous enregistrons les images de la webcam avec une fréquence de 15 frames pas seconde.

2.5 Problèmes rencontrés

2.5.1 Limitation des sessions à l'ENSICAEN

Lors du projet, nous avons rencontrés de nombreux problèmes. L'un des plus embêtant est la limite des sessions à l'Ensicaen. En effet, nos sessions dispose du taille limité, nous pouvons uniquement travailler sous Windows (car c'est seulement sous cet environnement que sont installés les outils Android) hors nous disposons d'uniquement 150 Mo, or rien qu'avec Firefox, si nous ne nettoyons par régulièrement l'historique, la session se retrouve complète. Nous avons malheureusement expérimenté le soucis et perdu ainsi, une après midi de travail.

2.5.2 Problèmes rencontrés lors du développement Android

Nous avons eu un problème de rotation, les coordonnées que l'on converser se révélait mauvaise. On avait en effet un problème de rotation, afin d'éviter cela on force le mode portrait.

La première version de notre application était capable d'enregistrer 15 frames par seconde, or au final on obtenait seulement une trentaine de valeurs, cela était du à la conversion du format YUV au format RGB que l'on effectuer à chaque fois que nous rentrons dans la fonction onPreviewFrame, comme la conversion est coûteuse $O(n^2)$, on perdait des valeurs. Pour optimiser, le nombre de valeurs nous effectuons maintenant la conversion une fois le timer fini. On réalise alors dans la boucle une simple sauvegarde des données bruts. Toutefois cette méthode, nous a causé pas mal de soucis, notamment des out of memory, c'est à dire, que nous réalisions une allocation trop grande par rapport à la mémoire disponible ...

Même si l'algorithme fonctionnait avec des vidéos, ceci ne voulait pas dire qu'il fonctionnerait sous Android. En effet lorsque avec la caméra on se prend en vidéo on bouge légèrement, ce qui pose de nombreux problèmes de stabilité. Les variations que nous obtenons alors ne seraient plus celle de la pulsation mais celles du mouvement régulier de la main.

3 Résultats

3.1 Cadre idéal

Nous avons effectuer différents tests durant le projet. En utilisant les vidéos fournis sur le site du MIT. Notre logiciel arrive à obtenir des résultats très correct. En effet, par exemple comme on peut le voir sur cette capture d'écran. Après notre magnification réalisé, on retrouve une fréquence cardiaque d'environ 53 battements par minute (BPM), l'article du MIT lui trouve une fréquence de 54 BPM.

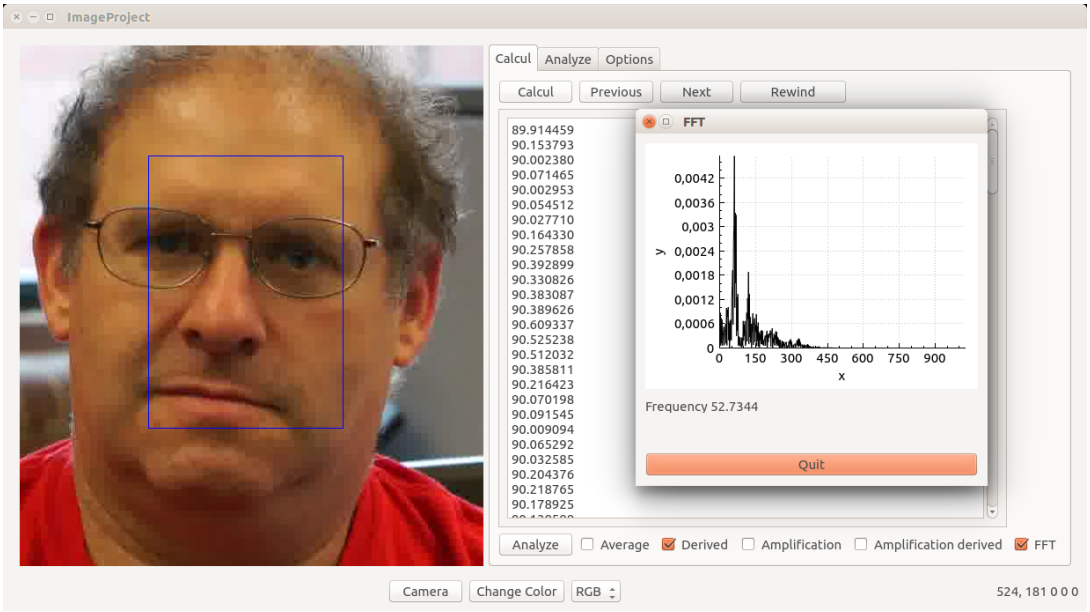

FIGURE 3.1 – Vidéo source du MIT analysé avec notre logiciel.

3.2 Webcam

Avec une webcam, nos résultats sont moins précis, mais sont assez correct pour être utiliser.

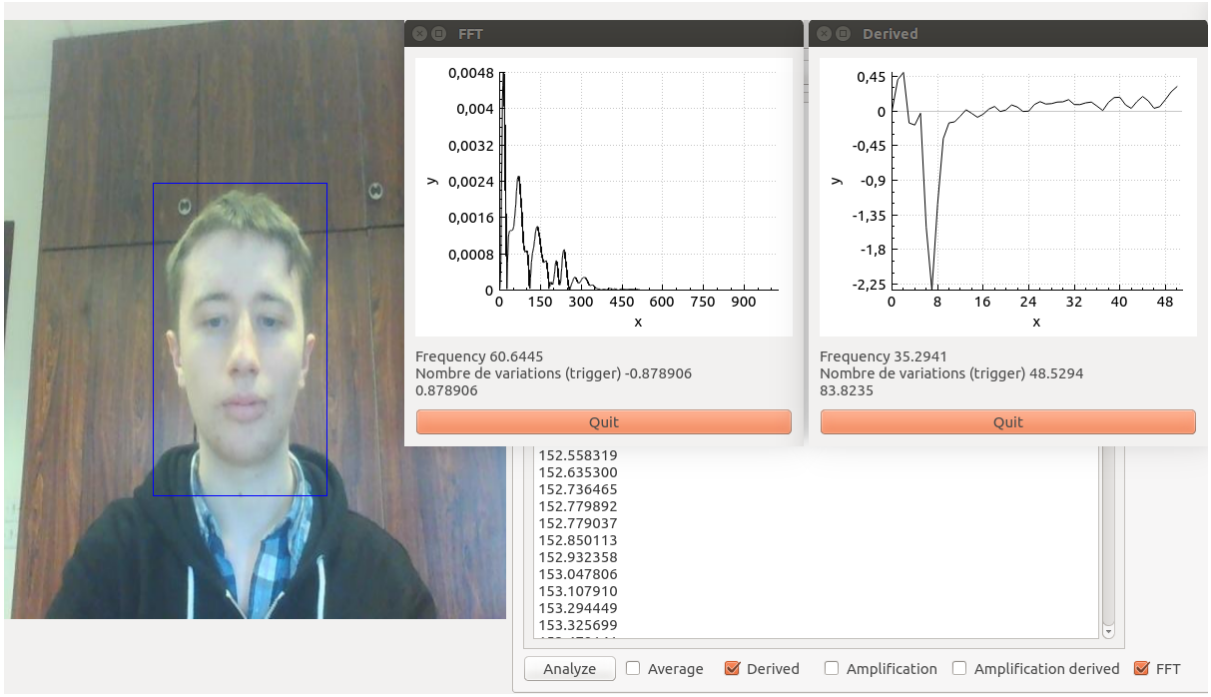

FIGURE 3.2 – Vidéo pris par la webcam et analysé avec notre logiciel.

3.3 Android

Notre plus gros problème a bien entendue était lors de nos tests sur Android, la stabilité de la vidéo et le nombre de frames capturés. Actuellement notre application est capable de capturer des frames et appliqué notre algorithme mais nos résultats restent erronées.

4 Conclusion

4.1 Apports techniques

Ce projet nous a permis de mettre en œuvre l'enseignement que nous avons eu par l'ENSICAEN, l'utilisation de Github, les cours de C++ et de conception d'interface graphique. Il nous a également permis d' apprendre le développement mobile et l'utilisation de librairie comme OpenCV.

4.2 Apports scientifiques

Nous avons également pu mettre à profits les cours que nous avons eu sur les différents espace de couleurs, surtout utile comme nous l'avons vu lors du développement Android. Le plus gros défi fut la mise en place de techniques du traitement du signal, telle que la FFT ou encore la réalisation de filtre basse fréquence.

4.3 Bilan

Ce projet, nous a forcé à aller chercher très régulièrement dans la documentation car nous découvrions une nouvelle technologie et à réutiliser tout ce que nous connaissions, ça a donc été un vrai aboutissement de notre formation.

5 Bibliographie

http://en.wikipedia.org/wiki/Fast_Fourier_transform

http://en.wikipedia.org/wiki/Window_function

https://developer.android.com/guide/index.html

A Annexes