OpenFaceTracker

BEASSE Maxime
étudiant 4A, ESIEA
Laval, FRANCE
beasse@et.esiea.fr

***Abstract*—OpenFaceTracker (OFT) est une librairie de détection et de reconnaissance faciale Open Source implémentée en C++. L'objectif est d'apporter un module d'analyse faciale simple, rapide et performant, à d'autres projets, qu'elle que soit leur nature. Le développement d'OFT a débuté en 2016 et a vu beaucoup de changements dans les technologies et algorithmes utilisés. Cet article rapporte le fonctionnement et l'usage de la version 6.0.X, développée en 2020. Ce projet est soutenu par le laboratoire « Confiance Numérique et Sécurité » (CNS) de l'ESIEA Ouest.**

Keywords—détection faciale, reconnaissance faciale, visage, Open Source, c++, cross-platform

INTRODUCTION

La détection et la reconnaissance faciale sont des problèmes qui ont depuis longtemps été étudiés et documentés. La capacité de détecter un visage à partir d'une image et pouvoir l'identifier par ordinateur est une fonctionnalité qui est nécessaire dans beaucoup de projets mais peu facile d'utilisation et souvent imprécise. C'est pourquoi OpenFaceTracker a été proposé, une solution pour n'importe quelle application, flexible, et sans cesse améliorée. La particularité d'OFT, par rapport à d'autres dépendances est que la base de données est entièrement gérée par la libraire, enlevant ainsi un poids aux utilisateurs. En plus d'être une librairie C++, une application Linux/Windows est proposée pour une utilisation directe et simple. Il est à noter que ce rapport est basé sur la version 6.0.X d'OpenFaceTracker, et que de futures modifications du fonctionnement interne invalideront certaines sections de ce papier, la documentation technique sera mise-à-jour.

I. DEMARRAGE RAPIDE

A. Installation

OpenFaceTracker dépends de librairies externes pour fonctionner. Pour faciliter l'installation, des scripts ont été rédigés pour Linux (Ubuntu, Debian/Kali, Mageia, CentOS et similaires) et Windows 10 (Windows 7 n'a pas été testé mais devrait être supporté).

Ces scripts vont ainsi installer et configurer les libraires suivantes et leurs propres dépendances :

- OpenCV v4.3.0 (avec [contrib] et [ffmpeg])
- cppdb v0.3.1(avec sqlite[tool])
- boost-filesystem v1.73.0
- curl v7.71.1
- nlohmann-json v3.9.1

Les étapes d'installations sont décrites dans le dépôt Github du projet. Il est à noter que lors de prochaines mises à jour, la liste de dépendances sera sujette à changements. Les scripts seront respectivement modifiés pour mettre à jour celles-ci. Aussi, une attention particulière a été apportée à la limitation de l'espace nécessaire aux dépendances, mais il reste à prévoir un espace conséquent pour le bon fonctionnement de la librairie (notamment sur Windows).

Sur Windows, le gestionnaire de paquets vcpkg [1] est installé automatiquement si nécessaire et est utilisé pour compiler les dépendances. Veuillez prévoir un espace de travail conséquent pour cette installation.

B. Compilation

Une fois les dépendances installées, il ne reste qu'à compiler les sources pour générer :

- L'application desktop oftapp(.exe) sous `/bin`
- La librairie dynamique oft (so/dll et a/lib) sous `/lib`
- La documentation Doxygen de la librairie sous `/doc`

Encore une fois, les étapes de compilation sont détaillées dans le dépôt Github du projet. A noter que, sur Linux et Windows, le script cmake génère un Makefile. Il est possible de changer le générateur en modifiant le fichier `PreLoad.cmake` [2].

II. FONCTIONNEMENT

L'application OpenFaceTracker, qui n'est qu'un wrapper de la librairie, est très simple d'utilisation. Dans cette section, nous verrons les possibilités et le fonctionnement global de l'application.

A. Source

Les sources acceptées par OFT sont nombreuses. Tout d'abord, tout fichier local de type JPEG, PNG, AVI, MP4 ou MKV. OFT propose également un module de téléchargement de fichier distant. Ainsi, passé l'URL atteignable d'un fichier, le programme va télécharger le fichier automatiquement dans `/tmp` puis continuer son analyse avec le fichier en cache.

Les flux URL de schémas HTTP ou RTSP sont également supportés, pour analyser un flux en direct comme une caméra IP par exemple.

Enfin, les caméras connectées à l'ordinateur sont également supportées, cela comprend les webcams notamment.

B. Base de données

OFT gère elle-même sa base de données comprenant les informations des personnes connues. La gestion de ces données est faite via les options `--add` et `--remove`. Chaque personne se voit attribuer un label personnalisable et un vecteur de données qui l'identifie.

L'ajout d'une personne se fait en indiquant une source – qui n'affiche que la personne à ajouter – et son label. Le label ne sert qu'à l'utilisateur pour identifier quelqu'un de son côté. Il est recommandé de choisir un identifiant anonyme lorsque le flux de communications entre l'application et l'utilisateur est à risque.

Par défaut, l'application va, à chaque image, détecter les visages et ne les prélever pour la personne à ajouter que si l'utilisateur a enfoncé le clic gauche au même moment. L'option `--yes` permet de prendre en compte tous les visages détectés, **sans confirmation de l'utilisateur**. C'est une option à utiliser avec précaution pour deux raisons :

- La détection faciale n'est pas assez fiable et il y a souvent des faux positifs si les conditions ne sont pas optimales.
- Il faut faire attention à n'utiliser une source qui ne contient que la personne à ajouter.

Dans tout les cas, l'ajout d'une personne dans la base de données doit être faite dans un environnement supervisé.

L'algorithme de reconnaissance faciale, expliquée III.B, supporte la mise à jour. Ainsi, il est recommandé de définir une personne avec plusieurs sources différentes pour apporter des variations pour de meilleures performances.

C. *Configuration*

Les modules d'OpenFaceTracker dépendent de fichiers extérieurs, comme la base de données ou le fichier de log. Un fichier de configuration dans `data/config.json` est donc mis à disposition pour faciliter l'utilisation et le déplacement des fichiers.

III. Cœur de OFT

Le cœur d'OpenFaceTracker correspond aux modules de détection et de reconnaissance faciale. Les algorithmes choisis et leur fonctionnement seront détaillés dans cette section.

A. *Détection*

Le module de détection doit, pour une image donnée, détecter les visages de l'image, puis retourner les rectangles correspondant à leur position dans l'image. L'algorithme utilisé dans OFT est la méthode Viola-Jones [3].

Fig. 1. Méthode Viola-Jones (Haar-features Cascade Classifier)

Cette méthode, dont on peut voir le fonctionnement Fig. 1, est une cascade de caractéristiques qui va chercher à identifier un visage en fonction des différences de nuances entre certaines zones de celui-ci. Il a pour avantage d'être très rapide et d'avoir des coûts de computation très faibles. L'intérêt est également que l'on peut entraîner ces cascades pour détecter n'importe quel objet, et il existe un choix assez conséquent de cascades sur Internet pour la détection de visages.

Cependant, cela reste une méthode qui dépends beaucoup des variations de luminosité, de l'inclinaison de la tête et des reflets sur les lunettes. Et il existe de nombreuses autres solutions qui sont plus efficaces, produisent moins de faux positifs et sont tout aussi rapides.

OpenCV proposant une implémentation simple de cet algorithme, nous avons gardé l'utilisation de celui-ci. Aussi, la particularité de la cascade que l'on utilise est qu'il ne détecte les visages que de face et peu inclinés. Filtrer les visages détectés pour ne garder que des visages bien alignés est nécessaire pour le bon fonctionnement de l'algorithme de reconnaissance faciale utilisé. Un changement d'algorithme de détection est envisageable, mais n'est pas prioritaire et ne changera en rien l'utilisation d'OpenFaceTracker.

(voir `FacialDetection` *dans la documentation technique)*

B. *Reconnaissance*

Le principe général de la reconnaissance faciale est de monter une base de données d'images de visages sujets connus (i.e. labellisés). Ces vecteurs de données à haute-dimension (une image de 60x60px est un vecteur de dimension 3600) sont ensuite projetés dans un espace vectoriel de dimension moindre (admettons 150 axes) pour former un point. L'opération de projection et de réduction de dimension est faite de telle sorte que les visages similaires sont proches dans cet espace vectoriel, et inversement pour les visages dissimilaires. Ainsi, lors de la projection d'une image de visage à identifier, la distance avec les visages connus correspond à un « *score de similarité* » dit *confidence*. C'est cette valeur qui permet de déterminer quel est le visage le plus similaire pour attribuer au visage inconnu une identité. On attribue –arbitrairement ou non– un seuil qui, s'il est dépassé par la *confidence*, signifie que le visage testé est inconnu.

Le plus gros de ce problème est l'algorithme de réduction de dimension. Un algorithme très connu pour cela est *Eigenface* [4]. Celui-ci effectue une Analyse en Composante Principale [5] (PCA) à partir des visages connus. Cette opération va dégager de la base de données les caractéristiques qui séparent au mieux les visages connus pour former un espace vectoriel en prenant ces composantes pour axes. On peut utiliser ces caractéristiques dans des curseurs pour générer des visages à partir du visage moyen (cf. Fig. 2).

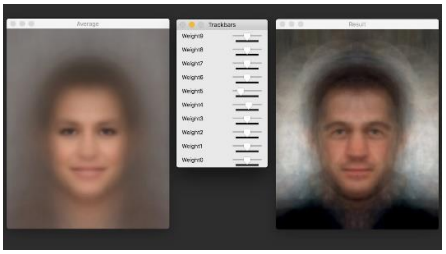

Fig. 2. Génération de visages grâce aux composantes principales.

Cependant, OpenFaceTracker a pour vocation d'être utilisé n'ayant que très peu de sujets connus dans la base de données, ce qui rend *Eigenface* inexploitable.

Une solution est de peupler la base de données de manière *générique* (i.e. qui généralise). Le site *thispersondoesnotexist.com* [6] génère à chaque rafraichissement de la page un visage totalement aléatoirement grâce à StyleGAN2 [7]. Les visages générés sont très réalistes (cf. Fig. 3), représentent toutes les ethnicités, sont soumises à diverses variations et ont très rarement des défauts.

Fig. 3. Echantillon de visages générés par StyleGAN2

Ainsi, le module de reconnaissance faciale d'OFT repose sur l'Analyse en Composantes Principales de 50 000 images de visages générés aléatoirement, mais qui offrent tant de variations qu'ils permettent de généraliser l'idée d'un visage humain. La méthode de projection et le calcul de distance reste identique à *Eigenface*.

Quelques avantages à l'utilisation de cette méthode :

- Une fois exécuté, un PCA est défini par trois matrices. Ces matrices, une fois sauvegardées dans un fichier, peuvent être réutilisées n'importe quand. Le calcul du PCA à 50 000 x 3600 valeurs a pris du temps, mais il n'est pas nécessaire de le recalculer.
- Lorsque l'on projette plusieurs images du même visage soumis à des variations, ces points forment un nuage dans l'espace vectoriel. Ainsi, le calcul de la moyenne des échantillons approxime le centre de ce nuage de points. De plus, il est possible de calculer la moyenne de manière incrémentale [8], ce qui veut dire que l'on peut mettre à jour le vecteur de données d'un sujet déjà connu.

A l'heure d'écriture de ce rapport, il n'est sauvegardé dans la base de données que l'approximation du centre du nuage de points d'un sujet. Cette méthode apporte des problèmes notamment concernant le calcul du seuil à partir duquel on considère un visage inconnu. On peut imaginer une prise en compte de la variance, qui est également calculable de manière incrémentale [8]*, pour approximer la taille de ce nuage de point pour permettre de calculer un seuil variable plutôt que de garder un seuil constant (ce qui est le cas dans la version v6.0.X).*

Lors de l'ajout ou de la mise à jour d'un sujet, c'est la moyenne des projections qui est sauvegardée dans la base de données avec le nombre d'échantillons prélevés pour le calculer.

(voir `FacialRecognition` *dans la documentation technique)*

IV. Conseils et bonnes pratiques

Dans cette section sont détaillés des conseils et des bonnes pratiques pour profiter au mieux des performances d'OFT.

A. Variation des données

Que cela soit au moment de choisir les sources pour enregistrer un nouveau sujet ou au moment de prélever les échantillons, il faut veiller à bien fournir la base de données d'échantillons variés et nombreux. Cela passe par l'utilisation de sources différentes, capturées dans des conditions différentes, la prise d'échantillons du visage dans diverses positions, avec diverses expressions, sous un éclairage variant…

Il faut garder en tête qu'il faut essayer de généraliser le visage d'un sujet pour maximiser les taux de reconnaissance du modèle.

B. Automatisation

Il faut faire attention lors de l'automatisation de l'ajout d'un sujet. A l'heure de la version 6.0.X, tous les visages détectés (même s'il y en a plusieurs en même temps) sont prélevés si l'utilisateur clique gauche ou l'option `--yes` est activée. Il est donc fortement recommandé de fournir une source ne comportant que le sujet à ajouter, ainsi que de vérifier que le module de détection faciale n'affiche pas trop de faux positifs. Dans la pratique, si moins de 5% des données sont erronées, les données seront jugées praticables.

C. Accepter les erreurs

OpenFaceTracker est encore en développement, et de nombreuses améliorations et idées sont encore à implémenter pour parfaire l'algorithme. Il arrive qu'une personne soit faussement identifiée. Dans ces cas-là, on pourra inspecter la confidence de la prédiction, ou comparer les résultats d'une image avec les images qui suivent (pour une vidéo) et ainsi analyser les résultats sur une période plutôt qu'à un temps *t* (cette dernière fonctionnalité sera implémentée nativement ultérieurement).

Conclusion

Une attention particulière a été apporté à ce projet quant à la rapidité d'exécution, les performances de reconnaissance et la facilité d'utilisation de la librairie. Les technologies utilisées ont été profondément étudiées et sont sans cesse en phase d'amélioration. En cas de problème, d'issue à rapporter, de retour ou d'idées à partager, veuillez m'en faire part par email.

Remerciements

Je souhaiterai remercier Richard REY, qui a eu l'idée du projet, en est le Project Manager, et qui a su me faire confiance pour le maintien de celui-ci.

References

[1] github.com/microsoft/vcpkg

[2] cmake manual - cmake-generators

[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.

[4] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Maui, HI, USA, 1991, pp. 586-591, doi: 10.1109/CVPR.1991.139758.

[5] Analyse en Composantes Principales

[6] thispersondoesnotexist.com

[7] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 8107-8116, doi: 10.1109/CVPR42600.2020.00813.

[8] Calcul incrémental de la moyenne et de la variance