Rapport de soutenance

Quentin GALBEZ – Maelys SAM YOU
Francisco LARA RICO – Matisse BAP
Promo EPITA 2026
Classe B1
Groupe EOS

15 février 2023

Projet S4 : EOS

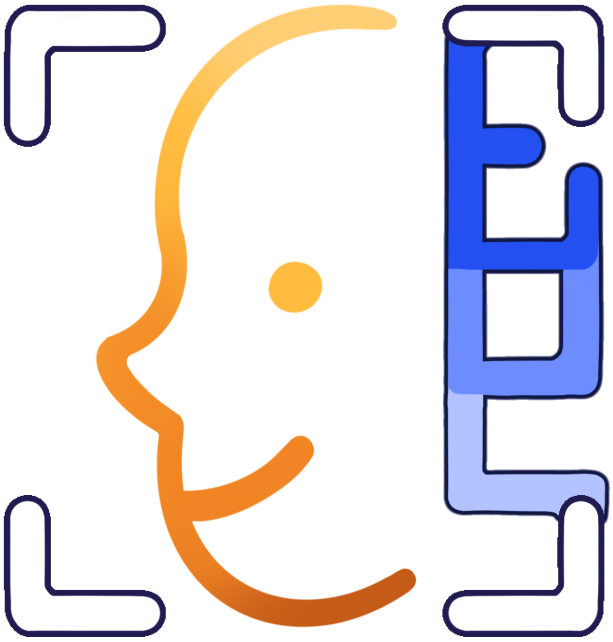

Table des matières

1 Introduction

Dans ce rapport, nous vous expliquerons ce que nous avons pu réaliser au cours de ce mois, nos problèmes rencontrés et notre avancement global.

1.1 Présentation du Groupe

Le groupe s'appelle EOS pour Evolution of System. Tous les membres ont déjà eu l'occasion de travailler ensemble. En effet, Maelys, Quentin et Matisse étaient dans le même groupe de projet de S3 et Francisco a été d'une aide pour des TPs de programmation.
Chaque membre de ce groupe a ses difficultés et ses qualités qu'ils mettront à profit lors de ce projet. Il y a une bonne cohésion et tous les membres de l'équipe feront en sorte que cette cohesion persiste tout au long du semestre.

1.2 Présentation des membres

1.2.1 Quentin

Lors de ce projet, je serai en charge du groupe et de l'interface, mais j'aiderai grandement Matisse pour le réseau neuronal. Lors du projet S3, j'étais chargé de l'interface c'est pourquoi lors de ce projet je pourrai plus appronfondir mes connaissances sur ce sujet, à l'opposé je n'ai jamais étudié les réseaux neuronaux. Cela sera pour moi une occasion d'apprendre et d'élagir ma palette de connaissance de l'informatique. Néanmoins je serai toujours disponible pour aider les membres de l'équipe si nous rencontrons des difficultés lors de ce projet.

J'ai confiance en mon équipe pour réaliser ce projet, je sais que nous avons une bonne cohésion et que l'entraide sera présente lors de ce projet.

1.2.2 Maelys

Pour ce projet, je serai responsable du traitement d'image pour la détection des expressions faciales, que je réaliserai aussi à l'aide de Francisco. Par la suite, j'aiderai Quentin à coder l'interface de notre application.

Lors de la recherche du sujet pour le projet, l'idée d'un programme reconnaissant les expressions faciales m'est venue et m'a beaucoup intriguée. Bien que cela ait déjà été réalisé, il est intéressant de se pencher sur ce sujet pour savoir comment cela fonctionne. Le projet de S4 est donc une opportunité pour en apprendre d'autant plus et de relever un nouveau défi à l'aide de mes trois autres partenaires.

J'espère pouvoir mener à bien ce projet et que l'on puisse produire une application dont nous serons fiers !

1.2.3 Francisco

Avec sincérité, je ne pense pas du tout avoir bien travaillé pour le projet du deuxième semestre, je n'ai pas dédié suffisamment d'heures à ce travail. J'ai l'intention d'évoluer et de travailler comme il le faut pour ce projet de reconnaissance d'expressions faciales. Je pense que le plus grand apprentissage que je pourrai tirer de cette expérience est d'apprendre à s'organiser et à bien travailler en équipe, de la même façon que pour le projet du troisième semestre.

D'ailleurs, je suis assez content de mon rendement lors du projet S3. Bien que j'avançais peu à peu au début, vers la fin j'ai bien travaillé et bien communiqué avec mes camarades. Même

si le résultat de tout notre travail n'était pas ce que je m'attendais, ce n'est pas le résultat du travail ce que je vais retenir ou apprendre de ce projet, mais plutôt l'expérience de travailler en équipe, de diriger une équipe, de souffrir le stress et de continuer le travail malgré que les attentes ne s'accomplissent pas. Le fait d'avoir vécu cette expérience va être très utile pour ce nouveau projet, selon moi, mais surtout, pour ma carrière comme ingénieur.

Ensuite, en ce qui concerne mon champ de travail, je m'occupe principalement de la détection de l'expression faciale dans une image procéssée au préalable. Pour cela il va falloir une bonne communication avec les personnes chargées du traitement de l'image.

1.2.4 Matisse

Je suis Matisse Bap et j'assumerai la responsabilité de la mise en place et de la gestion du réseau neuronal ainsi que du site internet pour ce projet. J'ai un grand intérêt pour notre sujet de recherche, en particulier pour la partie liée au réseau neuronal que j'ai découverte lors du projet S3. Je suis motivé à poursuivre mes connaissances et mes compétences dans ce domaine pendant le projet S4.

En outre, je serai en charge de la gestion du site internet pour les différentes soutenances, ce qui convient parfaitement à mes compétences car je sais que c'est une tâche moins lourde, ce qui me permettra de me concentrer davantage sur le réseau neuronal.
Enfin, j'espère que nous allons atteindre tous les objectifs que nous nous sommes fixés, voire même les objectifs supplémentaires. Je suis confiant en les compétences de mon équipe et j'estime qu'en nous donnant à fond, nous pourrons obtenir des résultats très satisfaisants.

2 Traitement d'image (Maelys)

Avant d'effectuer la détection et reconnaissance de l'expression, il faut traiter l'image en entrée pour qu'elle puisse être lisible par le programme. Pour la première soutenance, un traitement simple de l'image est disponible. Ce traitement comprend les étapes suivantes : mise en gris de l'image, accentuer le contraste si besoin et enlever les bruits.

2.1 Mise en gris

Une image est grise lorsque ses composantes RGB ont la même valeur. Un moyen simple d'effectuer la fonction grayscale aurait été de faire la moyenne des trois composantes puis d'appliquer cette moyenne aux composantes du pixel. Cependant, il s'agit d'un calcul assez simple qui ne donne pas un résultat satisfaisant car l'image obtenue ne représente pas toutes les nuances de gris. Un autre calcul tout aussi simple est l'expression suivante :

Gray = (Red*0.3 + Green*0.59 + Blue*0.11)

Une fois la valeur de gris calculée, il suffit de l'appliquer aux trois composantes du nouveau pixel. Ainsi, à partir de l'image initiale, la fonction applique les changements de pixels sur la nouvelle surface. Cela permet de faire une « copie » de l'image initiale sans la modifier. La nouvelle surface créée servira pour les fonctions suivantes du prétraitement.

2.2 Contraste

La fonction contraste prend en paramètre deux SDLSurface : un pour l'image avant traitement, et un étant l'image résultat. Il y a aussi la valeur de contraste étant un int. Par la suite, on convertit cette valeur en double pour effectuer les prochains calculs. Tout d'abord, une constante est calculée grâce à la valeur de contraste suivant cette formule :
Factor = 259*(255+contrastvalue)) / (255*(259-contrastvalue)
Ensuite, on parcourt l'image et on modifie chaque pixel en changeant ses trois composantes de couleurs :
NewR = trunc(factor*(r-128)+128) ;
NewG = trunc(factor*(g-128)+128) ;
NewB = trunc(factor*(b-128)+128) ;

La function trunc() vient de la librairie math.h et permet de garder les nouvelles valeurs entre 0 et 255. Lors de l'appel de la fonction, la valeur 128 est utilisée comme valeur de contraste puisqu'elle permet d'avoir un contraste suffisant et non excessif.
Il y a aussi une fonction auxiliaire qui détermine si l'image nécessite d'être contrastée.
Le contraste est la différence entre les parties claires et foncées. Ainsi, une image qui a besoin d'être contrastée a peu de différence entre ces parties.
La fonction determineContrast prend en paramètre une SDLSurface. Il y a deux valeurs min et max qui vont permettre de savoir la valeur de luminosité minimal et maximal dans l'image. Elle renvoie une valeur de contraste. Pour chaque pixel, on calcule sa valeur de luminosité grâce à ses composantes de couleurs :

Int brightness = 0.2126*r + 0.7152*g + 0.0722*b

Si cette valeur de luminosité est inférieure à min, alors min prend sa valeur. Sinon, si elle est supérieure à max, max prend sa valeur. À la fin, la valeur de contraste est le quotient de 255 par la différence de max et min. Une image ne nécessitant pas de contraste aura une valeur de contraste de 1, et dans le cas contraire, sa valeur de contraste sera supérieure à 1.

2.3 Réduction du bruit

Maelys a implémenté une méthode parmi de nombreuses pour la réduction de bruits. Il s'agit de l'algorithme median.
La première fonction prend en paramètre une SDLSurface. Chaque pixel est remplacé par le médian de tous ses pixels voisins. Pour ce faire, un tableau de 9 a été utilisé pour stocker les valeurs des pixels voisins. Ensuite, il fallait trier ce tableau par ordre croissant et récupérer la valeur à la 4ieme position à partir de 0, car il s'agit du médian. On applique enfin cette valeur au pixel actuel et ainsi de suite. C'est un algorithme très efficace contre les bruits de type poivre et sel, mais n'est pas forcément convainquant pour les « pires » photos.
Une autre fonction a été implémentée avec un raisonnement similaire. Dans la deuxième fonction, on récupère cette fois-ci la moyenne des pixels voisins. Par un même procédé, on additionne les valeurs et à la fin, on divise le résultat par le nombre de pixels.
La fonction utilisée dans le programme est la deuxième. Bien qu'elle ne soit pas parfaite, elle élimine suffisamment de pixels pour obtenir une image satisfaisante.

2.4 Conclusion du traitement d'image

Le traitement d'image actuel est d'abord suffisant pour la détection du visage. Bien évidemment, il sera complété et amélioré au cours du projet, dépendant des besoins de chaque partie. La fonction la plus importante pour l'instant est la mise en gris puisque la détection du visage se fait à partir d'une image grise. Les ajouts possibles sont des fonctions plus complexes pour enlever différents types de bruits.

3 Détection du visage (Maelys)

La détection du visage était plus compliquée que prévu. Maelys avait réalisé des recherches avant de commencer le projet et s'est rendu compte que la bibliothèque OpenCV permettait d'effectuer des reconnaissances d'objets, notamment de visages. Elle est disponible pour les langages C++, Java et Python. Cependant, cette bibliothèque avait été originellement créée pour le C, donc Maelys a pu utiliser le C API.

3.1 Installation OpenCV

Le plus gros problème que Maelys a rencontré est l'installation et la configuration de OpenCV. En effet, OpenCV est une librairie open-source pour faire de la reconnaissance d'objets, ici, de visages. L'avantage de cette librairie est d'être dans un premier temps open-source et donc très documentée et régulièrement mise à jour, mais également d'avoir énormément de fonctionnalités. Le problème c'est que cette librairie, étant originellement créée pour fonctionner avec des programmes en C, a évolué avec le temps et s'est implémenté davantage aux langages Python et C++, au détriment du langage C. OpenCV est donc divisé en plusieurs versions : 4.0 et 3.0 (anciennement les version 1.0 et 2.0). La version 3.0, anciennement les versions antérieures 1.0 et 2.0, est majoritairement faite pour le C et peu pour le C++. Les versions antérieures étaient en revanche faites pour le C uniquement. La version 4.0 est en revanche faite pour les versions

plus récentes comme Python et C++. Cette version propose bien plus de fonctionnalités du fait de la jeunesse de ces langages par rapport au C. C'est pour cette raison que cette version est majoritairement utilisée depuis des années.
Maelys avait fait l'erreur d'installer la version 4.7.0 alors qu'elle est faite pour le C++ majoritairement et pour Python. Elle n'avait donc aucune possibilité d'utiliser cette librairie malgré ses recherches pour récupérer les fichiers en C de ces fichiers. Elle a donc installé une autre version 4.0, mais le résultat était sensiblement identique pour les mêmes raisons. La solution était de revenir aux versions 3.0.
En réalité, ces versions ont l'air d'être très anciennes mais il faut bien comprendre que la librairie est open-source, ce qui permet aux deux versions d'être en développement simultanément. La version 3.4.16 que nous allons utiliser dans la suite du projet est sortie le 10 octobre 2021. Ce qui est relativement récent par rapport à l'âge de la librairie.
En résumé, nous allons utiliser la version la plus récente possible compatible avec le langage C pour nous permettre d'avoir les dernières fonctionnalités. L'équipe a donc pu utiliser les fonctions nécessaires au développement de l'application.

3.2 Face_detection.c

Le deuxième problème a été de comprendre ce qu'était un classifier, et comment le réaliser. Un classifier se rapproche d'un réseau neuronal, et va permettre la détection de l'objet souhaité. À partir d'un classifier entrainé, la détection est très simple. Pour créer un classifier, il faut d'abord générer les caractéristiques pseudo-Haar. Ce sont des caractéristiques utilisées en vision par ordinateur pour la détection d'objet dans des images numériques. Il existe plusieurs façons de générer ces caractéristiques, notamment via la méthode de Viola et Jones, qui est celle utilisée dans ce projet. Ensuite il faut choisir le meilleur set de caractéristiques de pseudo-Haar pour la détection de visages, puis trouver plusieurs exemples d'images positives (avec visage) et négatives (sans visage) pour entrainer le classifier. Pour l'entrainer, on peut utiliser Adaboost, un algorithme qui combine plusieurs classifiers « faibles » en un classifier « fort » en sélectionnant les meilleurs caractéristiques de pseudo-Haar. Finalement, il faut vérifier la précision du classifier.
Par chance, Maelys a pu trouver des classifiers déjà entrainés sur des visages.
Par la suite, il a fallu comprendre les fonctions disponibles en C. Les fonctions en C et en C++ ne sont pas les mêmes et il existe bien plus de fonctions dans les nouvelles versions. Par conséquent, il a fallu s'adapter avec les fonctions disponibles et comprendre leurs documentations qui était très pauvres.
La librairie utilisée pour la détection des visages est objdetect.h. Elle contient les fonctions suivantes : cvLoadHaarClassifierCascade et cvHaarDetectObjects.
Tout d'abord cvLoadHaarClassifierCascade permet de charger un classifier. Il prend en paramètre le path vers le classifier en .XML. Si le classifier n'a pas pu être chargé, la fonction renvoie NULL. En effet, c'était un autre des problèmes que Maelys a rencontré. Le classifier utilisé au début n'était pas adapté avec la version d'OpenCV qu'elle utilisait, et donc ne pouvait être chargé. Après plusieurs recherches pour trouver une solution, elle a trouvé un bon classifier.
Ensuite, cvHaarDetectObjects est la fonction qui utilise le classifier chargé pour détecter l'objet voulu sur l'image donnée en paramètre. Le plus compliqué avec cette fonction était de comprendre les valeurs en paramètre, puisque c'était des valeurs de types propres à OpenCV, notamment « CvMemStorage» et « CvSeq ». Tous les types spécifiques et « basiques » de OpenCV sont situés dans la librairie « types_c.h» et « core_c.h». De manière simple, CvMemStorage est un pointer vers le stockage mémoire d'un objet, et CvSeq est une structure de donnée utilisée pour représenter des séquences d'objets. La fonction renvoie donc un CvSeq contenant les (ou dans le cas du projet le) visage(s) détecté(s).
Une fois cette étape faite, il faut récupérer le visage détecté. Pour cela, il suffit de récupérer le

premier (et unique) élément de la séquence renvoyée par cvHaarDetectObjects et de le cast en CvRect*. CvRect est un objet propre en OpenCV afin de représenter des rectangles. Ainsi, on obtient un pointer vers un rectangle.
Finalement pour tracer le rectangle de la détection, on sauvegarde dans des variables les coordonnées et les dimensions du rectangle grâce à la classe CvRect, et on trace le rectangle à l'aide de ces variables, et de la fonction cvRectangle.
Pour éviter les memory leaks ou erreur de segmentation, on libère tout l'espace mémoire alloué grâce à cvClearSeq, cvReleaseHaarClassifierCascade et cvReleaseMemStorage. Enfin, on sauvegarde l'image résultat sous le nom de « face_detected.bmp» grâce à cvSaveImage, inclut dans les librairies « imgcodecs_c.h» et «imgproc_c.h ».

3.3 Main.c

Le main est assez basique. Il prend en paramètre le path vers l'image où l'on souhaite détecter le visage. Il charge l'image, appelle la fonction DetectFace pour la détection et enfin libère l'espace mémoire occupé.
Pour charger l'image, on utilise cvLoadImage, qui prend en paramètre le path de l'image, et un entier indiquant de quelle couleur est l'image en entrée (0 pour le grayscale). Cette fonction renvoie un pointer de type IplImage, qui est le type propre à OpenCV pour les images. Maelys l'a ensuite cast en CvArr*, qui est un «sous-type » de IplImage, et qui permet d'avoir l'image sous forme de tableaux ou de matrices.
Puis, après la détection du visage, il ne restait plus qu'à libérer l'espace avec cvReleaseImage.

3.4 Conclusion de la détection du visage

La prochaine étape pour finir complètement la partie détection du visage est d'implémenter la fonction qui pourra « découper » l'image de sorte à n'avoir que le visage détecté, et non la photo entière. À voir avec le reste du groupe pour savoir si l'étape est nécessaire ou non.

FIGURE 1 – Image avant détection

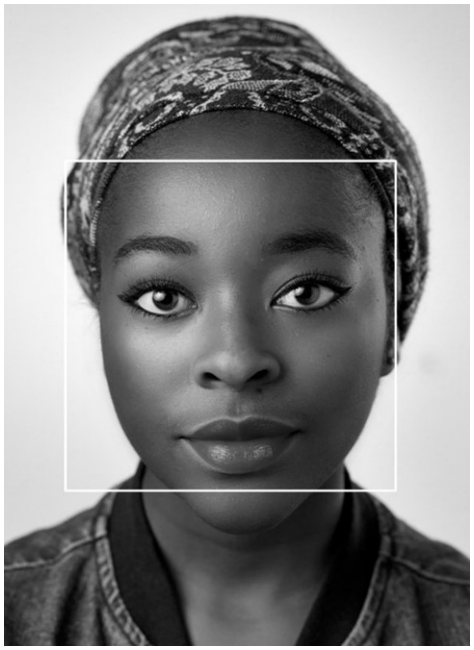

FIGURE 2 – Image après détection

4 Détection des traits du visage (Francisco)

4.1 Idée principale

Francisco est le principal responsable de cette partie, il doit donc partir du visage detecté par la partie de Maé et communiquer le résultat au réseau neuronal qu'est la partie de Matisse.

Pour l'instant Francisco a réfléchi à la détection de l'expression faciale et un peu à l'implémentation du réseau neuronal, ce qui lui concerne quand même, vu qu'il doit savoir ce qu'il doit renvoyer à Matisse. La première idée que nous avons eu pour détecter l'expression faciale derrière un visage est de traiter l'image pour obtenir une image en noir et blanc qui corresponde uniquement aux traits du visage comme les pliages de la peau et la forme des yeux, de la bouche et des sourcils (voir Figure 4).

De cette façon on aurait une image en noir et blanc ne contenant que le strictement nécessaire pour déduire l'expression faciale d'un visage humain, qu'on pourra donner en argument au réseau neuronal, comme pour le projet du troisième semestre.

4.2 Comment s'y prendre

Pour la détection des différentes parties du visage, nécessaire pour en éliminer les éléments superflus, pourrait se faire grâce à la librairie OpenCV, comme pour la détection du visage. Pour mettre l'image originale en noir et blanc, ainsi qu'enlever la plupart du bruit de l'image, nous pouvons utiliser les fonctions déjà codées pour le projet OCR du S3.

Figure 3 – Image originale

FIGURE 4 – Idée d'image après détection de visage et traitement des traits

Tout de même, nous avons nos doutes par rapport à la détection des expressions faciales et à l'implémentation du RN : ne seront peut-être toutes les images du dataset trop similaires, vu que c'est toutes des visages avec deux yeux, une bouche et deux sourcils ? Nous allons continuer de faire des recherches et d'y réfléchir pour mener à bien ce projet.

4.3 Avancement

De plus, Francisco a commencé à coder quelques fonctions pour détecter quelques formes géométriques dans une image, par exemple une fonction qui détecte s'il y a un cercle de centre C et de rayon R dans l'image donnée au point de coordonées X et Y :

Cette fonction qui détecte des cercles pourrait être appelée plusieurs fois sur la zone la plus probable d'apparition des yeux -c'est-à-dire, le deuxième quart de l'image en partant du haut- pour détecter les pupilles, ce qui pourrait être important pour détecter les yeux. Cette fonction détecte les pixels noirs à une distance R du centre C et calcule la proportion PIXELS_NOIRS/TOTAL_PIXELS, si ce rapport est supérieur à 70%, on considère qu'il y a un cercle de centre C de coordonnées X,Y et de rayon R.

De toutes façons, il pense commencer à apprendre à utiliser OpenCV pour s'en servir pour la détection de certaines parties du visage, comme les yeux ou le nez.

4.4 Conclusion sur la détection des traits

Pour cette soutenance Francisco est un peu en retard, mais il compte avoir un meilleur rendement dorénavant. Il n'est pas content avec ce qu'il a fait jusque là, mais ça lui arrive souvent aux projets et il est habitué. Il reste motivé pour travailler avec ses camarades par la suite, de toutes façons il finit toujours par se rattraper.

5 Conclusion

Cette période avant la première soutenance est surtout marquée par beaucoup de recherches afin de savoir comment s'organiser ou avancer. Nous avons encore besoin de réflexion et de doutes à dissiper en ce qui concerne le réseau neuronal et la reconnaissance des expressions faciales. Malgré cela, la détection du visage a pu nous redonner confiance et nous sommes plus optimistes pour le bon avancement du projet.
En ce qui concerne le traitement d'image et la détection, nous respectons le planning fixé (qui était de 40% pour les deux, pour la première soutenance).