



ÉCOLE
CENTRALELYON

ÉCOLE CENTRALE LYON

UE PRO
RAPPORT

Suivi du regard d'un nourrisson pour des évolutions du développement cognitif

Élèves :
Knani MELEK
Mezzi MOHAMED

Enseignant :
Stéphane DERRODE
Davy DALMAS
Jean-Remy HOCHMAN

2 avril 2024

Table des matières

1	Introduction	3
2	Présentation générale du projet	4
2.1	Introduction	4
2.2	Cadre de projet	4
2.3	Présentation de l'entreprise	4
2.4	Problematique du projet	5
2.5	Étude de l'existant	5
2.5.1	Description de l'existant	5
2.5.2	Critique de l'existant	6
2.5.3	Solution préconisée	7
2.6	Méthodologie de développement adoptée	8
2.7	Etude comparative des Méthodologies	8
2.8	Selection du processus de développement Agile	8
2.9	Conclusion	9
3	Etat de l'art	10
3.1	Introduction	10
3.2	Etude comparative des outils disponibles pour la détection des yeux	10
3.2.1	Cascade de Haar	10
3.2.2	MediaPipe	11
3.2.3	DLib	12
3.2.4	OpenFace	12
3.3	Techniques pour le suivi du regard	13
3.3.1	Suivi du regard basé sur des techniques informatiques	13
3.3.2	OpenFace	13
3.3.3	iCatcher+	14
3.4	Conclusion	15
4	Réalisation	16
4.1	Introduction	16
4.2	Environnements de travail	16
4.2.1	Poste de travail	16
4.2.2	Environnement de développement logiciel	16
4.3	Approche de détection et de suivi de la pupille pour les nourrissons	17
4.3.1	Introduction	17
4.3.2	Détection de la pupille	17
4.3.3	Calibrage	17
4.3.4	Projection sur écran avec visage fixe lors de l'expérimentation	20
4.3.5	Adaptation du calibrage et projection sur l'écran	22
4.3.6	Conclusion	23
4.4	Approche de suivi de regard en espace tridimensionnel	24
4.4.1	Introduction	24
4.4.2	Pipline	24
4.4.3	Détection des Points importants du visage	24
4.4.4	Calcul du score de regard	25

4.4.5	Calcul de la matrice de rotation de la caméra	26
4.4.6	Projection des axes du regard	28
4.4.7	Ajustement des axes en fonction du score corrigé	29
4.5	Conclusion	29
4.6	Approche de suivi du regard en utilisant WebGazer	30
4.7	Conclusion	31
5	Conclusion et perspectives	32

1 Introduction

Le présent projet représente une collaboration entre l'équipe Imagine du Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS) et le Babylab de l'Institut des Sciences Cognitives Marc Jeannerod, localisé à Bron. Le Babylab se consacre à l'étude du développement cognitif des nourrissons âgés de 3 à 18 mois, en utilisant la technique d'oculométrie basée sur le spectre infrarouge. L'oculométrie est une méthode qui observe et mesure les mouvements des yeux. Elle permet de comprendre comment les gens regardent et traitent l'information visuelle. Afin de diversifier les possibilités de recrutement pour leurs expériences, le laboratoire aspire à mettre en place des outils de tests en ligne et à distance.

Notre rôle dans ce projet se focalise sur deux objectifs principaux. Premièrement, nous recensons et évaluons les outils existants utilisés pour l'oculométrie, en examinant leur fonctionnement (comme les modèles d'œil, les réseaux de neurones profonds, etc.) et en évaluant leurs performances (précision, mesure du diamètre de la pupille). Deuxièmement, nous travaillons sur le développement d'un algorithme d'oculométrie utilisant des images de webcams standard (type PC) dans le domaine visible. Cette démarche vise à étendre les possibilités d'expérimentation à distance, en utilisant des outils largement accessibles, tout en garantissant des résultats suffisamment précis pour l'étude du développement cognitif des nourrissons.

Le premier chapitre comprendra une description de l'organisme hôte, le cadre du projet. Nous aborderons également un aperçu des travaux de recherche existants, en passant ensuite à l'explication de la méthodologie que nous avons adoptée pour la gestion du projet.

Le second chapitre présentera les différents outils et bibliothèques que nous pouvons exploiter.

Au cours du troisième chapitre, nous présenterons une description détaillée des étapes de développement, des choix technologiques et des différentes approches explorées.

Le présent rapport s'achève sur une conclusion générale et nous proposons quelques perspectives pour le présent travail.

2 Présentation générale du projet

2.1 Introduction

La réussite d'un projet dépend incontestablement de la réalisation d'une étude préliminaire approfondie, visant à établir des bases théoriques solides et à définir un cadre bien fondé pour atteindre les objectifs fixés. Dans le cadre de la mise en œuvre de ce projet, une approche méthodique a été adoptée. Celle-ci consiste en une analyse critique approfondie des travaux existants. Cette démarche permettra d'identifier de manière précise les problématiques à traiter, étayées par une méthodologie claire et un calendrier précis, organisant ainsi de manière rigoureuse les différentes étapes du travail à accomplir.

2.2 Cadre de projet

Ce projet de stage marque notre deuxième année à l'École Centrale de Lyon, dans le cadre du module **Projet d'Application et de Recherche**. La réalisation de ce projet vise à mettre en pratique et à consolider les enseignements reçus en génie logiciel, en programmation et en conception. De plus, il a pour objectif l'acquisition de nouvelles compétences et connaissances en explorant de nouvelles technologies.

2.3 Présentation de l'entreprise

Babylab, laboratoire de développement cognitif infantile, est une infrastructure de recherche spécialisée dans **l'étude scientifique du développement cognitif précoce des nourrissons et des jeunes enfants**. Ce labo est équipé de manière à faciliter la réalisation d'expériences et d'observations méthodiques visant à explorer divers aspects du développement cognitif chez cette population, notamment la perception, la compréhension du langage, la mémoire, et d'autres facettes liées à la pensée.

Au sein de notre projet, notre focalisation se dirige vers le domaine spécifique du Babylab dédié à la "Mémoire et apprentissage". Cela englobe l'exploration méthodique des capacités mémoriaires et des processus d'apprentissage chez les nourrissons, à travers l'utilisation de paradigmes expérimentaux soigneusement adaptés à leur stade de développement.

Pendant notre travail, nous avons eu l'occasion de visiter les locaux du Babylab. Les images ci-dessous vous donnent un aperçu de l'environnement que nous y avons trouvé.



(a) Salle obscurcie pour mener les expériences. À droite, le poste de contrôle de l'opération.



(b) Exemple d'expérience.



FIGURE 2 – Babylab [1].

2.4 Problématique du projet

Le Babylab se consacre à la création de paradigmes expérimentaux pour explorer le développement cognitif des nourrissons âgés de 3 à 18 mois. En utilisant la technique d'oculométrie (eye-tracking) basée sur le spectre infrarouge en laboratoire, le babylab cherche à élargir ses possibilités de recrutement de nourrissons pour des expériences à distance (via une visio depuis leur lieu de vie). Dans cette optique, le laboratoire souhaite développer des outils de tests en ligne et à distance et améliorer la recherche sur le développement cognitif des nourrissons en développant des outils de suivi du regard accessibles, précis et adaptés aux besoins du Babylab.

2.5 Étude de l'existant

2.5.1 Description de l'existant

Tobii :

Tobii est une entreprise suédoise spécialisée dans les technologies de suivi oculaire. Leur produit phare, le Tobii Eye Tracker, est un dispositif de suivi oculaire qui utilise des capteurs pour détecter et suivre les mouvements des yeux de l'utilisateur. Cela permet aux utilisateurs d'interagir avec des ordinateurs, des dispositifs électroniques et des interfaces utilisateur de manière plus naturelle, que par l'usage d'un clavier ou d'une souris.



FIGURE 3 – Tobii[2].

EyeTribe :

L'Eye Tribe Tracker est un dispositif commercial destiné à la gestion d'applications visuelles sur des ordinateurs de bureau et des tablettes. Ce module compact, se connecte facilement via un port USB 3.0. Une fois le logiciel nécessaire installé, **le système suit les mouvements oculaires de l'utilisateur, transformant ces actions en commandes de contrôle.** Cette technologie permet une interaction fluide et naturelle avec l'interface, offrant une expérience utilisateur innovante.



FIGURE 4 – The Eye Tribe [3].

2.5.2 Critique de l'existant

Coûts : Le Tobii Eye Tracker est souvent considéré comme relativement cher par rapport à d'autres dispositifs de suivi oculaire sur le marché. Cela peut être un facteur limitant pour les utilisateurs ou les entreprises avec des budgets restreints.

Installation : L'installation du Tobii Eye Tracker peut nécessiter une certaine expertise technique, et le processus peut être perçu comme complexe par certains utilisateurs. Cela peut entraîner des défis pour ceux qui ne sont pas familiers avec la configuration matérielle et logicielle.

Capteurs spécifiques : Le Tobii Eye Tracker utilise des capteurs spécifiques pour détecter les mouvements oculaires. Bien que cela garantisse une certaine précision, cela peut également signifier que le dispositif est moins flexible en termes de compatibilité avec d'autres capteurs.

Éclairage : Bien que le Tobii Eye Tracker soit conçu pour fonctionner dans des conditions d'éclairage variées, il peut tout de même être sensible à des niveaux de lumière extrêmes. Cela peut limiter son efficacité dans des environnements non standard.

Analyse des limites de Tobii : Une compréhension approfondie des limites spécifiques du Tobii Eye Tracker, telles que le coût élevé, la complexité d'installation, ou la nécessité de capteurs spécifiques.

2.5.3 Solution préconisée

Pour surmonter les limites associées au Tobii Eye Tracker qui est utilisé par le Babylab, notre solution consistera à explorer et à développer une alternative basée sur une technologie d'oculométrie utilisant des caméras standard de type PC. Cette approche visera à réduire les coûts et la complexité tout en élargissant l'accessibilité et l'applicabilité de la technologie de suivi oculaire. Les étapes possibles pour développer cette solution pourraient inclure :

- **Recherche sur les technologies alternatives** : Exploration des technologies d'oculométrie basées sur des caméras standard de type PC. Cela pourrait inclure l'examen d'algorithmes et de méthodes innovants qui permettent un suivi oculaire précis sans nécessiter des dispositifs spécialisés coûteux.
- **Développement d'algorithmes adéquats** : Concevoir et développer des algorithmes d'oculométrie robustes et précis qui peuvent fonctionner efficacement avec des images de webcams standard. Cela pourrait impliquer l'utilisation des méthodes avancées pour extraire des informations précises sur les mouvements oculaires.
- **Tests et validation** : Évaluer rigoureusement la performance de la nouvelle solution par rapport au Tobii Eye Tracker. Cela pourrait impliquer des tests comparatifs dans divers scénarios et conditions pour s'assurer que la solution développée offre des résultats comparables, voire meilleurs, tout en étant plus accessible. Il est à noter que cette étape n'a pas été complètement traitée dans ce travail.
- **Intégration dans des contextes spécifiques** : Adapter la solution développée pour répondre aux besoins spécifiques du suivi oculaire dans le cadre de l'étude du développement cognitif des nourrissons [10]. S'assurer que la solution peut être intégrée de manière transparente dans ces contextes d'utilisation spécifiques.

2.6 Méthodologie de développement adoptée

2.7 Etude comparative des Méthodologies

La sélection de la méthodologie appropriée pour notre projet représente une étape critique et déterminante dans la réalisation de nos objectifs avec succès. Par conséquent, nous devons accorder une attention particulièrement importante à l'évaluation approfondie des différentes options qui s'offrent à nous, afin de choisir celle qui convient le mieux à nos besoins spécifiques.

Méthodes	Avantages	Inconvénients
En cascade	Simplicité Modèle de prédiction Documentation précoce	Rigide Tests tardifs Gestion de projet complexe
En V	Meilleure gestion de la qualité Structurée Identification précoce des problèmes	Processus séquentiel Rigide face aux changements Tests tardifs
Agile	Livraison incrémentielle Adaptabilité et coopération Flexibilité	Difficulté à estimer les délais Complexité Documentation insuffisante
En CRISP	Adaptabilité à l'équipe Meilleure communication Flexibilité	Lourdeur Compliquée Limitée

TABLE 1 – Etude des différents modèles de développement.

2.8 Sélection du processus de développement Agile

Après une analyse approfondie, nous avons opté pour **la méthodologie Agile** en raison de sa capacité à offrir une structure **claire et flexible**, ainsi que pour sa capacité à s'adapter efficacement **aux besoins changeants du projet**. Son approche itérative et ses cycles de développement courts favoriseront la collaboration étroite, permettant ainsi une livraison continue de fonctionnalités à forte valeur ajoutée. De plus, la **transparence et la communication** accrues qu'elle encourage garantiront une gestion efficace des risques et une satisfaction optimale du client tout au long du processus de travail.

Progression du Sprint

Ci-dessous, un tableau illustrant les tâches effectuées par sprint durant le processus d'analyse et de développement de notre solution.

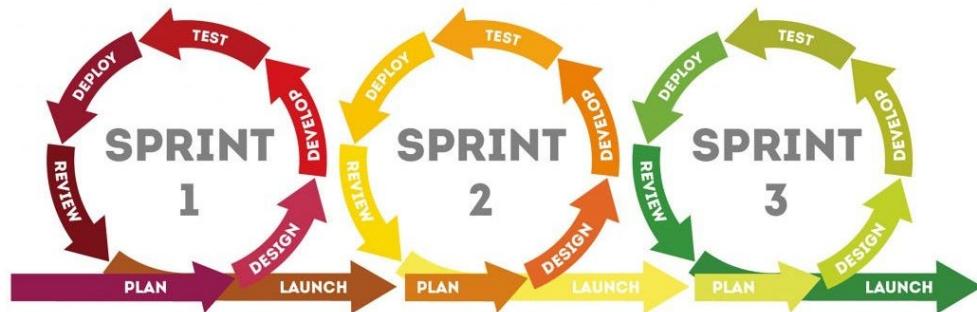


FIGURE 5 – Méthodologies agiles.

Récapitulatif des tâches accomplies par Sprint	
Sprint	Tâches effectuées
Sprint 1	<ul style="list-style-type: none"> — Revue de la littérature — Analyse documentaire — Analyse des besoins
Sprint 2	<ul style="list-style-type: none"> — Mise en place du pipeline général — Choix des technologies — Étude de faisabilité
Sprint 3	<ul style="list-style-type: none"> — Développement d'un algorithme de suivi de regard sur visage fixe — Rédaction d'une première version du rapport
Sprint 4	Adaptation du calibrage avec les mouvements de visage
Sprint 5	<ul style="list-style-type: none"> — Développement de deux autres approches de suivi du regard — Rédaction du rapport

2.9 Conclusion

En synthèse, ce chapitre a pris en considération la problématique du projet et a effectué une analyse approfondie de l'état actuel, mettant en lumière ses contraintes et ses insuffisances. De plus, une étude comparative des applications existantes a été menée pour identifier des solutions visant à améliorer la situation actuelle et à surmonter ses limites. Enfin, une évaluation approfondie des différentes méthodologies a été réalisée dans le but de sélectionner la stratégie la mieux adaptée à notre projet.

3 Etat de l'art

3.1 Introduction

Dans le cadre de notre projet, nous sommes confrontés à plusieurs outils et bibliothèques que nous pouvons exploiter. Cependant, il est important d'optimiser ce choix en fonction du cas d'utilisation spécifique. Dans ce chapitre, nous entreprendrons tout d'abord une étude comparative des méthodes de détection de la pupille, une étape essentielle dans le suivi du regard de l'enfant. Ensuite, nous présenterons différentes techniques pour le suivi du regard dans un espace tridimensionnel du nourrisson.

3.2 Etude comparative des outils disponibles pour la détection des yeux

3.2.1 Cascade de Haar

La méthode Haar Cascade est une technique de détection d'objets largement utilisée en vision par ordinateur, notamment implémentée dans des bibliothèques telles qu'OpenCV.

Comme l'illustre la figure ci-dessous, cette méthode [4] repose sur des caractéristiques visuelles appelées "**Cascades**" qui sont formées à partir d'exemples positifs et négatifs. Pour la détection des yeux, par exemple, Haar cascade est entraînée sur une grande variété d'images contenant des yeux (exemples positifs) et des régions sans yeux (exemples négatifs). Une fois la cascade formée, elle peut être utilisée pour détecter rapidement la présence d'yeux dans de nouvelles images en analysant différentes échelles et positions. Bien que la **méthode Haar Cascade** soit appréciée pour sa rapidité d'exécution, elle peut présenter des limitations en termes de précision, surtout comparée à des approches plus récentes basées sur l'apprentissage profond.

Néanmoins, elle reste une solution efficace pour des applications en temps réel où la latence est critique, comme la surveillance vidéo ou le suivi du regard.

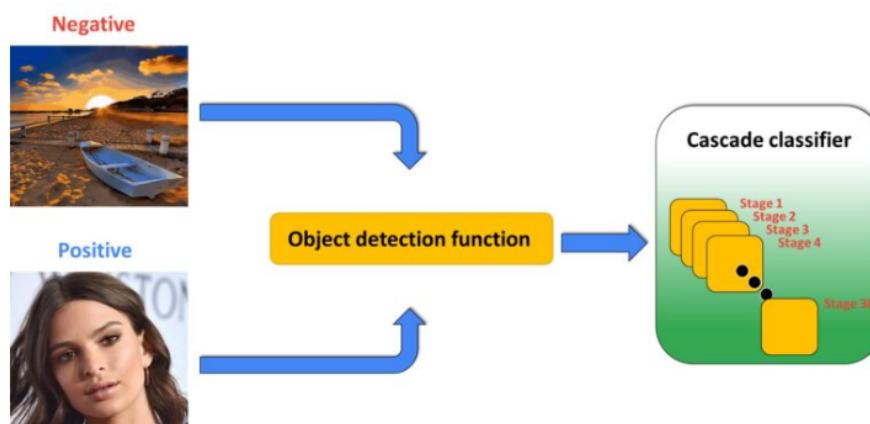


FIGURE 6 – Pipeline générale de la méthode Haar.

- **Points forts :**
 - Simple à implémenter et à comprendre.
 - Performances acceptables pour la détection de visages et d'autres objets dans des conditions d'éclairage contrôlées.
 - Faible exigence de puissance de calcul.
- **Points faibles :**
 - Moins précis que les approches basées sur les réseaux de neurones pour des cas d'utilisation complexes ou des environnements difficiles.
 - Limité dans sa capacité à détecter des variations de pose ou d'expression faciale.

3.2.2 MediaPipe

MediaPipe [5] est une bibliothèque open-source développée par Google qui offre une multitude de fonctionnalités de traitement d'image et de vision par ordinateur. En ce qui concerne la détection des yeux, **MediaPipe** propose un module spécifique qui utilise des techniques avancées telles que l'apprentissage profond pour identifier et suivre les yeux dans les flux vidéo en temps réel. L'un des principaux avantages de **MediaPipe** est sa facilité d'utilisation et son extensibilité, permettant aux développeurs d'intégrer facilement des fonctionnalités de suivi oculaire dans leurs applications sans avoir à concevoir des modèles de détection à partir de zéro.

De plus, **MediaPipe** est conçu pour fonctionner efficacement sur différents types de matériel, offrant ainsi une flexibilité d'implémentation.

Comme illustré ci-dessous, MediaPipe offre une méthode simple pour détecter une vaste gamme de points d'intérêt qui peuvent s'avérer utiles lors de l'étape de projection du regard sur l'écran.

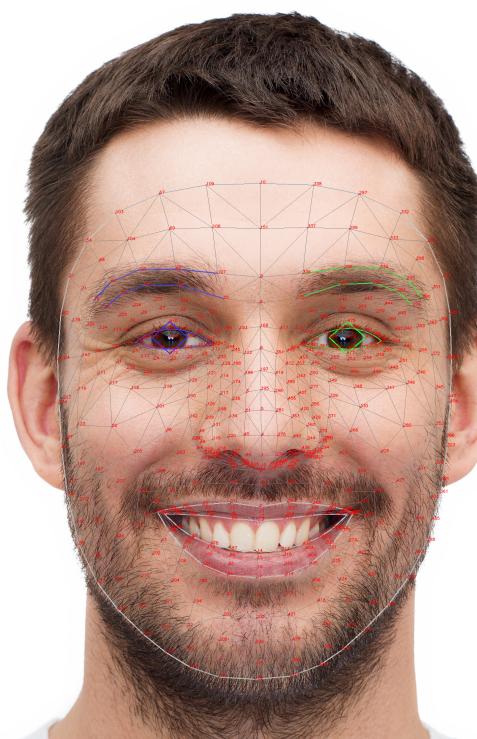


FIGURE 7 – Points d'intérêt possible à explorer avec MediaPipe.

— **Points forts :**

- Offre une grande variété de pipelines pour la vision par ordinateur, y compris la détection faciale, la reconnaissance de gestes, etc.
- Facile à utiliser avec des modèles pré-entraînés.
- Intégration avec TensorFlow pour la personnalisation et l'extension.
- Performances en temps réel pour de nombreux cas d'utilisation.

— **Points faibles :**

- Moins de flexibilité que d'autres méthodes pour les tâches personnalisées.
- Nécessite une certaine expertise pour une personnalisation avancée.

3.2.3 DLib

DLib [?] est une bibliothèque logicielle open source écrite en langage C++, offrant une gamme de fonctionnalités pour la vision par ordinateur, l'apprentissage automatique, et d'autres domaines connexes. Elle est largement utilisée dans la recherche et le développement de solutions liées à la reconnaissance faciale, à la détection d'objets, et à d'autres tâches de traitement d'images. DLib propose des implémentations efficaces d'algorithmes de pointe dans ces domaines, et fournit également des outils et des ressources pour la création de systèmes de vision par ordinateur robustes et performants.

— **Points forts :**

- Puissant pour la détection de visage et la reconnaissance faciale.
- Possibilité de formation personnalisée sur des ensembles de données spécifiques.
- Supporte plusieurs langages de programmation, y compris Python et C++.
- Dispose de fonctionnalités avancées telles que la détection de points de repère faciaux.

— **Points faibles :**

- Peut nécessiter une puissance de calcul élevée pour des tâches complexes.
- Courbe d'apprentissage abrupte pour les utilisateurs débutants.

3.2.4 OpenFace

OpenFace [6] est une bibliothèque logicielle open-source développée principalement en langage Python, offrant des outils avancés pour l'analyse et la reconnaissance faciale. Cette bibliothèque met en œuvre des techniques de pointe en vision par ordinateur et en apprentissage automatique pour détecter, localiser et identifier des visages dans des images et des vidéos. OpenFace propose également des fonctionnalités avancées telles que la reconnaissance d'expressions faciales, l'estimation des angles et des points clés du visage, ainsi que la mesure de similarité entre visages. Grâce à sa flexibilité, sa robustesse et sa précision, OpenFace est largement utilisée dans divers domaines, notamment la sécurité, la surveillance, la réalité virtuelle, et la recherche en sciences sociales.

- **Points forts :**
 - Robuste pour la reconnaissance faciale dans diverses conditions d'éclairage et de pose.
 - Fournit des embeddings faciaux permettant des tâches avancées telles que la vérification d'identité et la recherche de visages similaires.
 - Open-source et activement maintenue.
- **Points faibles :**
 - Peut nécessiter un grand nombre de données d'entraînement pour obtenir des performances optimales.
 - Exigeant en termes de puissance de calcul pour l'entraînement et l'inférence sur de grands ensembles de données.

3.3 Techniques pour le suivi du regard

3.3.1 Suivi du regard basé sur des techniques informatiques

Le **suivi du regard** basé sur des techniques informatiques [9] s'articule autour de plusieurs étapes clés pour fournir des informations précises sur le point de regard de l'utilisateur sur l'écran. Tout d'abord, la méthode repose sur la détection des yeux, où des algorithmes de traitement d'image identifient et suivent la position des yeux dans le champ visuel. Ensuite, afin d'obtenir des données de suivi précises et adaptées à chaque utilisateur, le processus de calibrage intervient. Pendant cette étape, l'utilisateur est généralement guidé à fixer des points spécifiques à l'écran, permettant au système de comprendre la relation entre les mouvements oculaires détectés et les positions réelles du regard sur l'écran.

Enfin, une fois le calibrage effectué, le système détermine une transformation adéquate pour projeter les données de suivi du regard dans le référentiel de l'écran. Cette transformation permet de convertir les coordonnées du regard dans l'espace en des coordonnées précises sur l'écran, offrant ainsi une mesure du point de regard de l'utilisateur.

3.3.2 OpenFace

OpenFace [6] excelle dans le **suivi du regard** en offrant une analyse approfondie de l'expression faciale et des mouvements oculaires dans l'espace, permettant ainsi de fournir des informations plus riches sur le comportement visuel, notamment la direction du regard dans l'espace tridimensionnel comme indiqué dans la figure ci-après. Cependant, il est important de noter que l'analyse des mouvements oculaires dans l'espace fournit une vue d'ensemble du comportement visuel de l'utilisateur. Pour obtenir des coordonnées précises du point de regard sur l'écran, des étapes supplémentaires, telles que la calibration, sont généralement nécessaires. Ainsi, bien que la **capacité de suivi spatial d'OpenFace** soit puissante, elle requiert une étape de post-traitement pour rendre les résultats applicables et interprétables dans le contexte spécifique de l'écran.

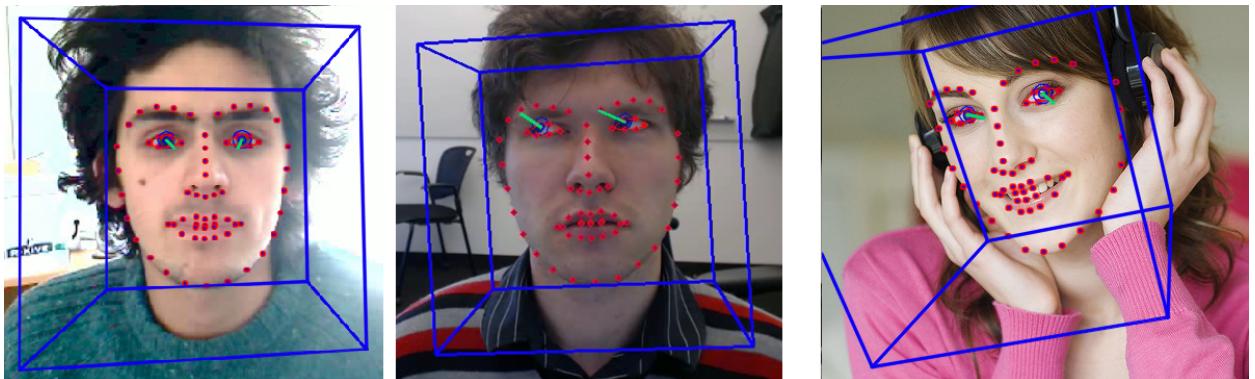


FIGURE 8 – Suivi du regard avec Openface.

3.3.3 iCatcher+

iCatcher+[8] est un système innovant qui se compose de trois parties principales : un détecteur de visage, un classificateur de visage et un classificateur de regard.

Le détecteur de visage identifie les zones probables du visage du participant, puis le classificateur de visage détermine si ces zones appartiennent à un nourrisson ou à un adulte. Enfin, le classificateur de regard est chargé d'estimer la direction du regard pour l'image centrale, en utilisant les cinq patchs sélectionnés avec leur taille et leur position dans le cadre.

iCatcher+ utilise une approche unique en traitant des séquences de cinq images consécutives, appelées "**Datapoints**". Chaque datapoint représente une fenêtre mobile d'observation, permettant au système de prendre en compte l'évolution du regard sur une courte période. Durant cette fenêtre de cinq images, l'image du milieu est spécifiquement ciblée pour la prédiction de la direction du regard (**GAUCHE**, **DROITE**, **LOIN**). Les quatre images adjacentes servent de contexte pour comprendre le mouvement visuel dans le temps. Cette méthodologie offre une compréhension dynamique du comportement visuel des participants, car elle capture les changements dans le regard au fil du temps plutôt que de se baser uniquement sur une image fixe. Ainsi, **iCatcher+** exploite ces datapoints pour enrichir sa capacité à anticiper et classifier les mouvements oculaires.

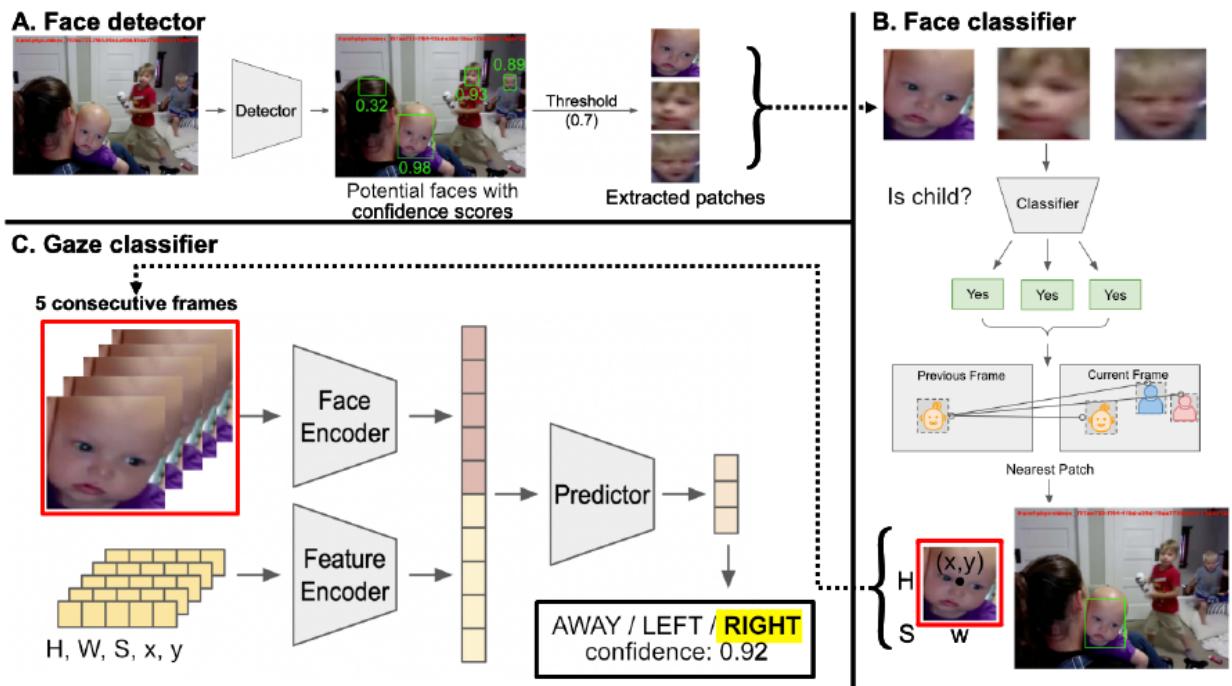


FIGURE 9 – Architecture de iCatcher+.

3.4 Conclusion

En conclusion de cette section, nous constatons la présence d'une variété d'outils et de bibliothèques potentiellement exploitables. Nous avons passé en revue les différentes alternatives, illustrant ainsi la richesse des choix disponibles pour notre projet. Cette diversité offre une base solide pour la prochaine phase de notre travail qui illustre la solution la plus adaptée à nos besoins spécifiques en matière de suivi du regard.

4 Réalisation

4.1 Introduction

Dans ce chapitre, nous allons plonger au cœur du processus de développement de notre solution, révélant ainsi les éléments clés de sa mise en œuvre. Vous aurez l'opportunité de suivre de manière approfondie les différentes phases de réalisation, depuis la sélection des technologies jusqu'à la concrétisation du projet. Au cours de ce chapitre, nous offrirons une description détaillée des étapes de développement, des choix technologiques, des différentes approches explorées, ainsi que des exemples de code et des captures d'écran pour une meilleure compréhension du fonctionnement de la solution.

4.2 Environnements de travail

4.2.1 Poste de travail

Poste n°1 : ASUS TUF A17

Processor : AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz

Graphics : NVIDIA GeForce RTX 3050 Laptop GPU GDDR6 @ 4GB (128 bits)

Memory : Total system memory :16GB

Poste n°2 : Lenovo l340-15irh

Processor : Intel i5 9300h

Graphics : NVIDIA GeForce GTX 1650

Memory : Total system memory :8GB

4.2.2 Environnement de développement logiciel

Python : Python un langage de programmation interprété largement utilisé dans le domaine du traitement d'images, qui offre une simplicité et une flexibilité remarquables. Des bibliothèques populaires telles que OpenCV [7] (Open Source Computer Vision Library) qui fournit des fonctionnalités avancées pour manipuler, analyser et traiter des images directement depuis Python.

4.3 Approche de détection et de suivi de la pupille pour les nourrissons

4.3.1 Introduction

Dans cette section, nous détaillerons le processus de suivi du regard du nouveau-né, mettant en lumière les différentes étapes de réalisation. Tel que représenté dans la figure, le processus démarre par la détection précise de la pupille. Ensuite, le processus exige la mise en œuvre d'une étape de calibrage. Une attention particulière doit être portée à l'adaptation au mouvement du visage, compte tenu de la fréquence et de la rapidité des mouvements chez les nourrissons. Enfin, une fois la pupille détectée, calibrée et adaptée au mouvement du visage, le système projette le point de regard estimé sur l'écran.

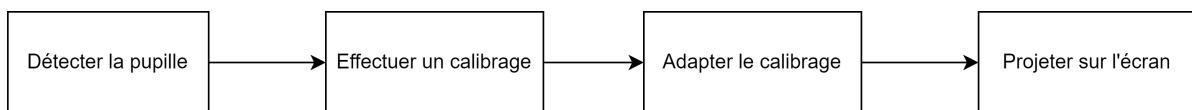


FIGURE 10 – Illustration des étapes clés du suivi du regard

4.3.2 Détection de la pupille

Le processus débute par l'initialisation des indices des landmarks, essentiels pour localiser les pupilles et les bords des yeux. Ces indices sont soigneusement sélectionnés pour garantir une identification précise des zones clés du visage.

La phase suivante consiste en la capture vidéo, où chaque image est convertie en format RGB pour optimiser les étapes de traitement. Les landmarks du visage sont ensuite extraits grâce à la bibliothèque MediaPipe FaceMesh, avec une attention particulière portée à la localisation des pupilles. Cette démarche permet d'obtenir des repères significatifs, utilisés ultérieurement pour calculer les centres des pupilles (cf figure 12).

Les coordonnées calculées des centres des pupilles sont ensuite exploitées pour dessiner des points distinctifs sur l'image vidéo. Cette visualisation des repères contribue à la compréhension et à la validation du suivi des pupilles.

Dans l'ensemble, cette séquence d'opérations constitue une première étape cruciale dans le développement d'un système de suivi des yeux, capitalisant sur les capacités de la bibliothèque MediaPipe FaceMesh pour extraire des landmarks précis du visage (cf figure 13).

4.3.3 Calibrage

Le processus de calibrage est essentiel pour garantir la précision du suivi du regard. Pour effectuer ce calibrage, l'utilisateur est guidé pour cliquer sur des points spécifiques dans un ordre défini, généralement en commençant par le coin supérieur gauche et en suivant le sens des aiguilles d'une montre (cf figure 14). Ces points servent de référence pour établir une relation spatiale entre les coordonnées des landmarks détectés et les dimensions réelles de l'écran.

Une fois que les points sont calibrés, les coordonnées calculées des centres des pupilles sont utilisées pour ajuster le suivi du regard en fonction de la position de l'utilisateur par rapport à l'écran. Cela permet d'obtenir une cartographie précise entre les mouvements



FIGURE 11 – Résultat de détection des points qui délimitent les yeux

```
# Define indices for iris and eye borders
LEFT_IRIS = [474, 475, 476, 477]
RIGHT_IRIS = [469, 470, 471, 472]
L_H_LEFT = [33]
L_H_RIGHT = [133]
R_H_LEFT = [362]
R_H_RIGHT = [263]
```

FIGURE 12 – Indices des landmarks essentiels pour localiser les pupilles

des yeux et les zones d'intérêt à l'écran, améliorant ainsi l'expérience utilisateur dans divers contextes d'application, tels que la navigation sur un écran ou le contrôle d'une interface utilisateur.



FIGURE 13 – Calibrage en cours.

4.3.4 Projection sur écran avec visage fixe lors de l'expérimentation

La projection des points sur l'écran, réalisée à l'aide du calibrage, repose sur une transformation géométrique précise. Cette transformation permet d'ajuster les coordonnées des points détectés en fonction des dimensions réelles de l'écran, assurant ainsi une précision optimale dans le positionnement des points projetés.

Pour ce faire, des facteurs multiplicatifs sont appliqués lors de la transformation afin de prendre en compte les tailles de l'écran. Ces facteurs sont calculés lors du processus de calibrage, où l'utilisateur est guidé pour marquer des points spécifiques sur l'écran. Ces points servent de référence pour établir une relation spatiale entre les coordonnées des points détectés et les dimensions réelles de l'écran.

Comme illustré dans la figure ci-dessous, nous définissons tout d'abord les distances minimales entre les points de référence obtenus lors du calibrage. Les facteurs de transformation sont alors déterminés comme le rapport entre les distances entre les points sur l'écran et les distances correspondantes des points de référence.

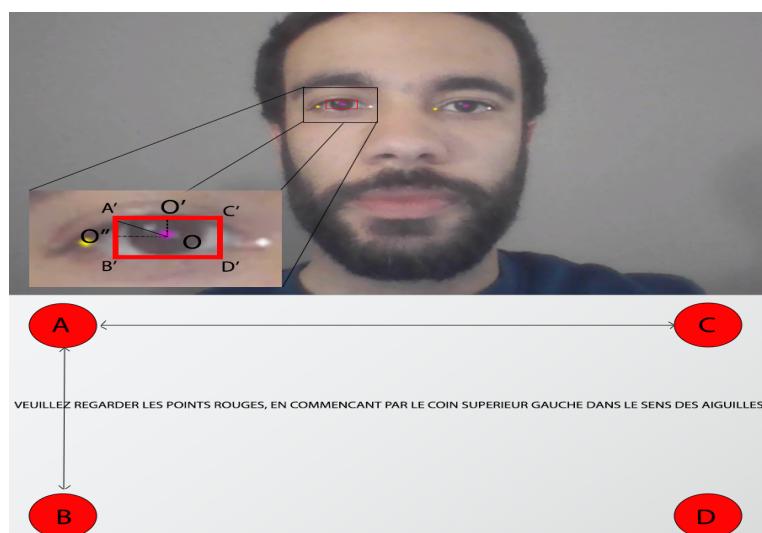


FIGURE 14 – Méthode de projection sur écran.

Pour obtenir les facteurs de calibration, nous calculons les distances moyennes selon les axes x et y :

$$\text{dist_moy_x} = \min(AC, BD)$$

$$\text{dist_moy_y} = \min(AB, DC)$$

Nous effectuons également ces calculs pour les points de référence oculaires :

$$\text{dist_moy_x_pup} = \min(A'C', B'D')$$

$$\text{dist_moy_y_pup} = \min(A'B', D'C')$$

Les facteurs de calibration sont alors obtenus comme suit :

$$\text{fact_x} = \frac{\text{dist_moy_x}}{\text{dist_moy_x_pup}}$$

$$\text{fact_y} = \frac{\text{dist_moy_y}}{\text{dist_moy_y_pup}}$$

Une fois que les facteurs de calibration sont déterminés, ils sont utilisés pour ajuster les coordonnées des points détectés, garantissant ainsi que les points projetés correspondent précisément aux positions souhaitées sur l'écran. L'estimation est donnée par :

$$x = OO' \times \text{fact_x}$$

$$y = OO'' \times \text{fact_y}$$

Cette approche permet d'obtenir une cartographie précise entre les mouvements des yeux et les zones d'intérêt à l'écran, améliorant ainsi l'expérience utilisateur dans divers contextes d'application, tels que la navigation sur un écran ou le contrôle d'une interface utilisateur.

Ci-dessous, une figure illustrant l'estimation du regard (disque en rouge) par l'algorithme décrit précédemment avec la webcam pour un visage fixe.

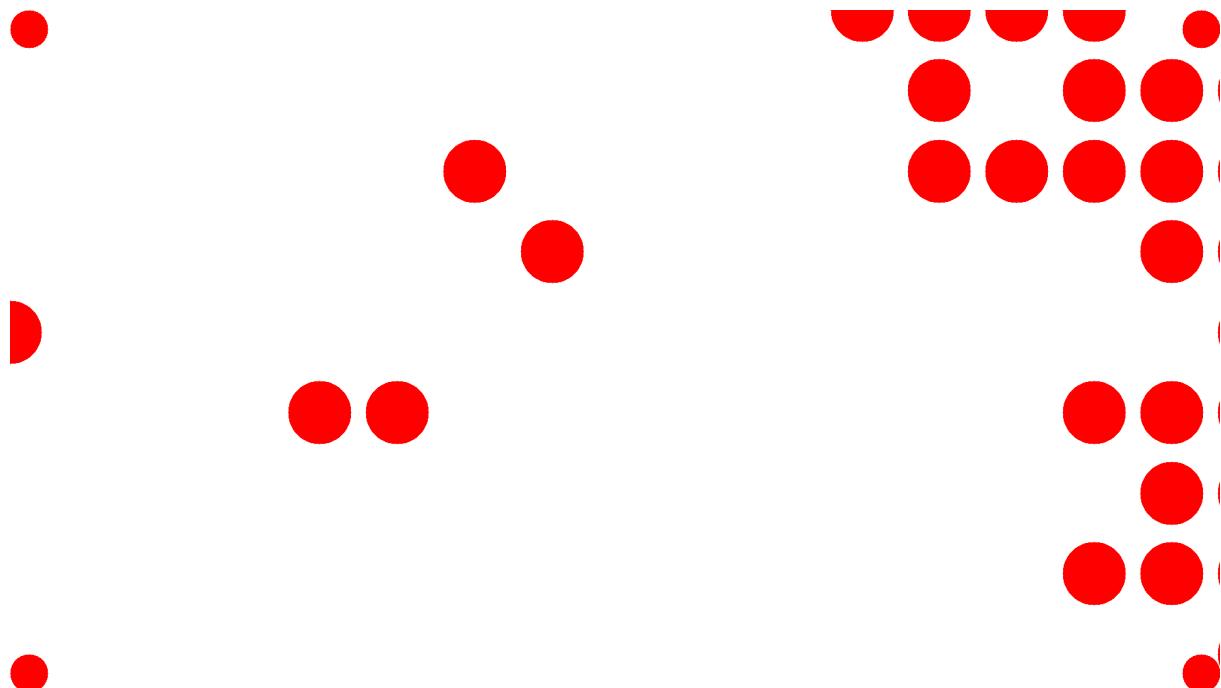


FIGURE 15 – La projection de la vision de l'utilisateur sur un écran blanc présentée par des cercles rouges.

4.3.5 Adaptation du calibrage et projection sur l'écran

Dans cette étape, l'objectif est d'adapter le calibrage aux mouvements du visage. Pour ce faire, on commence par extraire la partie du visage contenant les yeux en utilisant la bibliothèque MediaPipe, puis on applique une technique de traitement d'image appelée "transformation perspective". Ces deux processus combinés permettent de rendre les régions désirées du visage relativement indépendantes des mouvements du visage par rapport à la caméra comme le montre la figure ci-dessous.

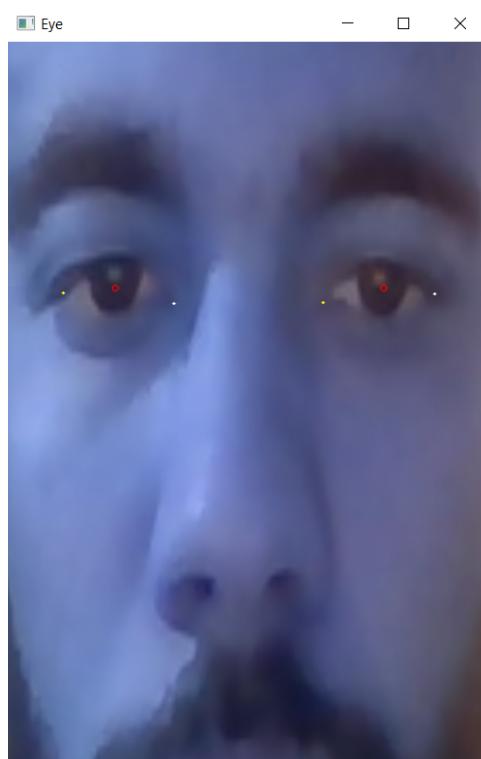


FIGURE 16 – Image de la webcam après recadrage et transformation perspective.

La transformation perspective joue un rôle crucial en rectifiant les distorsions dues à l'orientation de l'écran par rapport au visage de l'utilisateur. Cela garantit que les points projetés correspondent toujours avec précision aux positions souhaitées sur l'écran, quelle que soit l'orientation de celui-ci par rapport au visage.

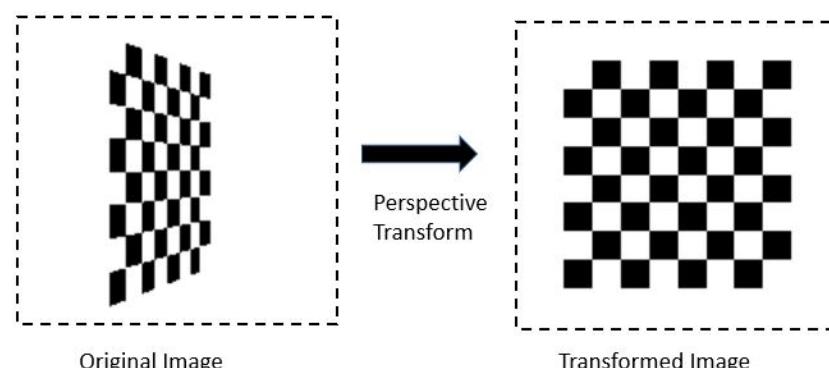


FIGURE 17 – Principe de transformation perspective.

Cette technique implique l'application d'une transformation géométrique à l'image capturée du visage de l'utilisateur. Cette transformation ajuste les perspectives de l'image, permettant ainsi de corriger les déformations qui peuvent survenir lorsque l'écran est vu sous différents angles ou orientations par rapport au visage de l'utilisateur, comme indiqué dans la figure suivante. En rendant une partie de l'image indépendante de l'orientation, la transformation perspective assure la précision du suivi des mouvements oculaires.

En intégrant la transformation perspective dans le système de suivi des mouvements oculaires, on garantit que les coordonnées des points projetés sont toujours correctement ajustées en fonction des mouvements du visage de l'utilisateur et de l'orientation de l'écran. Cela assure une interaction fluide et précise entre l'utilisateur et l'interface graphique, indépendamment des variations dans l'orientation du visage par rapport à la caméra.

4.3.6 Conclusion

Dans cette approche, nous commençons par la détection de la pupille ainsi que de quelques points d'intérêt du visage, puis nous effectuons un calibrage et projetons sur l'écran en considérant un visage fixe. Une fois que la projection est réalisée sur un visage fixe, nous passons à l'adaptation du calibrage aux mouvements du visage en combinant le recadrage et la transformation perspective. Cette méthode donne des résultats significatifs, surtout avec les petits mouvements du visage, même brusques, mais elle reste sensible aux mouvements verticaux du visage.

4.4 Approche de suivi de regard en espace tridimensionnel

4.4.1 Introduction

Dans cette deuxième approche, nous avons accordé une attention particulière au mouvement du visage, en mettant en œuvre des techniques pour détecter et suivre les changements dans la direction du regard. Pour cela, nous avons élaboré un système de calcul de score permettant de déterminer avec précision où l'utilisateur focalise son regard. Cette démarche nous a permis d'ajuster la projection du regard en fonction de ce score, assurant ainsi une localisation plus exacte des points d'intérêt visuels.

4.4.2 Pipeline

Dans cette approche, nous avons concentré nos efforts sur cinq parties essentielles pour parvenir à un suivi précis du regard. Tout d'abord, la détection des points clés du visage a été mise en avant, assurant ainsi une identification précise des zones d'intérêt. Ensuite, nous avons élaboré un processus de calcul du score de regard, permettant une évaluation minutieuse de la direction et de l'intensité du regard de l'utilisateur. Parallèlement, une attention particulière a été portée au calcul de la matrice de rotation. En utilisant cette matrice, nous avons procédé à la projection des axes du regard dans un espace tridimensionnel. Enfin, nous avons réalisé un ajustement précis de cette projection en fonction du score de regard obtenu, assurant ainsi une localisation précise des points d'intérêt visuels.



FIGURE 18 – Illustration des étapes clés pour la deuxième approche.

4.4.3 Détection des Points importants du visage

Dans notre projet de suivi du regard, nous avons mis en place un processus en plusieurs étapes pour détecter et suivre les mouvements oculaires avec précision. En utilisant la bibliothèque MediaPipe, nous avons extrait les landmarks du visage, en accordant une attention particulière aux repères du nez et des coins de la bouche pour une localisation précise. Ensuite, nous avons effectué un changement de repère pour repositionner les coins des yeux comme centres de référence, facilitant ainsi le calcul des mouvements du regard dans un espace tridimensionnel. En travaillant avec les points 3D, nous projetons les mouvements du regard dans cet espace pour une analyse plus approfondie et une meilleure compréhension des interactions visuelles.

4.4.4 Calcul du score de regard

Dans la deuxième partie de notre pipeline, nous avons implémenté des calculs sophistiqués pour évaluer le score de regard, une étape cruciale pour déterminer la direction précise du regard de l'utilisateur. En utilisant les coordonnées des landmarks du visage détectés par la bibliothèque MediaPipe, nous avons calculé les déplacements horizontaux et verticaux des pupilles pour chaque œil. À l'aide de ces informations, nous avons mis en place des algorithmes pour déterminer les scores de regard, en tenant compte des variations entre les positions actuelles et précédentes des pupilles. Ces calculs sont réalisés séparément pour les yeux gauche et droit, et chaque score est normalisé pour garantir une évaluation précise. Nous avons également intégré des mécanismes de seuillage pour filtrer les mouvements abérants et ainsi améliorer la stabilité de notre estimation du regard. Grâce à cette approche rigoureuse, notre système est capable de fournir une estimation dynamique et précise de la direction du regard de l'utilisateur.

```

# Calculate left x gaze score
if (face_2d[243,0] - face_2d[130,0]) != 0:
    lx_score = (face_2d[468,0] - face_2d[130,0]) / (face_2d[243,0] - face_2d[130,0])
    if abs(lx_score - last_lx) < threshold:
        lx_score = (lx_score + last_lx) / 2
    last_lx = lx_score

# Calculate left y gaze score
if (face_2d[23,1] - face_2d[27,1]) != 0:
    ly_score = (face_2d[468,1] - face_2d[27,1]) / (face_2d[23,1] - face_2d[27,1])
    if abs(ly_score - last_ly) < threshold:
        ly_score = (ly_score + last_ly) / 2
    last_ly = ly_score

# Calculate right x gaze score
if (face_2d[359,0] - face_2d[463,0]) != 0:
    rx_score = (face_2d[473,0] - face_2d[463,0]) / (face_2d[359,0] - face_2d[463,0])
    if abs(rx_score - last_rx) < threshold:
        rx_score = (rx_score + last_rx) / 2
    last_rx = rx_score

# Calculate right y gaze score
if (face_2d[253,1] - face_2d[257,1]) != 0:
    ry_score = (face_2d[473,1] - face_2d[257,1]) / (face_2d[253,1] - face_2d[257,1])
    if abs(ry_score - last_ry) < threshold:
        ry_score = (ry_score + last_ry) / 2
    last_ry = ry_score

```

FIGURE 19 – Calcul du score de la direction du regard.

- `if (face_2d[243,0] - face_2d[130,0]) != 0:` Cette ligne vérifie si la différence des coordonnées x entre deux points de repère faciaux (243 et 130) n'est pas égale à zéro. Cela est nécessaire pour éviter une division par zéro dans le calcul suivant.
- `lx_score = (face_2d[468,0] - face_2d[130,0]) / (face_2d[243,0] - face_2d[130,0])`
Cette ligne calcule le score de direction du regard sur l'axe x pour l'œil gauche. Il est obtenu en prenant la différence des coordonnées x entre deux points de repère faciaux spécifiques (468 et 130), puis en la divisant par la différence des coordonnées x entre deux autres points de repère faciaux (243 et 130).
- `if abs(lx_score - last_lx) < threshold:` Cette ligne vérifie si la différence entre le score de direction du regard actuel (`lx_score`) et le dernier score enregistré (`last_lx`) pour l'axe x est inférieure à un seuil spécifié (`threshold`). La présence de cette condition permet de filtrer les variations brusques ou les valeurs aberrantes dans les données de suivi du regard.
- `lx_score = (lx_score + last_lx) / 2` : Si la condition précédente est satisfaite, cette ligne moyenne le score de direction du regard actuel avec le dernier score enregistré pour lisser les fluctuations. Cela permet d'obtenir une estimation plus stable de la direction du regard.

→ Pour fournir une indication visuelle de la direction du regard, une modification est apportée au score calculé pour chaque axe (x et y). Après avoir calculé le score de direction du regard, une constante de 0.5 est soustraite de chaque score. Ce faisant, la plage des valeurs des scores est décalée de sorte que -0.5 représente la direction extrême gauche, 0 représente le centre et 0.5 représente la direction extrême droite (ou vers le haut pour l'axe y et vers le bas pour l'axe y). Ce décalage fournit une indication claire de la direction vers laquelle le regard est dirigé.

4.4.5 Calcul de la matrice de rotation de la caméra

Etude théorique

La calibration de la caméra est un processus crucial dans le domaine de la vision par ordinateur et de la vision informatique. Elle vise à estimer les paramètres intrinsèques et extrinsèques de la caméra, ce qui est essentiel pour obtenir des mesures précises à partir d'images capturées.

Paramètres intrinsèques :

Les paramètres intrinsèques de la caméra comprennent la distance focale, le centre optique et la distorsion radiale.

- **Distance focale (f)** : C'est la distance entre le centre optique de la lentille et le point focal, où les rayons lumineux se croisent et forment une image nette sur le plan image.
- **Centre optique (ou point principal)** : Il s'agit du point à partir duquel les rayons lumineux sont projetés sur le plan image sans déviation. Il est généralement situé au centre de l'image, à l'intersection de l'axe optique et du plan image.
- **Distorsion radiale** : C'est une aberration optique courante dans les lentilles de caméra, où les rayons lumineux ne se projettent pas uniformément sur le capteur d'image, provoquant une déformation des objets, en particulier vers les bords de l'image.

Matrice de la caméra

La matrice de la caméra, également appelée matrice intrinsèque, est une matrice qui représente les paramètres intrinsèques de la caméra. Elle est généralement représentée comme suit :

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Où f_x et f_y sont les distances focales horizontale et verticale, c_x et c_y sont les coordonnées du centre optique (ou principal point), et s est le facteur de distorsion radiale, qui est généralement 0 pour des caméras bien calibrées.

La calibration de la caméra est généralement réalisée en utilisant des techniques telles que la calibration à partir de motifs planaires ou des images de calibrage spécifiques. Une fois ces paramètres déterminés, ils peuvent être utilisés pour corriger la distorsion et effectuer des calculs de géométrie 3D à partir d'images 2D.

Etude Pratique

```
# Solve PnP
_, l_rvec, l_tvec = cv2.solvePnP(leye_3d, face_2d_head, cam_matrix, dist_coeffs, flags=cv2.SOLVEPNP_ITERATIVE)
_, r_rvec, r_tvec = cv2.solvePnP(reye_3d, face_2d_head, cam_matrix, dist_coeffs, flags=cv2.SOLVEPNP_ITERATIVE)

# Get rotational matrix from rotational vector
l_rmat, _ = cv2.Rodrigues(l_rvec)
r_rmat, _ = cv2.Rodrigues(r_rvec)
```

FIGURE 20 – Calcul des vecteurs de translation et de rotation.

1. `_, l_rvec, l_tvec = cv2.solvePnP(leye_3d, face_2d_head, cam_matrix, dist_coeffs, flags=cv2.SOLVEPNP_ITERATIVE)` : Cette ligne résout le problème de la perspective-n-point (PnP) pour estimer la pose de l'œil dans l'espace 3D. Cette fonction retourne un vecteur de rotation (`l_rvec`) et un vecteur de translation (`l_tvec`) qui décrivent la pose de l'œil gauche dans l'espace 3D par rapport à la caméra.
2. `l_rmat, _ = cv2.Rodrigues(l_rvec)` : Cette ligne convertit le vecteur de rotation de l'œil gauche (`l_rvec`) en une matrice de rotation 3x3 à l'aide de la fonction `cv2.Rodrigues()`. Cela est nécessaire car `solvePnP` renvoie le vecteur de rotation plutôt que la matrice de rotation directement. La matrice de rotation (`l_rmat`) est ensuite utilisée pour représenter la rotation de l'œil dans l'espace 3D.

4.4.6 Projection des axes du regard

Pour projeter les axes représentant le regard dans un espace tridimensionnel, la fonction `cv2.projectPoints()` est utilisée comme suit :

```
l_axis, _ = cv2.projectPoints(axis, l_rvec, l_tvec, cam_matrix, dist_coeffs)
```

Cette ligne projette les axes représentant le regard gauche (`l_axis`) dans l'espace 3D en utilisant les paramètres suivants :

- `axis` : Les points décrivant les axes dans l'espace 3D. Ces points sont généralement définis pour représenter les axes X, Y et Z dans un espace tridimensionnel.
- `l_rvec` : Le vecteur de rotation représentant la rotation de l'œil dans l'espace 3D.
- `l_tvec` : Le vecteur de translation représentant la translation de l'œil dans l'espace 3D.
- `cam_matrix` : La matrice qui contient les paramètres intrinsèques de la caméra.
- `dist_coeffs` : Les coefficients de distorsion utilisés pour corriger les distorsions de l'image.

Cette fonction renvoie les projections des axes du regard gauche (`l_axis`) dans le repère 2D de l'image.



FIGURE 21 – Résultat de l'estimation du regard dans l'espace 3D.

On observe à partir de l'image les marqueurs 3D des deux yeux, lesquels indiquent la direction précise du regard. Ces marqueurs peuvent être utilisés pour déterminer où l'utilisateur focalise son attention dans l'espace tridimensionnel.

4.4.7 Ajustement des axes en fonction du score corrigé

Dans cette étape, les vecteurs de rotation de chaque œil sont ajustés en fonction des scores corrigés du regard. Les scores corrigés, obtenus à partir des coordonnées normalisées du regard, sont utilisés pour ajuster les vecteurs de rotation et ainsi affiner la précision de la détection du regard.

Pour l'œil gauche, le vecteur de rotation `l_rvec` est converti en un tableau numpy `l_gaze_rvec`, puis les composantes du vecteur sont modifiées en fonction du score corrigé du regard. La composante correspondant à l'axe Z est ajustée en fonction de l'écart entre le score et la valeur médiane, multiplié par un facteur de correction. De même, la composante correspondant à l'axe X est ajustée en fonction de l'écart entre le score et la valeur médiane multiplié par un autre facteur de correction. Ce processus permet d'adapter le vecteur de rotation en fonction de la direction et de l'intensité du regard.

Le même processus est appliqué à l'œil droit, où le vecteur de rotation `r_rvec` est converti en `r_gaze_rvec` et ajusté en fonction du score corrigé du regard droit.

En ajustant les vecteurs de rotation en fonction des scores corrigés du regard, cette étape contribue à améliorer la précision et la fidélité de la représentation du regard dans l'espace tridimensionnel, ce qui est essentiel pour les applications de suivi du regard en temps réel.

4.5 Conclusion

Dans cette section, nous avons exploré le processus de projection et de représentation du regard dans un espace tridimensionnel. En utilisant des techniques telles que la résolution du problème de **la perspective-n-point (PnP)** et la manipulation des vecteurs de rotation, nous avons pu estimer la pose des yeux dans l'espace 3D par rapport à la caméra. En projetant des axes représentant le regard sur l'image et en ajustant ces axes en fonction des scores corrigés du regard, nous avons amélioré la précision et la fidélité de la détection du regard.

Cette approche permet d'obtenir une représentation plus précise du regard dans un environnement tridimensionnel, ce qui est essentiel pour les applications de suivi du regard en temps réel .

Cependant, malgré les progrès réalisés, il reste des défis à relever, notamment la précision de la détection dans des conditions de faible luminosité ou avec des obstacles visuels. En poursuivant la recherche dans ce domaine et en développant des méthodes innovantes, nous pouvons continuer à améliorer les performances des systèmes de suivi du regard et à ouvrir de nouvelles possibilités pour leur utilisation dans divers domaines d'application.

4.6 Approche de suivi du regard en utilisant WebGazer

Pour réaliser le suivi du regard, nous avons utilisé WebGazer, une bibliothèque JavaScript open-source conçue pour suivre les mouvements oculaires à l'aide d'une webcam. En utilisant cette bibliothèque, nous avons développé une interface simple en HTML, CSS et JavaScript qui utilise l'API de WebGazer pour capturer les données de suivi du regard et les afficher de manière conviviale pour l'utilisateur.

La figure ci-dessous illustre les résultats obtenus grâce à cet outils. En intégrant notre interface personnalisée avec WebGazer, nous avons pu observer les mouvements oculaires des utilisateurs et estimer les points de regard sur l'écran (point en rouge dans la figure 23).

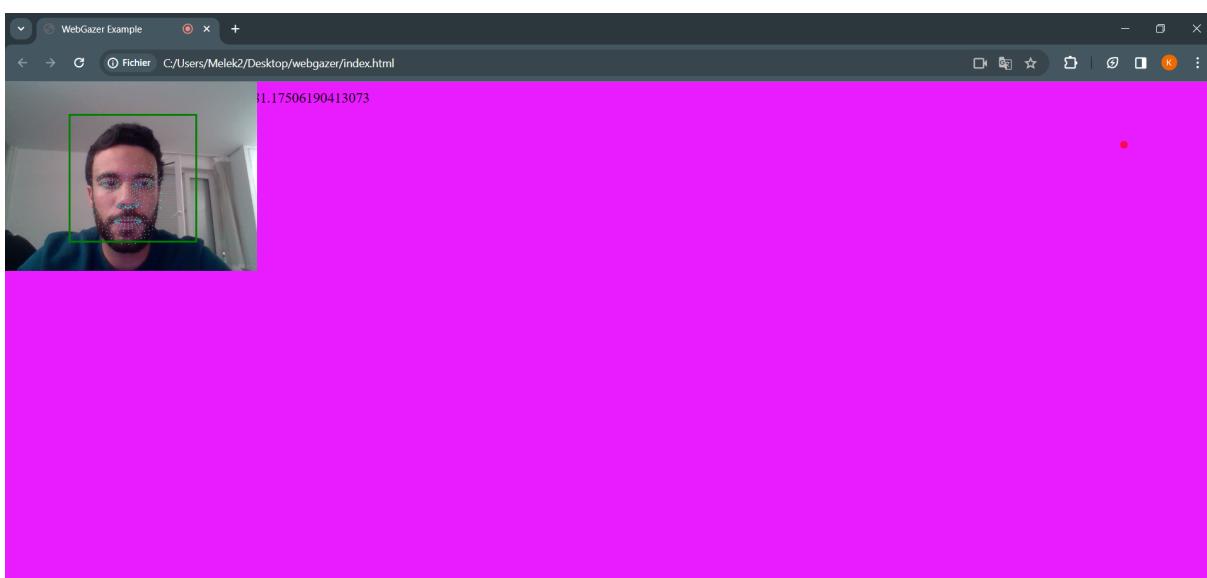


FIGURE 22 – Capture d'écran de notre interface utilisant WebGazer.

WebGazer offre une solution accessible et simple à mettre en œuvre pour le suivi du regard, en exploitant les capacités de traitement d'image des navigateurs web modernes.

L'utilisation de WebGazer dans notre étude a permis une estimation du point de regard, même pour les visages en mouvement. Cette bibliothèque utilise des techniques avancées de suivi oculaire, combinant la détection de la pupille avec des algorithmes de calibration et d'adaptation au mouvement du visage.

Étant une bibliothèque prête à l'emploi, WebGazer offre des fonctionnalités de base pour le suivi du regard sans offrir beaucoup de possibilités de personnalisation avancée. Cette limitation peut rendre WebGazer inadéquat pour le suivi du regard des bébés dans notre cas d'utilisation spécifique.

4.7 Conclusion

Dans cette section, nous avons présenté le fonctionnement de notre projet à travers des captures d'écran, illustrant les différentes étapes, de la détection des yeux à la projection des points. Cet algorithme assure un suivi précis du regard et une interaction fluide avec l'interface utilisateur. Nous avons détaillé une première approche pour le suivi du regard chez les nourrissons, ainsi qu'une autre méthode en espace tridimensionnel. Enfin, nous avons exploré une alternative utilisant WebGazer. En résumé, notre rapport met en avant notre expertise dans le suivi du regard, en proposant des solutions adaptées aux différents besoins des utilisateurs.

5 Conclusion et perspectives

Notre objectif principal dans ce projet était de concevoir, développer et mener des recherches sur le développement d'un algorithme de suivi du regard de nourrissons pour des évaluations de développement cognitif, capable de fonctionner efficacement avec des images de webcams standard et de faciliter l'utilisation de cette plateforme. Dans ce rapport, nous avons présenté en détail notre démarche, depuis la définition des objectifs jusqu'à la phase de réalisation. Nous avons commencé par exposer la problématique à résoudre, les objectifs que nous nous sommes fixés, ainsi que le contexte de notre travail. Ensuite, nous avons réalisé une analyse approfondie des solutions existantes, tout en présentant notre propre solution qui répond aux besoins de notre projet. Le développement de cette plateforme nous a permis de mettre en pratique nos compétences en développement et en gestion de projet.

En ce qui concerne les perspectives d'amélioration et de développement futur pour notre algorithme, voici quelques axes à considérer :

validation et études sur le terrain : Pour valider l'efficacité et l'exactitude de notre algorithme dans un contexte réel, il serait essentiel de mener des études et des tests sur le terrain. Ces études nous permettraient de recueillir des données empiriques sur la performance de notre solution dans des situations d'utilisation réelles, tout en identifiant les éventuels points faibles à améliorer.

Augmenter la précision de l'algorithme : Bien que notre algorithme soit déjà capable de fournir des résultats précis, il existe toujours des possibilités d'amélioration de sa précision. Nous pourrions envisager d'explorer des méthodes avancées de traitement d'image et d'apprentissage automatique pour affiner la détection des landmarks et améliorer la précision du suivi du regard.

Intégrer des fonctionnalités d'analyse de données avancées : En plus du suivi du regard, nous pourrions envisager d'intégrer des fonctionnalités avancées d'analyse de données pour extraire des informations supplémentaires à partir des données recueillies. Cela pourrait inclure des outils pour analyser les modèles de mouvement oculaire, détecter les anomalies et identifier les tendances de développement cognitif chez les nourrissons.

Ces perspectives d'amélioration contribueront à renforcer l'efficacité et la pertinence de notre algorithme de suivi du regard de nourrissons dans le domaine de l'évaluation du développement cognitif.

Références

- [1] [https://babylablyon.fr/.](https://babylablyon.fr/)
- [2] [https://www.tobii.com/.](https://www.tobii.com/)
- [3] [https://theeyetribe.com/theeyetribe.com/about/index.html.](https://theeyetribe.com/theeyetribe.com/about/index.html)
- [4] [https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08.](https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08)
- [5] [https://developers.google.com/mediapipe.](https://developers.google.com/mediapipe)
- [6] [https://github.com/TadasBaltrusaitis/OpenFace/blob/master/README.md.](https://github.com/TadasBaltrusaitis/OpenFace/blob/master/README.md)
- [7] [https://opencv.org/.](https://opencv.org/)
- [8] Yotam Erel, Kat Adams Shannon, Junyi Chu, Kimberly Megan Scott, Melissa Kline Struhl, Peng Cao, Xincheng Tan, Peter K Hart, Gal Raz, Sabrina Piccolo, Catherine Mei, Christine Potter, Sagi Jaffe-Dax, Casey Lew-Williams, Joshua Tenenbaum, Katherine Fairchild, Amit Bermano, and Shari Liu. iCatcher+ : Robust and automated annotation of infant's and young children's gaze direction from videos collected in laboratory, field, and online studies. preprint, PsyArXiv, May 2022.
- [9] Chetana Krishnan, Vijay Jeyakumar, and Alex Noel Joseph Raj. Real-time eye tracking using heat maps. *Malaysian Journal of Computer Science*, 35(4) :339–358, October 2022.
- [10] Chang Huan Lo, Nivedita Mani, Natalia Kartushina, Julien Mayor, and Jonas Hermes. e-Babylab : An Open-source Browser-based Tool for Unmoderated Online Developmental Studies. preprint, PsyArXiv, February 2021.