

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

www.polytech.univ-tours.fr

**Projet Recherche & Développement
2015-2016**

Application d'aide à l'interaction homme/machine pour les personnes handicapées

Tuteurs académiques

Mohamed SLIMANE

Donatello CONTE

Étudiants

Florian TISSIER (DI5)

26 septembre 2016

Liste des intervenants

Nom	Mail	Qualité
Florian TISSIER	florian.tissier@etu.univ-tours.fr	Étudiant DI5
Mohamed SLIMANE	mohamed.slimane@univ-tours.fr	Tuteur académique, Département infomatique
Donatello CONTE	donetello.conte@univ-tours.fr	Tuteur académique, Département infomatique

Avertissement

Ce document a été rédigé par Florian Tissier susnommé l'auteur.

L'école polytechnique de l'université François Rabelais de Tours est représentée par Mohamed Slimane et Donatello Conte susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document :

Florian Tissier, *Application d'aide à l'interaction homme/machine pour les personnes handicapées*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2015-2016.

```
@mastersthesis{
  author={Tissier, Florian},
  title={Application d'aide à l'interaction homme/machine pour les personnes handicapées},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2015-2016}
}
```

Table des matières

Introduction	1
I Recherche	3
1 Cahier des charges du projet	4
1 MOA.....	4
2 MOE.....	4
3 Contexte et présentation	4
4 Problématique et objectif	4
5 Périmètre.....	5
6 Description fonctionnelle.....	5
6.1 Repérer le visage.....	5
6.2 Reconnaître l'expression.....	5
7 Budget.....	6
8 Délai.....	6
9 Livrable.....	6
2 Émotions universelles et mouvements du visage	7
1 Les 7 émotions universelles.....	7
2 Le FACS.....	8
3 La norme MPEG-4.....	8
3 État de l'art	11
1 3D.....	11
1.1 Techniques d'acquisition	11
1.1.1 Reconstruction à partir d'une image.....	11
1.1.2 Lumière structurée	12
1.1.3 Stéréo photométrie	12

1.1.4	Stéréo multi-vue	13
1.2	Dispositifs d'acquisition d'images.....	13
1.2.1	Kinect	13
1.2.2	Minolta Vivid 910.....	13
1.3	Base de données de visages 3D.....	14
1.3.1	BU-3DFE.....	15
1.3.2	BU-4DFE.....	15
1.3.3	Bosphorus 3D Face Database	15
1.3.4	Comparatif des différentes bases de données 3D	16
2	2D.....	16
2.1	Dispositifs d'acquisition d'images 2D	16
2.2	Bases de données de visages 2D	17
2.2.1	Cohn-Kanade (CK) [10].....	17
2.2.2	CK+ [16].....	18
2.2.3	Man-Machine Interaction (MMI) [27][20].....	18
2.2.4	MHI Mimicry [3] [13]	19
2.2.5	HCI Tagging [24] [12]	19
2.2.6	Comparatif des bases de données 2D.....	20
4	Architecture d'un système de reconnaissance d'émotions	23
1	Détection du visage	23
1.1	Absolu	23
1.2	Différentiel.....	24
1.3	Comparatif	24
2	Extraction des <i>features</i>	24
2.1	Filtres de Gabor	25
2.2	Composantes pseudo-Haar.....	26
2.3	Motifs binaires locaux.....	30
3	Classification	30
3.1	Classificateurs binaires.....	31
3.2	Classificateurs multi-classes.....	31
5	Nouvelle approche	34
1	Contexte de simulation	34
2	Modèle réalisé.....	35
3	Fonctionnement.....	37
4	Comparaison.....	37
5	Programme Matlab.....	38
6	Modification à apporter	38
6	Spécifications de l'application	40
1	Utilisateurs	40
2	Fonctions.....	40

7	Planning et outils utilisés	41
1	Planning du projet.....	41
2	Outils	42
2.1	Versioning	42
2.2	Gestion de projet.....	42
II	Développement	43
8	Identifications des modifications	44
1	Changement de la base d'apprentissage	44
2	Changement de la représentation des données	44
3	Utilisation de l'algorithme F-GPLVM	45
4	Représentation des données sous formes de trajectoires.....	45
9	Rédaction de l'article scientifique pour la conférence "Handicap 2016" de l'IFRATH	46
10	Réalisations des modifications	47
1	Création de la nouvelle base d'apprentissage	47
2	Représenter les données via la position des <i>features</i>	47
3	Utilisation de l'algorithme F-GPLVM (Faster GPLVM) pour créer l'espace latent	49
4	Représenter et comparer des trajectoires dans l'espace latent	50
11	Tests et résultats	56
12	Notice d'utilisation de mon application	58
1	Extraction des coordonnées des <i>features</i> des visages	58
2	Génération des données de construction l'espace latent	58
3	Affichage de l'espace latent ainsi que des données de test.....	59
13	Gestion de projet et planning final	60
1	Méthodologie	60
2	Gestion de version	60
3	Planning final	60
	Conclusion	62
	Annexes	64
A	Article Handicap 2016	65

Table des figures

2 Émotions universelles et mouvements du visage

1	Exemples d'AUs	8
2	Quelques Facial Features Points utilisés par la norme MPEG-4	9
3	Facial Animation Parameter Units utilisés par la norme MPEG-4.....	10

3 État de l'art

1	Reconstruction d'un visage 2D (1) en 3D (2) grâce à la méthode 3DMM.....	12
2	Exemple de lumière structurée	12
3	Exemple de reconstruction d'un objet grâce à la stéréo photométrie	13
4	Version 2 de la Kinect de Microsoft	14
5	Minolta Vivid 910.....	14
6	Comparatif de la qualité entre Kinect et Minolta.....	14
7	Exemple de données contenues dans BU-3DFE	15
8	Exemple de données contenues dans la base de données Bosphorus	16
9	Exemple de caméra PTZ.....	17
10	Exemple de données de CK (de haut en bas et de gauche à droite : neutre, surprise, joie, colère, dégoût)	18
11	Exemple de données de MMI	19
12	Exemple de données de MHI Mimicry.....	20
13	Même image avec 2 tags différents : le premier erroné (Kiss), le deuxième correct (Handshake)	21
14	Données contenues dans HCI Tagging.....	21

4 Architecture d'un système de reconnaissance d'émotions

1	Image utilisée pour vérifier l'effet des filtres de Gabor.....	26
2	Effet du changement du paramètre λ	27
3	Effet du changement du paramètre θ	27
4	Effet du changement du paramètre Ψ	28

5	Effet du changement du paramètre γ	28
6	Effet du changement du paramètre n	29
7	Effet du changement du paramètre b associé à σ	29
8	Composantes pseudo-Haar utilisé par Viola et Jones.....	29
9	Extension de composantes pseudo-Haar défini par Lienhart et Maydt.....	30
10	Fonctionnement des motifs binaires locaux	31
11	Exemple de projection dans un espace de plus grande dimension réalisé par SVM.....	32
12	Fonctionnement du kPPV avec $k=3$, 3 classes et 4 échantillons dont un indécis.....	33
 5 Nouvelle approche		
1	Situation de simulation pour la détection d'émotions	35
2	Fonctionnement du modèle probabiliste	37
3	Espace latent représenté grâce au programme Matlab.....	39
4	Exemples d'images retournées lorsque l'on se déplace sur l'espace latent	39
 10 Réalisations des modifications		
1	Images de notre base d'apprentissage pour l'émotion de la joie	53
2	66 features du visage trouvé grâce au programme de A .Asthana	54
3	Structure utilisée pour stocker les données	54
4	Espace latent permettant de représenter les émotions.....	55
5	Test : représentation de l'émotion de la joie utilisée lors de la construction de l'espace latent	55
 11 Tests et résultats		
1	Temps total passé dans chaque fonction lors de l'affichage de l'espace latent avec des données de test	57

Liste des tableaux

2 Émotions universelles et mouvements du visage	
1 Exemple de Facial Action Parameters	9
3 État de l'art	
1 Comparaison de bases de données 3D	16
2 Comparaison de bases de données 2D (P/S : Posée/Spontanée)	22
4 Architecture d'un système de reconnaissance d'émotions	
1 Avantages/inconvénients des 2 types de détecteurs.....	24
7 Planning et outils utilisés	
1 Planning du projet.....	41
10 Réalisations des modifications	
1 Données utilisées dans notre base d'apprentissage	47
11 Tests et résultats	
1 Données utilisées dans ma base de test	56
13 Gestion de projet et planning final	
1 Planning du projet.....	61



Introduction

L'interaction entre les hommes et les machines a toujours été un enjeu de taille. Arriver à faire communiquer un ordinateur avec un être humain est un défi de tous les jours et est de plus en plus présent dans notre quotidien. Nous pouvons par exemple citer *Siri* d'Apple qui permet de communiquer avec son smartphone simplement en parlant.

C'est dans cette optique de facilitation du quotidien grâce à l'interaction avec une machine que ce projet prend place.

Le Projet de Recherche et Développement (anciennement Projet de Fin d'Études) est un projet se déroulant durant toute la 5ème année de master ingénieur au sein de l'école Polytech Tours. Ce rapport va présenter les travaux que j'ai effectué durant toute la durée de ce projet.

Ce rapport va donc se diviser en deux grandes parties : tout d'abord la partie Recherche qui va contenir le cahier des charges du projet, l'état de l'art en matière de reconnaissance faciale d'émotions ainsi que toutes les bases théoriques dont j'aurais besoin par la suite.

La deuxième partie est la partie Développement qui va se concentrer sur les différentes étapes du développement de cette application d'aide aux personnes handicapées.

Dans la partie Recherche de ce rapport, après avoir défini le cahier des charges, je vais vous présenter les sept émotions universelles ainsi que deux normes majeures permettant de définir les mouvements du visage qui composent une expression.

Je vous présenterai ensuite un état de l'art en matière de reconnaissance faciale d'expression, notamment les différentes techniques permettant de capturer un visage et de reconnaître une émotion, tout d'abord en 3D puis en 2D, ainsi que les avantages et inconvénients de chacune de ces techniques. Dans chacune de ces étapes, je présenterai l'état de l'art actuelle en terme de matériel, de méthodes et également de base de données disponibles.

Je continuerai ensuite en présentant les différentes étapes nécessaires à la construction d'un système de reconnaissance faciale d'émotions performant et efficace.

Je définirai ensuite les spécifications de l'application ainsi que les choix qui ont permis cette définition. Je conclurai par un planning prévisionnel ainsi que par les méthodologies et les outils de suivi utilisés durant ce projet.

Dans la partie Développement, je présenterai, dans un premier chapitre, les modifications qui ont été identifiées (notamment grâce à un rendez vous avec Giuseppe Boccignone) et qui vont nous permettre de modifier le programme de Vitale et al. pour l'adapter à notre problématique.

J'expliquerai ensuite dans un second chapitre comment j'ai réalisé ces modifications.

Durant la phase d'identification des modifications, j'ai également dû rédiger un article scientifique, à propos de mes travaux réalisés durant ce projet, qui sera présenté, s'il est accepté, à la conférence Handicap 2016 organisé par l'IFRATH à Paris du 8 au 10 Juin 2016.

Pour ce projet de recherche et développement, j'ai été encadré par Donatello CONTE et Mohamed SLIMANE.

Première partie

Recherche

1

Cahier des charges du projet

1 MOA

- Donatello CONTE : Maître de conférence et enseignant chercheur en informatique à l'école Polytech Tours
- Mohamed SLIMANE : Professeur des Universités et enseignant chercheur en informatique à l'école Polytech Tours

2 MOE

Florian TISSIER : élève en dernière année de master ingénieur en informatique à Polytech Tours

3 Contexte et présentation

Ce projet de recherche et développement prend place dans le cursus de dernière année de master ingénieur dispensé à l'école Polytech Tours.

Monsieur Slimane étant membre d'une association s'occupant de personnes handicapées, il souhaitait pouvoir aider et faciliter la vie de ces derniers via un projet réalisé au sein de l'école.

Le but de ce projet de recherche et développement est de construire un système permettant, à partir d'un flux vidéo acquis grâce à une caméra, de détecter l'expression actuelle du visage d'une personne handicapée dans le but de réaliser certaines actions pouvant améliorer son bien être.

4 Problématique et objectif

Comment faciliter la vie quotidienne des personnes handicapées grâce à leurs émotions ?

L'objectif est de réaliser une application qui pourra détecter en temps réel les émotions d'une personne handicapée se trouvant dans une pièce à l'aide d'une caméra fixée à un mur de cette même pièce. La caméra devra être capable de bouger et de zoomer pour suivre le visage de la personne.

Une fois l'émotion détectée, des actions spécifiques devront être réalisées (ex : changement de la couleur de la lumière, lecture de musique douce...).

5 Périmètre

Ce projet se concentre principalement sur les personnes handicapées mais l'application qui résultera de ce projet pourra également être utilisée pour des personnes non handicapées. L'application devra être fonctionnel dans n'importe quelle pièce d'une maison ou d'une structure spécialisée dans l'accueil de personnes handicapées.

6 Description fonctionnelle

Le projet se découpe en 2 fonctions principales :

- Repérer le visage
- Reconnaître l'expression

Chacune de ces fonctions se décomposent en plusieurs sous-fonctions.

6.1 Repérer le visage

Cette première fonction principale va permettre de trouver le visage d'une personne, de le suivre et de zoomer dessus.

Les sous-fonctions suivantes seront donc nécessaires :

- Un algorithme de détection de visage dans une image
- Un algorithme de suivi de visage
- Un algorithme de zoom

Vous trouverez ci-après un descriptif plus précis de chacune de ces sous-fonctions.

Fonction F1 : Repérer le visage/Détection du visage	
Objectif	Détecter un visage dans un environnement quelconque.
Description	En analysant les frames d'un flux vidéo, cet algorithme nous renverra la position d'un cadre entourant le visage trouvé.
Contraintes	Cet algorithme doit être rapide.
Niveau de priorité	Haute
Fonction F2 : Repérer le visage/Suivi du visage	
Objectif	Suivre le visage entre plusieurs frames d'un flux vidéo dans le but de ne pas le perdre.
Description	En comparant 2 frames consécutives d'un flux vidéo, l'algorithme devra nous dire le déplacement du visage pour pouvoir le suivre avec la caméra. Il devra également être capable de faire le suivi si jamais le visage se retrouve occulté pendant quelques secondes.
Contraintes	Éviter le plus possible l'accumulation d'erreur de précision pouvant amener à la perte du visage.
Niveau de priorité	Haute
Fonction F3 : Repérer le visage/Zoom	
Objectif	Zoomer sur une zone de l'image.
Description	L'algorithme devra pouvoir zoomer sur le cadre contenant le visage renvoyé par la sous-fonction F1.
Contraintes	Disposer d'une caméra ayant la possibilité de zoomer.
Niveau de priorité	Moyenne

6.2 Reconnaître l'expression

Cette deuxième fonction principale va permettre d'identifier l'expression faciale de la personne.

Les sous-fonctions suivantes seront donc nécessaires :

- Un algorithme d'apprentissage
- Un algorithme d'extraction des points clés (*features*) du visage
- Un algorithme de classification

Vous trouverez ci-après un descriptif plus précis de chacune de ces sous-fonctions.

Fonction F4 : Reconnaître l'expression/Apprentissage	
Objectif	Apprendre au système à classifier les expressions en fonction d'une base d'apprentissage.
Description	Pour chaque élément dans la base d'apprentissage, une émotion lui sera associé. Cela va permettre au système d'apprendre à quelle expression du visage appartient une émotion.
Contraintes	La base d'apprentissage doit être assez fournie et pertinente pour permettre un apprentissage performant.
Niveau de priorité	Haute

Fonction F5 : Reconnaître l'expression/Extraction des <i>features</i> du visage.	
Objectif	Extraire les <i>features</i> du visage pour pouvoir ensuite réaliser une classification.
Description	Cet algorithme devra extraire les <i>features</i> du visage d'une personne se trouvant sur une frame d'un flux vidéo. Les <i>features</i> d'un visage sont par exemple le coin des yeux, la position de la pupille, les coins de la bouche, le nez, les joues... Les positions des <i>features</i> retournées permettront la classification.
Contraintes	
Niveau de priorité	Haute

Fonction F6 : Reconnaître l'expression/Classification	
Objectif	Classifier l'expression du visage et retourner l'émotion associée.
Description	L'algorithme devra classifier l'expression du visage de la personne récupérée depuis un flux vidéo, grâce aux positions des <i>features</i> , en fonction de l'apprentissage qui aura été effectué précédemment.
Contraintes	Cet algorithme doit être rapide et fiable (au moins 90% de reconnaissance).
Niveau de priorité	Haute

7 Budget

Ce projet ne dispose pas d'un budget précis.

Néanmoins en réalisant un état de l'art des matériels disponibles à notre projet, nous avons décidé de choisir une caméra ne dépassant pas les 500€.

8 Délai

La partie recherche devra être fini pour le 15 janvier 2016 et la partie développement (et donc le projet complet) devra être fini pour la fin du second semestre, c'est-à-dire fin mars.

9 Livrable

Le livrable de ce projet correspondra en une application Matlab qui réalisera la phase d'apprentissage et la classification des émotions.

2

Émotions universelles et mouvements du visage

Ce chapitre va me permettre d'introduire les notions nécessaires à la décomposition et donc à la reconnaissance d'une expression faciale et de son émotion associée.

Pour un simple sourire, nous utilisons une vingtaine de muscles (les muscles zygomatiques), il ne peut donc pas être décrit par un seul mouvement du visage mais plusieurs. C'est exactement la même chose pour une expression, on ne la reconnaît que grâce à l'ensemble des mouvements faciaux qui la composent. C'est sur ce principe qu'ont été créés les deux normes de description des mouvements du visage que je vais vous présenter : le FACS et MPEG-4.

L'utilisation de l'une ou l'autre de ces normes permet de définir entièrement le spectre des mouvements rentrant en jeu dans n'importe quelle émotion.

Mais tout d'abord, je vais vous introduire les sept émotions universelles qui seront utilisées tout au long de mon projet et donc de ce rapport.

1 Les 7 émotions universelles

A ce jour, il a été démontré qu'il existait 7 émotions qui partagent une expression universellement compréhensible.

On considère qu'une émotion possède une expression universelle si tout individu est capable d'exprimer cette émotion et est également capable de la reconnaître et de l'interpréter chez autrui.

Les sept émotions universelles sont donc les suivantes :

- la neutralité
- la joie
- la tristesse
- la colère
- la peur
- la surprise
- le dégoût

C'est Charles Darwin qui, en 1872 dans son livre [5], a introduit cette idée d'émotions universelles entre les hommes mais également entre différentes espèces. Il a observé que les hommes et les animaux partagent des émotions comprises par tous et qui sont nécessaires à leur survie.

Mais ce n'est qu'en 1971 que le psychologue Paul Ekman, après un voyage en Papouasie-Nouvelle-Guinée, a confirmé les théories de Darwin. Dans son article [7] écrit avec la participation de Wallace Friesen, il définit les 7 émotions universelles citées plus haut.

2 Le FACS

En 1978, Ekman et Friesen publie [6] et apporte une nouvelle pierre à l'édifice en définissant un système de codification manuelle des expressions du visage : le **Facial Action Coding System** (FACS). Ce système décompose tous les mouvements du visage en 46 **Action Units** (AU), chacune décrivant la contraction ou la décontraction d'un ou plusieurs muscles du visage. La Figure 1 représente certaines de ces AUs.

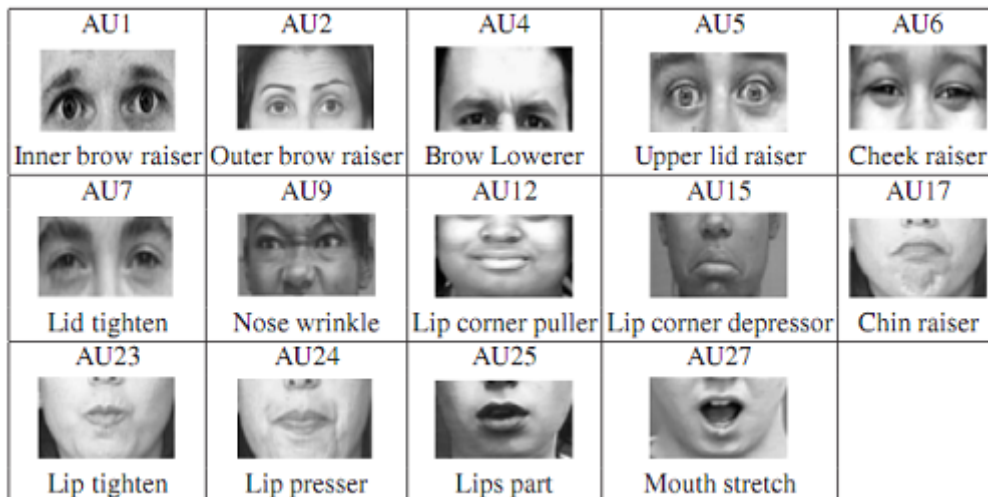


Figure 1 – Exemples d'AUs

La composition de plusieurs AU permet donc de décrire une expression et donc de reconnaître une émotion. Par exemple, un sourire et donc l'émotion de la joie est composé des AUs 6 (remontée des joues) et 12 (étirement du coin des lèvres).

La tristesse quand à elle va être composée des AUs 1, 4 et 15 et la colère des AUs 4, 5, 7 et 23.

N'importe quelle expression du visage peut donc être représentée par une combinaison d'AUs, ce qui fait de FACS le système le plus utilisé par les psychologues ainsi que par les personnes travaillant sur la reconnaissance faciale d'émotions.

Le système **FACS** possède également un degré d'intensité allant de A à E et permettant de spécifier l'intensité d'une AU :

- A : Très Faible
- B : Minime
- C : Moyen
- D : Sévère
- E : Maximum

La surprise peut donc être défini comme la combinaison des AUs 1, 2, 5B et 26.

Enfin des ajouts ont été apportés à cette norme. 13 nouveaux AUs ont été ajoutées pour décrire le mouvement de la tête et 7 autres pour le mouvement des yeux.

Nous arrivons donc à un total de 66 AUs, chacune possédant 5 intensités, permettant de décrire les mouvements faciaux.

Néanmoins, un autre système de codification fait concurrence au FACS et est également bien implanté dans le milieu de la reconnaissance d'émotions.

3 La norme MPEG-4

La norme MPEG-4, qui est une norme de codage vidéo, dispose de son propre système permettant de normaliser les mouvements du visage et de reconnaître des expressions.

Pour cela, ce système définit des points clés du visage (**Figure 2**) appelés **Facial Features Points** (FFP) auxquels seront appliqués des mesures pour créer des distances entre ces FFP (**Figure 3**) appelées **Facial Animation Parameter Units** (FAPU).

Ces FAPU vont servir à la description des mouvements musculaires appelés **Facial Action Parameters** (FAP, équivalent des AUs de la norme FACS). 68 FAPs sont recensés à ce jour, j'en ai regroupé quelques uns dans le tableau **Table 1**.

Le descriptif complet de ces FAP se trouve dans le document à cette adresse [[WWW10](#)], dans l'annexe numéro 1.

Table 1 – Exemple de Facial Action Parameters

Numéro	Nom	Description
3	open_jaw	Déplacement vertical de la mâchoire (n'affecte pas l'ouverture de la bouche)
7	stretch_r_cornerlip	Déplacement horizontal du coin droit de la lèvre intérieure
10	raise_b_lip_lm	Déplacement vertical du point médian entre le coin gauche et le bas de la lèvre intérieure
42	lift_r_cheek	Déplacement vertical de la joue droite

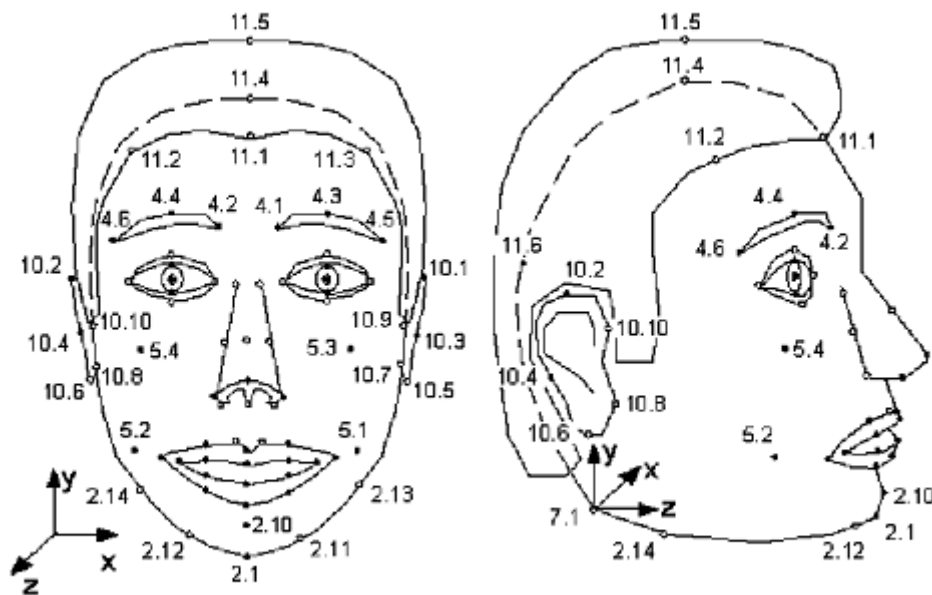


Figure 2 – Quelques Facial Features Points utilisés par la norme MPEG-4

Cependant, la norme de MPEG-4 est moins réaliste que FACS d'un point de vue musculaire.

Par exemple : l'AU 26 de FACS (« Jaw Drop ») décrit le mouvement d'abaissement du menton, cet abaissement est accompagné d'un abaissement de la lèvre inférieure. Or l'abaissement du menton de MPEG-4 (FAP 3 - open_jaw) ne décrit pas l'abaissement de la lèvre inférieure.

Nous allons maintenant voir quelles technologies sont disponibles pour réaliser l'acquisition des images qui seront à traiter par la suite.

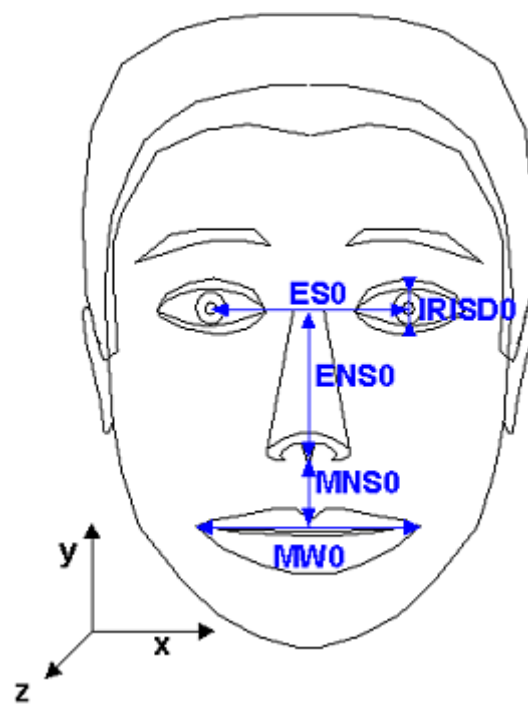


Figure 3 – Facial Animation Parameter Units utilisés par la norme MPEG-4

3

État de l'art

Dans ce chapitre, je vais vous présenter l'état de l'art des systèmes et techniques existants permettant de faire de la reconnaissance faciale d'émotion.

Mon projet se basant sur des images et vidéos en 2D, ce chapitre va principalement se concentrer sur les techniques d'acquisition et d'analyse d'images 2D.

Cependant, j'ai tout de même réalisé quelques recherches sur le matériel utilisé pour l'acquisition d'images en 3D ainsi que les bases de données disponibles et c'est donc par ces recherches que je vais entamer ce chapitre.

1 3D

Comme je l'ai expliqué plus haut, mon projet, et donc mon système de reconnaissance, utilisera des images 2D mais j'ai tout de même mené quelques recherches sur les images en 3D également.

Je vais tout d'abord vous présenter les techniques utilisées pour acquérir des images en 3D. Je vous exposerai ensuite certains matériels déjà existant permettant de faire de la capture d'images 3D et utilisant les techniques que je vous aurais présenté précédemment. Je terminerai enfin cette section sur la 3D en vous introduisant les bases de données de visages les plus connues et les plus utilisées par les systèmes de reconnaissance faciale d'émotion en 3D.

Les informations que vous allez trouver dans cette section proviennent en majorité de l'article [22].

1.1 Techniques d'acquisition

1.1.1 Reconstruction à partir d'une image

Il est possible, à partir d'une image en 2D capturée par une caméra basique, d'obtenir une image en 3D. La méthode la plus prometteuse est celle du **3D Morphable Model** (3DMM), qui consiste à apposer un visage en 3D (masque) sur l'image en 2D et de le modifier pour le faire correspondre avec l'image. Sont ensuite extraites les informations correspondant au masque modifié qui vont permettre de créer le visage de l'image en 3D.

La **Figure 1** présente un exemple de visage 3D récupéré depuis une image 2D.

Cette technique est très pratique et répandue car elle ne nécessite pas de matériel au coût exorbitant, une simple caméra est nécessaire.

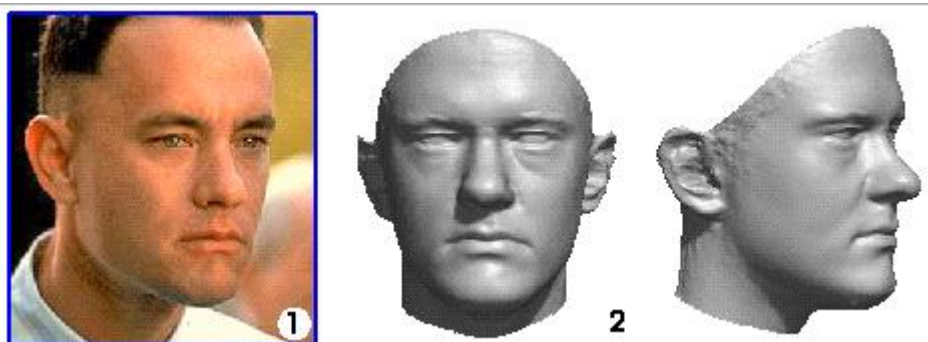


Figure 1 – Reconstruction d'un visage 2D (1) en 3D (2) grâce à la méthode 3DMM

1.1.2 Lumière structurée

Une autre technique, une des plus utilisées, est la technique de la lumière structurée.

Elle consiste à projeter plusieurs rais de lumière (visible ou infra-rouge) de longueur d'onde différente puis, à l'aide d'un capteur, de mesurer la déformation de ces rais de lumière pour construire le visage en 3D.

La **Figure 2** montre un exemple de lumière structurée.

L'utilisation de cette technique requiert d'avoir un matériel spécifique contenant un émetteur et un récepteur, cet outil peut aller de quelques centaines d'euros pour les moins chères à plusieurs dizaines de milliers d'euros pour les plus performantes.

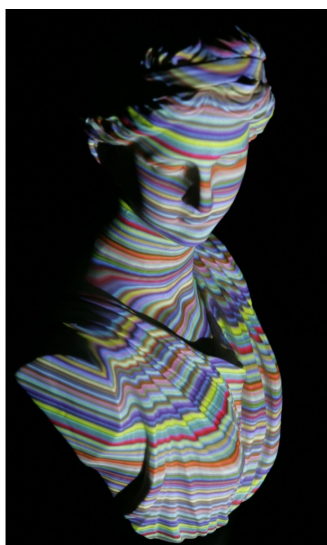


Figure 2 – Exemple de lumière structurée

1.1.3 Stéréo photométrie

La technique de la stéréo photométrie consiste à prendre plusieurs photos d'un même objet avec un même appareil sous différentes illuminations (lumière venant de droite, lumière venant de devant ...). Pour obtenir un visage en 3D, il ne reste qu'à assembler les photos obtenues.

La **Figure 3** montre un exemple de reconstruction d'un objet grâce à la stéréo photométrie.

Cette technique requiert également un équipement coûteux et n'est donc pas accessible à tout le monde.

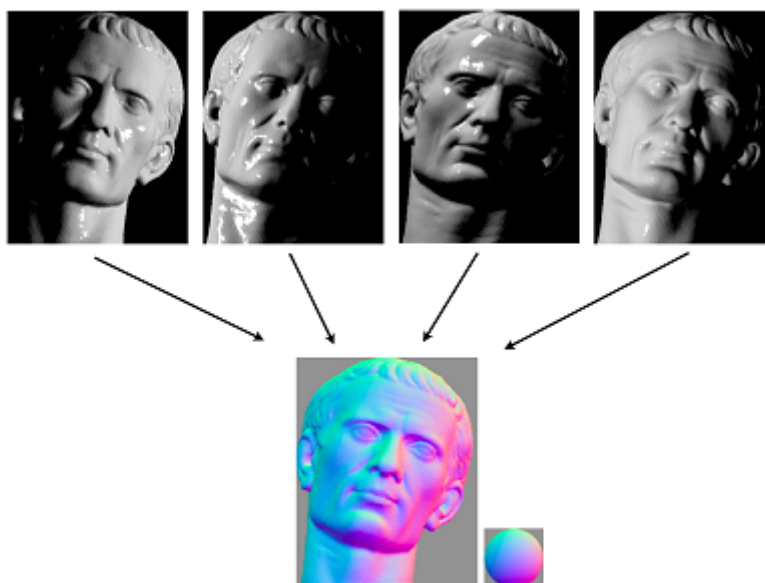


Figure 3 – Exemple de reconstruction d'un objet grâce à la stéréo photométrie

1.1.4 Stéréo multi-vue

C'est une technique très similaire à celle de la stéréo photométrie sauf qu'au lieu de prendre en photo un visage sous différentes illuminations, le visage est pris sous différents angles simultanément avec plusieurs appareils.

Cette technique requiert elle aussi un équipement spécifique coûteux.

1.2 Dispositifs d'acquisition d'images

Plusieurs types de dispositifs différents existent à l'heure actuelle permettant de capturer des images en 3D. La plupart d'entre eux se basent sur les techniques présentées précédemment.

Je vais ici vous présenter les deux dispositifs les plus connus.

1.2.1 Kinect

Probablement la caméra 3D la plus connue du grand public : la Kinect de Microsoft.

Créer initialement pour créer une immersion plus poussée pour les jeux de la console Xbox 360, elle a depuis été améliorée avec sa version Xbox One et ne sert plus exclusivement qu'à jouer aux jeux vidéos. Elle utilise la technique de la lumière structurée (infra-rouge dans ce cas) pour capturer les images en 3D de ce qu'elle filme.

La Kinect reste cependant une caméra 3D low-cost, de mauvaise qualité lorsqu'on le compare à d'autres dispositifs du même type, tel que celui que je vais présenter maintenant.

1.2.2 Minolta Vivid 910

Un autre dispositif très utilisé et utilisant lui aussi la technologie de la lumière structurée est le Minolta Vivid 910.

Un comparatif entre la qualité de Kinect et de Minolta est présenté en [Figure 6](#). On s'aperçoit très clairement du gouffre séparant ces deux dispositifs. Bien sûr le prix n'est pas le même, car nous passons de quelques centaines d'euros pour la Kinect à plusieurs dizaines de milliers d'euros pour le Minolta Vivid 910.



Figure 4 – *Version 2 de la Kinect de Microsoft*



Figure 5 – *Minolta Vivid 910*



Figure 6 – *Comparatif de la qualité entre Kinect et Minolta*

1.3 Base de données de visages 3D

Les dispositifs tels que le Minolta Vivid 910 présentés précédemment permettent également la création de bases de données de visage en 3D. Leurs grandes qualités permettent d'obtenir des images très précises, facilement exploitable.

Je vais maintenant vous présenter les bases de données les plus connues et pouvant être utilisée par la communauté scientifique.

1.3.1 BU-3DFE

Les premiers efforts pour récolter des données en 3D ont menés à la création de la base de données BU-3DFE (Binghamton University 3D Facial Expression [32]).

Les images contenues dans cette base sont statiques et ont été capturées par le dispositif 3dMD.

Elles se composent de 100 sujets âgés de 18 à 70 ans, dont 56% de femmes, et appartenant à différentes ethnicités.

Chaque sujet réalise les 7 expressions basiques (cf Section 1 (Chapitre 2)) et chaque expression, sauf l'expression neutre, est réalisée suivant 4 niveaux d'intensités. Pour chaque sujet, il y a donc 25 images différentes, ce qui nous donne au total 2500 images dans cette base de données.

Un exemple de données contenues dans cette base sont présentés en Figure 7.

Chaque donnée contient également la position de 83 points clés du visage.



Figure 7 – Exemple de données contenues dans BU-3DFE

1.3.2 BU-4DFE

Une extension de la base BU-3DFE a été réalisée dans le but d'obtenir un espace 3D dynamique, c'est-à-dire rajouter la dimension du temps dans les images de la base, en plus des 3 dimensions déjà présentes.

En effet, dans la version de base, il n'y avait qu'une seule image présente pour une expression et une intensité, mais dans le but d'obtenir des analyses plus performantes, inclure la notion du temps dans ces images devient indispensable.

La base de données BU-4DFE [31] se compose donc de 101 sujets (58 femmes) de différentes ethnicités, chaque sujet réalisant 6 des expressions basiques (toutes sauf la neutre). Chaque séquence d'émotion contient environ 100 frames, ce qui nous permet d'obtenir environ 60600 différentes frames dans la base.

1.3.3 Bosphorus 3D Face Database

La base de données Bosphorus [23] est composée de 105 sujets (45 femmes) dont la plupart sont de type Caucasien et dont un tiers sont des acteurs professionnels.

Chaque sujet réalise environ 35 expressions et toutes les images sont codées en terme de FACS (Section 2 (Chapitre 2)).

Plusieurs illuminations et occlusions du visage (barbe, moustache, lunettes...) sont également présentes pour chaque sujet. 24 points clés du visage sont également définis pour chaque donnée.

Un exemple de donnée est présente en Figure 8

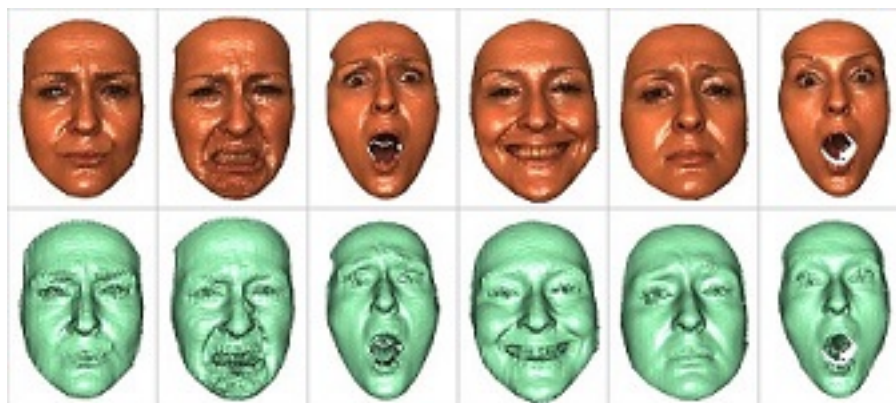


Figure 8 – Exemple de données contenues dans la base de données Bosphorus

1.3.4 Comparatif des différentes bases de données 3D

Précédemment, je ne vous est présenté que les bases de données les plus populaires. Cependant, beaucoup d'autres existent également.

J'ai donc réalisé un tableau (Table 1) recensant ces différentes bases de données publiquement disponible, et pouvant représenter un intérêt dans la reconnaissance d'émotions, avec leurs principales caractéristiques (données principalement tirées de [22]).

Table 1 – Comparaison de bases de données 3D

Nom	Type de données	Taille	Contenu
BU-3DFE[32]	Statique	100 adultes	6 émotions basiques avec 4 niveaux d'intensités
BU-4DFE[31]	Dynamique	101 adultes	6 émotions basiques
Bosphorus[23]	Statique	105 adultes (dont 25 acteurs)	6 émotions basiques, 24 AUs, occlusions
ICT-3DRFE[26]	Statique	23 adultes	6 émotions basiques, 2 expressions neutres, 4 orientations du regard (haut, bas, gauche, droite) et un visage "grimaçant"
D3DFACS[4]	Dynamique	10 adultes (dont 4 experts en FACS)	Jusqu'à 38 AUs par sujets
Gavabdb[18]	Statique	61 adultes	3 expressions : sourires ouverts/fermés et aléatoire

2 2D

Passons maintenant à ce qui va m'être utile à la réalisation de mon projet : la 2D.

Dans cette section, je présenterai tout d'abord rapidement les dispositifs permettant de récupérer des images en 2D, puis plusieurs bases de données de visages 2D et enfin des méthodes permettant de réaliser de la reconnaissance faciale d'expression.

2.1 Dispositifs d'acquisition d'images 2D

Contrairement à la 3D, il n'est pas nécessaire d'avoir des dispositifs extrêmement coûteux pour acquérir des images en 2D.

En effet, un simple appareil photo ou une simple caméra trouvés dans n'importe quelle commerce suffit amplement.

Plusieurs entreprises se sont spécialisées dans le domaine de reconnaissance d'émotions et permettent, par exemple, aux publicitaires de tester leur publicités et d'avoir un feedback sur ce que ressentent les spectateurs. Pour cela, ces entreprises([WWW2],[WWW1]) utilisent la webcam intégrée dans les ordinateurs pour ensuite traiter les images.

Dans le cadre de ce projet, nous utiliserons une caméra PTZ (Pan, Tilt, Zoom). Ces caméras permettent de faire une rotation selon l'axe Z (Pan), une rotation selon l'axe X (Tilt) et de zoomer selon l'axe Y. Souvent utilisé en temps que caméra de surveillance, dans le cadre de ce projet, ce type de caméra va nous permettre de se déplacer pour trouver la personne présente dans la pièce puis de zoomer sur son visage pour pouvoir ensuite l'analyser.

Un exemple de caméra PTZ est présenté en Figure 9.



Figure 9 – Exemple de caméra PTZ

Dans le cas d'images statiques (images, photos), la reconnaissance se fera directement sur l'image. Par contre dans le cas d'un flux vidéo récupéré via une caméra, c'est les *frames* (images constituant une vidéo) de la vidéo qui vont être analysées.

2.2 Bases de données de visages 2D

Je vais ici vous présenter différentes bases de données de visages 2D, libre d'accès à la communauté scientifique, intéressantes pour la recherche et permettant de réaliser une reconnaissance faciale d'expressions.

2.2.1 Cohn-Kanade (CK) [10]

Probablement la base de données de visage 2D la plus connue et la plus utilisée pour la reconnaissance faciale d'expressions, elle se compose de 97 sujets (65% femmes, 15% afro-américains et 3% asiatiques ou sud américain) âgés de 18 à 30 ans et réalisant les 6 expressions universelles (joie, tristesse, dégoût, peur, surprise et colère).

Cette base contient au total 486 séquences vidéos, du visage neutre au départ jusqu'à l'**apex** (le pic de l'émotion) à la fin. Ces séquences sont en niveaux de gris et digitalisé en tableaux de 640*480 pixels avec donc une précision de 8 bits (dû aux niveaux de gris).

Toutes ces séquences sont entièrement codées en termes de FACS et d'AUs mais ne contiennent pas de label spécifiant l'émotion présentée.

Toutes les expressions présentes dans les séquences vidéos contenues dans cette base sont posées et non spontanées, cela signifie qu'on a demandé à ces personnes de produire telle ou telle émotion, cela ne leur est pas venu d'eux-même spontanément. Les expressions présentées seront donc plus "exagérées" qu'en temps normal.

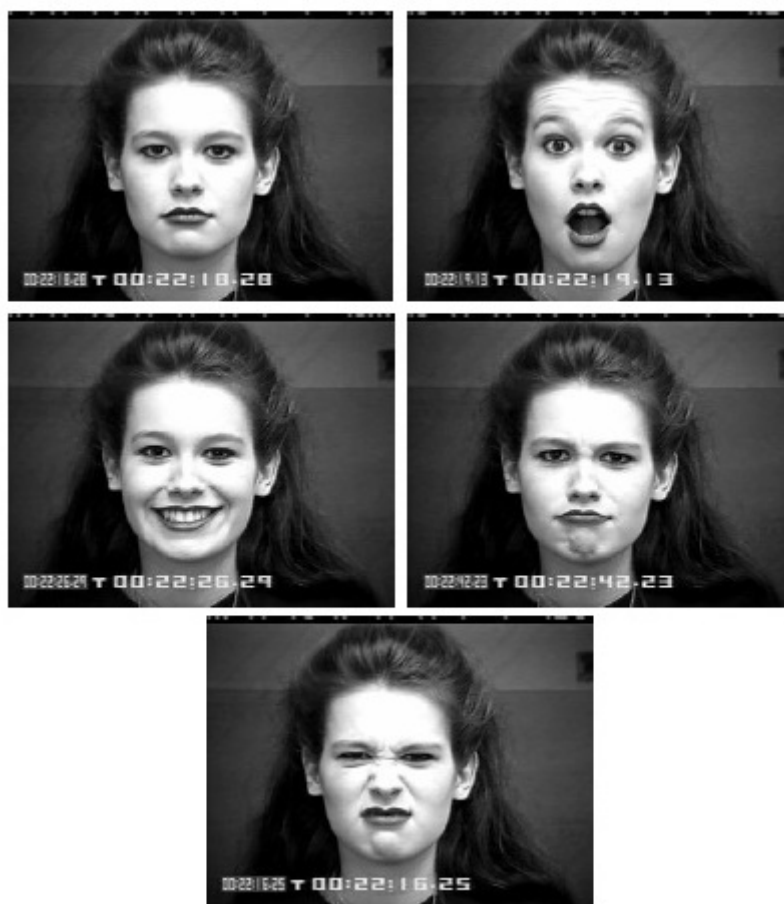


Figure 10 – Exemple de données de CK (de haut en bas et de gauche à droite : neutre, surprise, joie, colère, dégoût)

2.2.2 CK+ [16]

C'est la version améliorée de la base de données CK.

Cette fois-ci, elle contient des expressions posées et spontanées.

Dans le cas des expressions posées, le nombre de sujets a été augmenté 27% et le nombre des séquences de 22%. Les séquences sont toujours codées en termes de FACS et d'AUS avec, cette fois-ci, un label associé à l'expression présentée qui se trouvera dans les metadata de la vidéo.

Cette version propose également des protocoles et des résultats pour le suivi des points clés du visage ainsi que pour la reconnaissance d'émotion.

Actuellement, une troisième version de CK est en préparation, avec comme principale ajout la synchronisation entre une séquence frontale et une séquence orientée de 30 degrés par rapport à la vue frontal.

2.2.3 Man-Machine Interaction (MMI) [27][20]

Créée en 2002, cette base de données, comme CK+, se compose de deux parties : une partie posée et une partie spontanée.

Cette base contient plus de 2900 vidéos ainsi que des images en haute résolution de 75 sujets (48% de femmes, européen, africain et sud américain) âgés de 19 à 62 ans. Chaque séquence vidéo contient soit une expression dans son entièreté, soit un AU spécifique.

Les données de cette base sont également codées en termes de FACS et d'AUs.

L'avantage de cette base de données est qu'elle contient toute l'étendue d'une expression : visage neutre -

onset (début de l'expression) - **apex** (pic) - **offset** (fin de l'expression) - visage neutre .
Cela permet de réaliser une reconnaissance d'émotion dans le temps plus précise.
Certaines des séquences vidéos sont en couleur, les autres sont en niveaux de gris.



Figure 11 – Exemple de données de MMI

2.2.4 MHI Mimicry [3] [13]

Cette base de données est intéressante car elle diffère totalement des deux précédentes au niveau de sa construction.

Elle se compose de 54 vidéos (sessions) de 40 sujets (28 hommes, 12 femmes) entre 18 et 40 ans venant de l'Imperial College de Londres. Sur chaque session, 2 participants interagissent et chaque session est divisée en deux parties.

Tout d'abord une partie débat dans laquelle les deux participants vont discuter de politique. Ils seront donc soit du même avis, soit d'un avis opposé.

La deuxième partie consiste en un "jeu de rôle" : un participant joue le rôle d'un étudiant cherchant un appartement et le deuxième participant joue le rôle d'un propriétaire d'appartement. Le but de l'étudiant est donc de trouver un appartement et celui du propriétaire de louer son appartement, ils ont donc un but qui les relie.

Cela permet d'observer le comportement humain lors d'une discussion où l'on est d'accord ou non et lorsque l'on veut convaincre quelqu'un, comment nous "mimons" ou non le comportement de notre interlocuteur dans le but de montrer nos intentions, de nous faire accepter.

Chaque session est divisée en "épisodes d'intérêt" et chaque épisode est labellisé en fonction de ce qu'on y voit : sourire, hochement de tête, penchement du corps en avant ou en arrière.

Sont aussi labellisés le rôle de chaque participant (s'il parle ou s'il écoute) ainsi que leurs intentions (2 catégories) :

- Social Signal Expression (inconscient) : compréhension, accord, confusion, "liking"
- Desired Goal (conscient) : flatter l'autre, souligner la compréhension, exprimer l'accord, partager l'empathie, augmenter l'acceptation.

Tous les enregistrements ont été réalisés avec 15 caméras (7 par participants et une vue d'ensemble) et 3 micros (un micro de tête par participant et un au milieu de la pièce). Un exemple des images récupérées par les caméras se trouvent en [Figure 12](#)

Cependant, sur cette base de données, aucune notion de FACS ni d'AUs n'est présente, les données sont labellisées en fonction du ressenti et de l'auto-évaluation réalisée par les participants..

2.2.5 HCI Tagging [24] [12]

C'est avec cette base de données que j'ai travaillé à la réalisation de mon projet.



Figure 12 – Exemple de données de MHI Mimicry

Créée par la même équipe que celle qui a créé MHI Mimicry, cette base de données se compose de 27 participants (11 hommes, 16 femmes) de 19 à 40 ans.

Elle est divisée en 2 parties : la première partie contient 20 vidéos et la deuxième 28 vidéos et 14 images. Chaque vidéos contient la vidéo du visage, l'audio et les expressions vocales, la position où regardent les yeux et les signaux physiologiques (température du corps, rythme de la respiration, EEG, rythme cardiaque). Les vidéos ont été filmées via 6 caméras : 5 en niveaux de gris et d'orientations différentes (frontal, bas-gauche, bas-droite, vue d'ensemble et profil gauche) et une en couleur (frontal).

Un exemple des différentes données vidéos contenues dans cette base est en [Figure 14](#).

Concernant la partie 1 (Explicit Tagging) : chaque participant regarde plusieurs vidéos et à la fin de chaque vidéo un feedback sur leur ressenti leur a été demandé :

- émotion ressentie (parmi neutre, anxieux, amusé, triste, joyeux, dégoûté, en colère, surpris ou apeuré)
- excitation (sur une échelle de 1 à 9)
- agréabilité (sur une échelle de 1 à 9)
- dominance (sur une échelle de 1 à 9)
- prédictibilité (sur une échelle de 1 à 9)

Les vidéos visionnées par les participants lors de la partie 1 sont séparées par de petits clips courts neutre pour remettre l'expression de la personne à 0, la réinitialiser en quelque sorte.

Concernant la partie 2 (Implicit Tagging) : on diffuse à chaque participant une image ou une vidéo deux fois. La première fois sans rien puis la deuxième fois avec un *tag* censé décrire ce qu'il se passe dans la vidéo/image. Soit ce tag est correct soit il est erroné, le participant doit donc dire si il est d'accord ou non avec le tag attribué en appuyant respectivement sur un bouton vert ou rouge. Pour illustrer cela, un exemple est présent en [Figure 13](#). Cependant, comme les créateurs de cette base n'ont pas les droits pour ces images, les images accessibles que j'ai récupéré ne contiennent que les bords contrairement aux images diffusées aux participants.

Pareillement à MHI Mimicry, aucune notion de FACS ni d'AUs n'est présente, les données sont labellisées en fonction de l'auto-évaluation réalisée par tous les participants.

2.2.6 Comparatif des bases de données 2D

Pour récapituler toutes les informations que j'ai pu donner précédemment et pour introduire de nouvelles bases de données accessibles à la communauté scientifique dont je n'ai pas parlé, j'ai donc réalisé le tableau comparatif suivant :

Malheureusement, aucune base de données existantes ne concernent les personnes handicapées.



Figure 13 – Même image avec 2 tags différents : le premier erroné (Kiss), le deuxième correct (Handshake)



Figure 14 – Données contenues dans HCI Tagging

Table 2 – Comparaison de bases de données 2D (P/S : Posée/Spontanée)

Nom	P/S	Taille	Contenu	Label
CK[10]	P	97 sujets (65% femmes, 15% afro-américains et 3% asiatiques ou sud américain) de 18 à 30 ans	486 séquences vidéos des 6 émotions basiques	FACS
CK+[16]	P/S	Nombre de sujets augmenté 27%	Nombre de séquences augmenté 27%	FACS + émotion
MMI[27] [20]	P/S	75 sujets (48% de femmes, européen, africain et sud américain) de 19 à 62 ans	2900 vidéos des 6 émotions basiques ou d'AUs spécifiques	FACS
MHI Mimicry [3][13]	S	40 sujets (28 hommes, 12 femmes) entre 18 et 40 ans	54 vidéos (sessions)	Auto-évaluation
HCI Tagging [24][12]	S	27 sujets (11 hommes, 16 femmes) de 19 à 40 ans	Partie 1 : 20 vidéos ; Partie 2 : 28 images et 14 vidéos	Auto-évaluation
SAL [WWW9]	S	24 sujets	10 heures de vidéo : les sujets parle à une intelligence artificielle et leurs émotions sont changées en fonction des différentes personnalités de l'IA	Feeltrace
JAFFE[17]	P	10 femmes japonaises	213 images des 7 émotions basiques	Noté selon 6 adjectifs d'émotion par 60 sujets Japonais

4

Architecture d'un système de reconnaissance d'émotions

Dans ce chapitre je vais introduire les différentes parties nécessaires à la construction d'un système permettant de reconnaître des expressions et émotions.

Un tel système se constitue de 3 parties :

- Détection du visage
- Extraction des *features* (nez, bouches, yeux ...)
- Classification

Chacune de ces parties sera décrite plus précisément dans les sections suivantes avec un état de l'art sur les techniques existantes pour chacune d'entre elles.

Une partie des informations se trouvant de ce chapitre sont tirées de [30] écrit par les fondateurs de la société Emotient et qui présente un état de l'art des différentes parties d'un système de reconnaissances d'émotions.

1 Détection du visage

Depuis déjà plusieurs années, la détection de visage dans une image ou vidéo est devenu une réalité grâce aux algorithmes d'apprentissage.

La détection de visage sur une vidéo ou une image se retrouve de plus en plus dans notre quotidien. Tout d'abord dans nos appareils photos ou smartphones mais également sur les réseaux sociaux comme Facebook, qui repère les visages sur une photos lorsque l'on souhaite identifier des personnes, ou encore Snapchat, qui depuis la dernière mise à jour repère notre visage grâce à la caméra frontale d'un smartphone pour ensuite lui appliquer diverses animations et déformations.

Les algorithmes de détection de visage se divisent en 2 catégories : les absolus et les différentiels, chacun ayant leurs avantages et inconvénients.

1.1 Absolu

Les détecteurs absolus, aussi appelés détecteurs *frame-by-frame*, vont déterminer la position d'un visage sur chaque frame d'une vidéo, indépendamment des frames précédentes.

Le principal avantage de ces détecteurs est qu'ils sont très facilement parallélisable sur plusieurs frames d'une vidéo. Ils permettent également de réagir très rapidement si jamais le nombre de visages dans l'image change subitement et ne "dérive" pas au court du temps. Ici le terme "dériver" fait référence au fait de perdre la position d'un visage.

Cependant, comme dit précédemment, ces détecteurs n'utilisent pas de notions de temps qui pourraient

les rendre plus rapide et précis.

Le tout premier algorithme de détection qui est maintenant utilisé par la plupart de ces détecteurs est l'algorithme de **Viola-Jones** [28] créé en 2001.

Cet algorithme va tout d'abord apprendre un classificateur à différencier des visages d'autres objets grâce à un apprentissage réalisés avec des images de différentes tailles de visages ou de divers autres objets. A noter qu'il est préférable d'avoir un nombre d'objets quelconque très supérieur au nombre de visages si nous voulons que l'algorithme soit efficace.

Une fois l'apprentissage fini et que l'on passe une nouvelle image à l'algorithme, il va analyser cette dernière en extrayant plusieurs *patches*, des bouts de l'image, de différentes tailles qu'il va ensuite normaliser à une taille précise puis les donner au classificateur qui va se charger de définir si oui ou non se trouve un visage dans ce patch.

Cette étape d'extraction de patches peut également être paralléliser pour gagner en rapidité.

1.2 Différentiel

Contrairement aux détecteurs absolus, les détecteurs différentiels, aussi appelés *face trackers*, déterminent la position d'un visage sur une image grâce à sa position précédente. Si la position d'un visage est connu à l'instant t , le détecteur différentiel va se servir de cette position pour trouver celle à l'instant $t+1$.

Bien sûr, il faut initialiser la position à l'instant 0 pour faire fonctionner ces détecteurs. Pour cela il est possible d'utiliser un détecteur absolu sur la toute première frame du flux vidéo.

L'avantage de ces détecteurs est leur grande rapidité et précision. Cependant, l'inconvénient est que l'accumulation de petite erreurs sur les positions peut mener à la "dérive" du détecteur puisqu'il ne se remet jamais à 0 une fois lancé.

L'un des plus connus est l'algorithme **Active Appearance Model** (AAM).

Dans AAM, un visage est représenté comme un modèle en forme de maillage triangulaire composé d'environ 70 points. Ce modèle est construit grâce à un apprentissage sur différents visages réalisant différentes expressions et dans lesquels les *features* sont connus. Il va donc chercher à faire correspondre ce modèle avec tout visage se trouvant sur une image en le déformant pour faire correspondre les features du modèle à celles du visage. Les déformations possibles ont été calculé au préalable.

Sur la première frame de la vidéo, il suffit d'initialiser les positions des features de chaque visage (manuellement ou via un tracker absolu) puis, sur les frames suivantes, la position de ces features est trackée grâce aux différentes déformations possibles.

AAM ne permet pas que de détecter des visages mais permet également d'en extraire les features.

1.3 Comparatif

Voici un comparatif avantages/inconvénients de ces 2 types de détecteurs.

Table 1 – Avantages/inconvénients des 2 types de détecteurs

	Avantages	Inconvénients
Absolu	Facilement parallélisable, très réactif en cas de changement soudain du nombre de visage et ne "dérive" pas	Plus lent et moins précis que les différentiels
Différentiel	Très rapide et d'une grande précision	Peut "dériver" si de petites erreurs s'accumulent au fur et à mesure

2 Extraction des *features*

La deuxième partie d'un système de reconnaissance faciale d'émotions est l'extraction des *features* du visage précédemment repéré. Ces features sont les points clés d'un visage et se composent par exemple

de la position du coin des yeux, des lèvres, du nez, des joues, de la position de la pupille, etc. Pour extraire ces points et leurs positions, plusieurs algorithmes existent et je vais maintenant vous en expliquer quelques uns.

2.1 Filtres de Gabor

Dans le domaine de la vision par ordinateur, les filtres de Gabor sont principalement utilisés dans l'analyse de textures, la détection de contour et l'extraction d'éléments. C'est un filtre linéaire dont la réponse va être une sinusoïde modulée grâce à une fonction gaussienne.

Son fonctionnement est très simple : tout d'abord une fonction de Gabor est générée. Ensuite cette fonction sera appliquée à chaque pixel de l'image. Les résultats obtenus pour chaque pixel seront ensuite stockés et résulteront en la création d'une nouvelle image "filtrée".

Une fonction de Gabor peut s'écrire de 3 façons différentes.

Tout d'abord, seulement en forme réelle :

$$G(x, y; \lambda, \theta, \Psi, \sigma, \gamma) = e\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) * \cos\left(2\pi \frac{x'}{\lambda} + \Psi\right) \quad (1)$$

Ou seulement en forme imaginaire :

$$G(x, y; \lambda, \theta, \Psi, \sigma, \gamma) = e\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) * \sin\left(2\pi \frac{x'}{\lambda} + \Psi\right) \quad (2)$$

Ou bien en combinant (1) et (2) pour obtenir la fonction sous forme complexe :

$$G(x, y; \lambda, \theta, \Psi, \sigma, \gamma) = e\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) * e\left(i\left(2\pi \frac{x'}{\lambda} + \Psi\right)\right) \quad (3)$$

Avec :

$$\begin{aligned} x' &= x \cos \theta + y \sin \theta \\ y' &= -x \sin \theta + y \cos \theta \end{aligned}$$

Et avec comme paramètres :

- x et y : les coordonnées du pixel
- λ : la longueur d'onde du filtre, en pixel
- θ : l'orientation du filtre, en degré
- Ψ : le décalage de phase (phase offset), en degré
- σ : l'écart type de la gaussienne
- γ : le rapport d'aspect spatial (spatial aspect ratio)

Un dernier paramètre qui peut se révéler utile et ne se trouvant pas directement dans la formule de la fonction est le nombre d'orientation n . Ce paramètre va permettre de générer un banc de filtres de différentes orientations en faisant varier le paramètre θ qui prendra comme valeur $\theta + \frac{i\pi}{n}$ avec $i \in [0, n-1]$. Grâce à un site internet permettant de simuler des filtres de Gabor ([WWW8]) et au mode d'emploi associé ([WWW7]), j'ai pu voir l'effet qu'ont les différents paramètres sur le résultat final.

Tout d'abord, pour que mes "expériences" soient cohérentes, il fallait fixer une valeur par défaut pour chaque paramètre ainsi qu'une image de base. Les paramètres, lorsque je ne les modifierai pas, seront donc fixés à : $\lambda = 8$, $\theta = 0$, $\Psi = 0$, $\sigma = 4.48$ et $\gamma = 0.5$ et $n = 8$; quant à l'image de base, ce sera la célèbre photo de Lena (Figure 1).

J'ai donc fait varier chacun de ces paramètres, un seul à la fois, pour voir l'effet produit sur le résultat. Tous mes résultats, que je vais maintenant expliquer, sont présentés dans les figures de Figure 2 à Figure 7.

J'ai tout d'abord commencé par faire varier λ (Figure 2). Ce paramètre va permettre de faire varier la taille de notre filtre, ce qui va permettre de repérer des bords plus ou moins fins. En consultant le



Figure 1 – Image utilisée pour vérifier l'effet des filtres de Gabor

mode d'emploi du simulateur, on apprend que sa valeur doit être supérieure ou égale à 2 et qu'elle doit être plus petit qu'un cinquième de la taille de l'image pour que le filtre fonctionne correctement.

Puis j'ai fait varier θ (Figure 3). En variant l'orientation, cela va permettre au filtre de ne pas capter les mêmes contours : il ne va capter que les contours ayant la même orientation. C'est pour cela que l'on utilise un banc de filtre et non pas un seul filtre tout seul car sinon nous ne capturerons que des contours ayant la même orientation (vertical par exemple si $\theta = 0^\circ$). Ses valeurs sont comprises entre 0 et 360.

En modifiant Ψ (Figure 4), cela va changer la disposition du filtre, sa "symétrie". Les valeurs sont comprises entre -180 et 180. Les valeurs 0 et 180 correspondent respectivement aux symétries centrées "center-on" et "center-off", et les valeurs -90 et 90 correspondent aux dissymétries associées. Ce paramètre permet de changer "l'emplacement" des contours, comme montré sur la figure associée.

Le paramètre γ permet de définir l'ellipticité de la fonction de Gabor (Figure 5). Ses valeurs sont réelles et si $\gamma = 1$, alors le filtre sera de forme ronde. Plus ce paramètre est élevé, plus le filtre sera "plat". Modifier l'ellipticité permet d'avoir des contours plus ou moins précis.

En modifiant le paramètre n (Figure 6), comme je l'ai expliqué précédemment, cela aura pour conséquence de créer plusieurs filtres d'orientation différentes. Comme un filtre ne permet de trouver que les contours ayant la même orientation, avoir un banc de filtres assez fourni permet de trouver tous les contours d'une image.

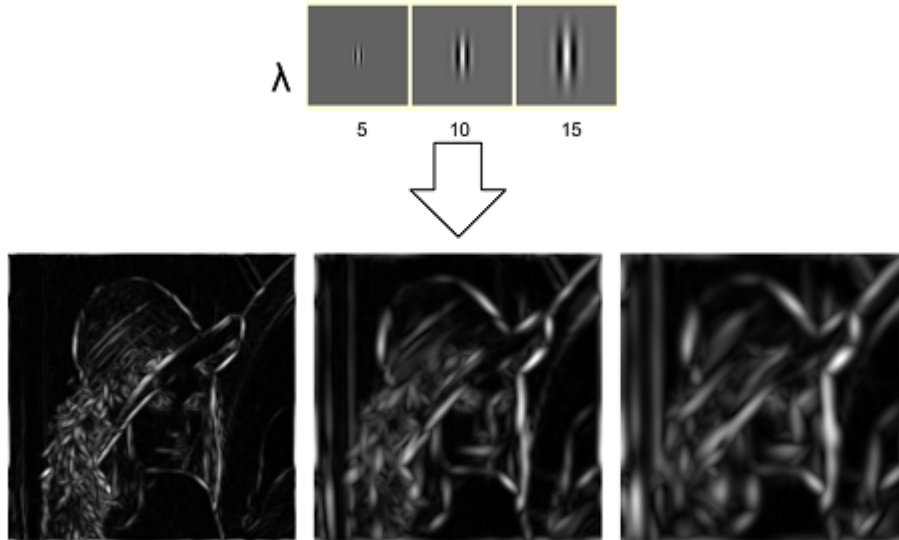
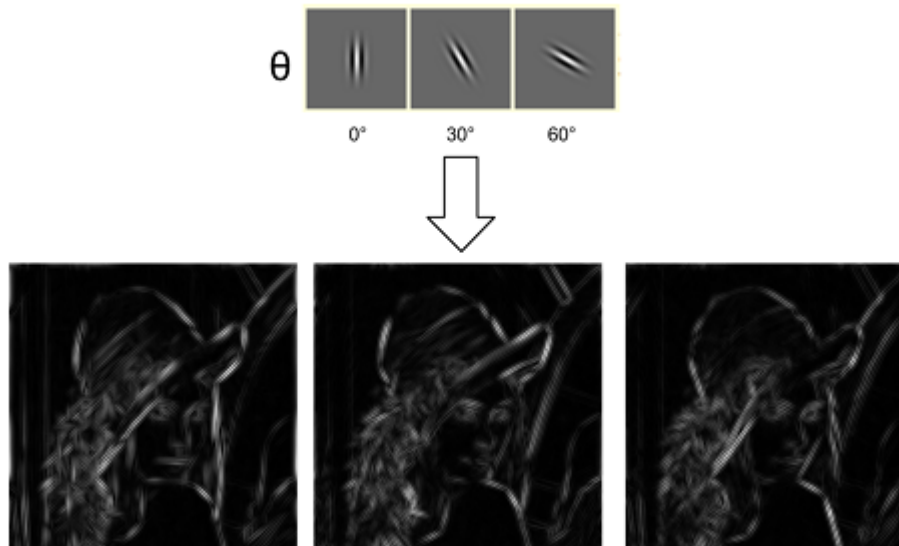
J'ai enfin terminé par le paramètre σ (Figure 7). Au départ, ce paramètre est inconnu et il se calcule grâce à un nouveau paramètre, b , correspondant à la largeur de la bande (en octave). On sait que :

$$b = \log_2 \left(\frac{\frac{\sigma}{\lambda} \pi + \sqrt{\frac{\ln 2}{2}}}{\frac{\sigma}{\lambda} \pi - \sqrt{\frac{\ln 2}{2}}} \right) \Leftrightarrow \frac{\sigma}{\lambda} = \frac{1}{\pi} \sqrt{\frac{\ln 2}{2}} * \frac{2^b + 1}{2^b - 1}$$

En définissant une valeur pour b , on définit donc la valeur de σ . Lorsque l'on choisit $b=1$, on obtient $\sigma = 0.56\lambda$. Plus b est petit, plus σ est grand. Dans mes expériences j'ai fixé b à 1, ce qui m'a donc donné $\sigma = 0.56 * 8 = 4.48$.

2.2 Composantes pseudo-Haar

Les composantes pseudo-Haar ont été créées par Paul Viola et Michael Jones en 2001 lors de la conception de leur détecteur de visage (cf [subsection 1.1](#) et [28]). Cette méthode se base sur les travaux de Papageorgiou et al. ([21]) qui décrivent des caractéristiques construites à partir d'ondelettes de Haar (d'où leurs noms

Figure 2 – Effet du changement du paramètre λ Figure 3 – Effet du changement du paramètre θ

de pseudo-Haar).

Les composantes utilisées par Viola et Jones sont présentées en [Figure 8](#).

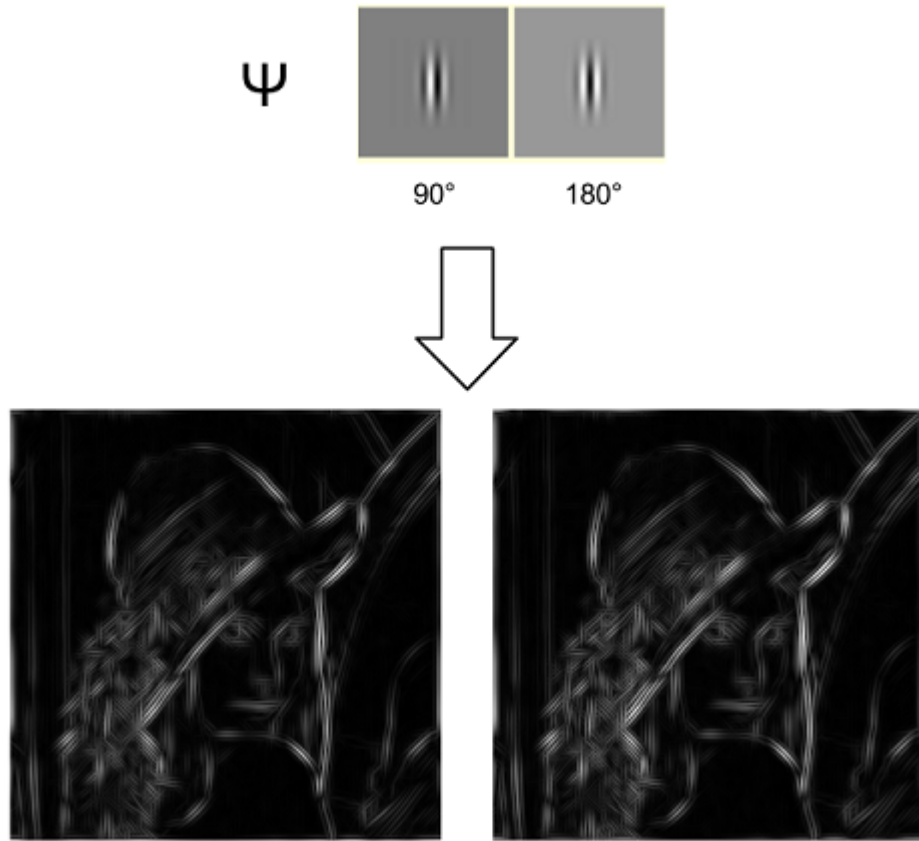
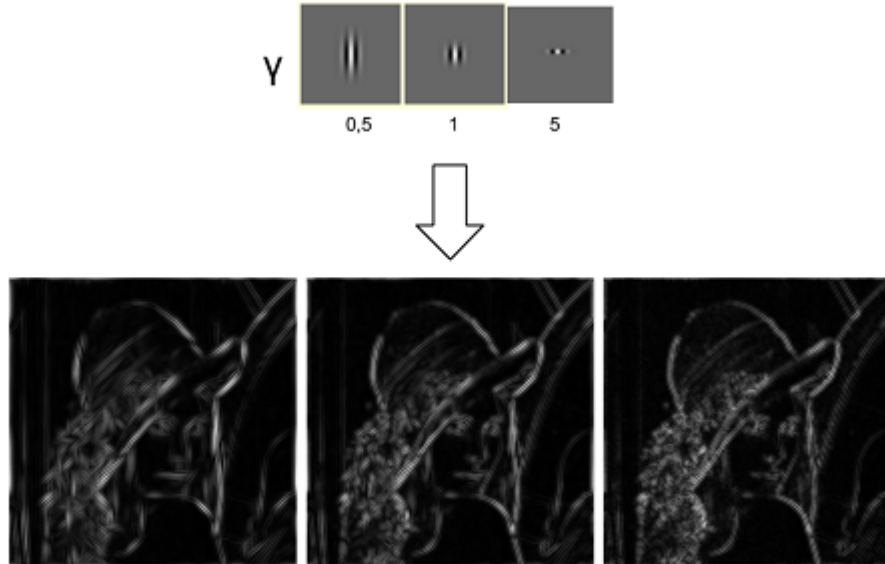
Ces composantes sont donc des masques, également appelées fenêtres de détection, composées d'un différent nombre de rectangle.

Leur utilisation est également très simple. Ces composantes sont appliquées à toutes les positions possibles sur l'image, tout d'abord de petite taille (20*20 pixels par exemple) puis en les agrandissant. A chaque position, on soustrait la somme des pixels contenus dans la partie noire à la somme des pixels contenus dans la partie blanche. Le résultat d'une caractéristique à une certaine position est donc un nombre réel qui va coder les variations du contenu pixellique.

Cette méthode est d'une grande rapidité lorsqu'elle est utilisée avec des **images intégrales**.

Une image intégrale est une image dérivée d'une image originale, de la même taille que cette dernière, et dont la valeur de chaque pixel est la somme des pixels au-dessus et des pixels à gauche du pixel courant de l'image originale. Sous forme mathématique, cela se traduit de cette façon :

$$imgInt(x, y) = \sum_{x' \leq x, y' \leq y} imgOri(x', y')$$

Figure 4 – Effet du changement du paramètre Ψ Figure 5 – Effet du changement du paramètre γ

Grâce à ces résultats sous forme de correspondance, il est même possible de calculer la valeur de chaque pixel de façon récursive pour gagner en rapidité de la façon suivante :

$$s(x, y) = s(x, y - 1) + \text{imgOri}(x, y)$$

$$\text{imgInt}(x, y) = \text{imgInt}(x - 1, y) + s(x, y)$$

avec $s(x, y)$ correspondant à la somme cumulée de la ligne x jusqu'à la colonne y .

Une fois l'image intégrale complètement calculée, avoir la somme de n'importe quel rectangle de

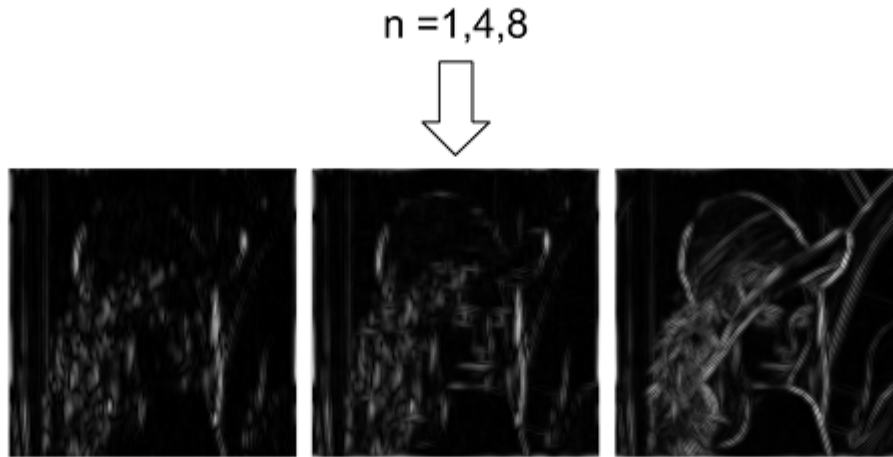
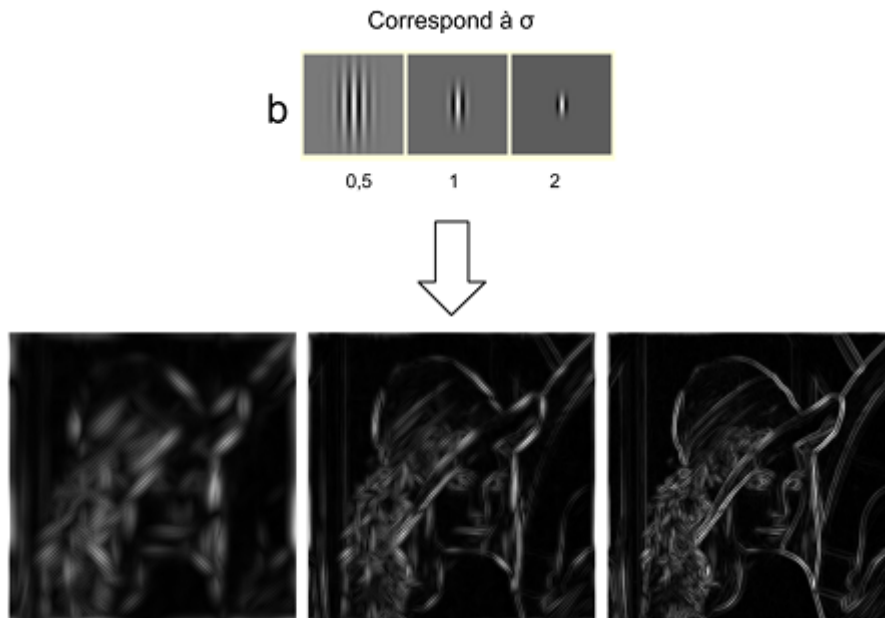
Figure 6 – Effet du changement du paramètre n Figure 7 – Effet du changement du paramètre b associé à σ 

Figure 8 – Composantes pseudo-Haar utilisé par Viola et Jones

l'image d'origine s'obtient en seulement 4 accès à l'image intégrale. En prenant un rectangle ABCD, cette somme s'obtient de cette manière :

$$\sum imgOri(ABCD) = imgInt(A) + imgInt(C) - (imgInt(B) + imgInt(D))$$

L'année suivante, en 2002, Lienhart et Maydt ([14]) ont créé une extension aux composantes créées par Viola et Jones. Ils ont créés des masques orientés de 45° , ce qui a pour conséquence d'améliorer d'environ 10% les performances par rapport à l'utilisation seule des masques créés par Viola et Jones. Ces masques permettent désormais de repérer des bords, des lignes ainsi que des "centres", telle que la pupille d'un oeil par exemple. Par contre, ils ont décidé de ne pas utiliser un des masques de Viola et Jones, le masque en forme de "damier". J'ai regroupé ces nouveaux masques dans la Figure 9.

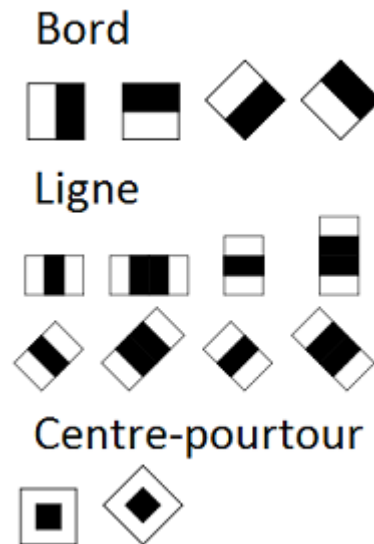


Figure 9 – Extension de composantes pseudo-Haar défini par Lienhart et Maydt

Pour aller de paire avec ces nouveaux masques et l'utilisation des images intégrales, une nouvelle formule est nécessaire qui calcule maintenant la somme dans un demi-rectangle orienté de 45° :

$$imgInt(x, y) = \sum_{x' \leq x, x' \leq x - |y - y'|} imgOri(x', y')$$

2.3 Motifs binaires locaux

La méthode des motifs binaires locaux, ou *local binary pattern (LBP)* en anglais, permet également de détecter des textures ainsi que des objets dans une image.

Cette méthode a été popularisée en 1996 ([19]) pour l'analyse des différentes textures d'une image après avoir été introduit pour la première fois en 1993 ([8]) pour mesurer le contraste d'une image.

Le principe de cette méthode est de comparer le niveaux de luminance d'un pixel d'une image avec celui de ses voisins. Cette méthode va également résulter en la création d'une nouvelle image de même taille que l'originale contenant le résultat du motif binaire local de chaque pixel.

Ces motifs se calculent de cette façon :

- Calculer la différence entre le pixel étudié et chacun de ses voisins pour obtenir une matrice
- Changer les valeurs de la matrice : les négatives deviennent 0 et les positives ou égales à 0 deviennent 1
- Multiplier, terme à terme, avec une matrice contenant les puissances de 2 de 1 (2^0) à 128 (2^7).
- Additionner le résultat de chaque multiplication pour obtenir le LBP

Ce fonctionnement est résumé dans la Figure 10 (image issue de la page Wikipédia des LBP).

Dans cet exemple, les pixels voisins utilisés pour calculer le LBP font partie du voisinage direct du pixel étudié mais il est également possible d'utiliser des pixels faisant partie d'un voisinage plus éloigné. Cela peut permettre de détecter des bords ou des coins dans une image (ex : coins des yeux).

3 Classification

Une fois les features extraites, il ne reste plus qu'à les analyser pour les classer.

Deux types de classificateurs existent : les systèmes basés sur des règles d'experts (*rule-based expert systems*) et les classificateurs avec apprentissage (*machine learning classifiers*). Les premiers sont de moins en moins répandus car ils sont beaucoup plus compliqués à utiliser que les autres. C'est donc sur les classificateurs avec apprentissage que je vais me concentrer.

Parmi ces classificateurs, nous pouvons les classer en 2 catégories : les binaires et les multi-classes.

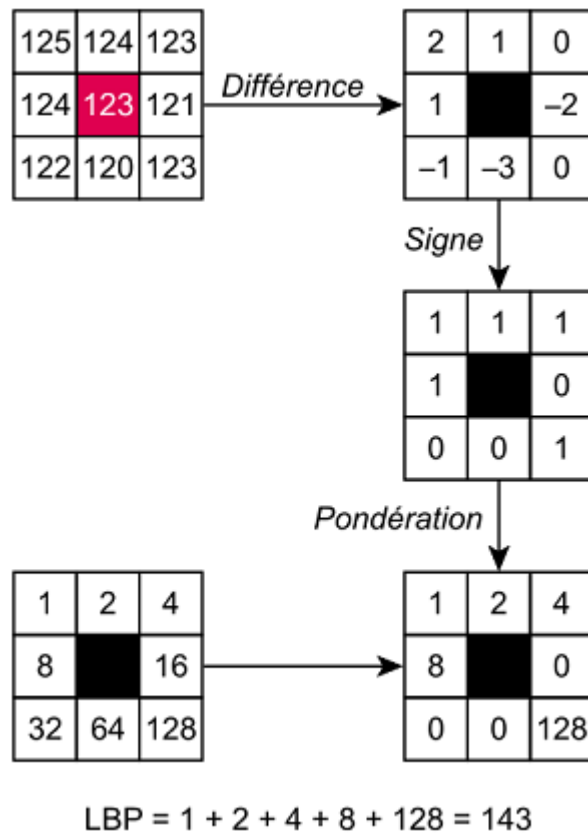


Figure 10 – Fonctionnement des motifs binaires locaux

3.1 Classificateurs binaires

Comme l'indique son nom, ce type de classificateur ne permet de classer une instance qu'entre 2 classes (ex : sourire VS pas de sourire).

Le classificateur binaire le plus connu est le **Support Vector Machine (SVM)**.

Les SVMs se basent sur 2 notions très importantes.

La première est la notion de *marge maximale*. Cette marge est la distance entre la frontière de séparation des 2 classes et les échantillons les plus proches appelés vecteurs supports. La frontière de séparation est choisie de telle sorte qu'elle maximise le plus la marge. C'est donc grâce à un apprentissage que l'on peut définir cette frontière.

La deuxième notion est celle de la transformation de l'espace de représentation en un espace de dimension plus grande, voir infinie. Cette notion est utile lorsque les données ne sont pas linéairement séparables dans l'espace d'origine. En projetant dans un espace de dimension plus grande, il est plus probable de trouver une séparation, comme nous le montre l'exemple en Figure 11.

Cette transformation est réalisée via une fonction noyau qui va permettre de transformer un produit scalaire dans un espace de grande dimension.

3.2 Classificateurs multi-classes

Contrairement aux binaires, ces classificateurs permettent de classer une instance selon un nombre de classes supérieur à 2.

Le classificateur multi-classes le plus connu est le **k-Plus Proche Voisin (kPPV)**.

Le kPPV fonctionne d'une manière très simple.

Tout d'abord un apprentissage grâce à une base d'apprentissage composée de couples "entrée-sortie".

Puis la phase de classification se fait en cherchant la distance minimum entre un nouvel échantillon

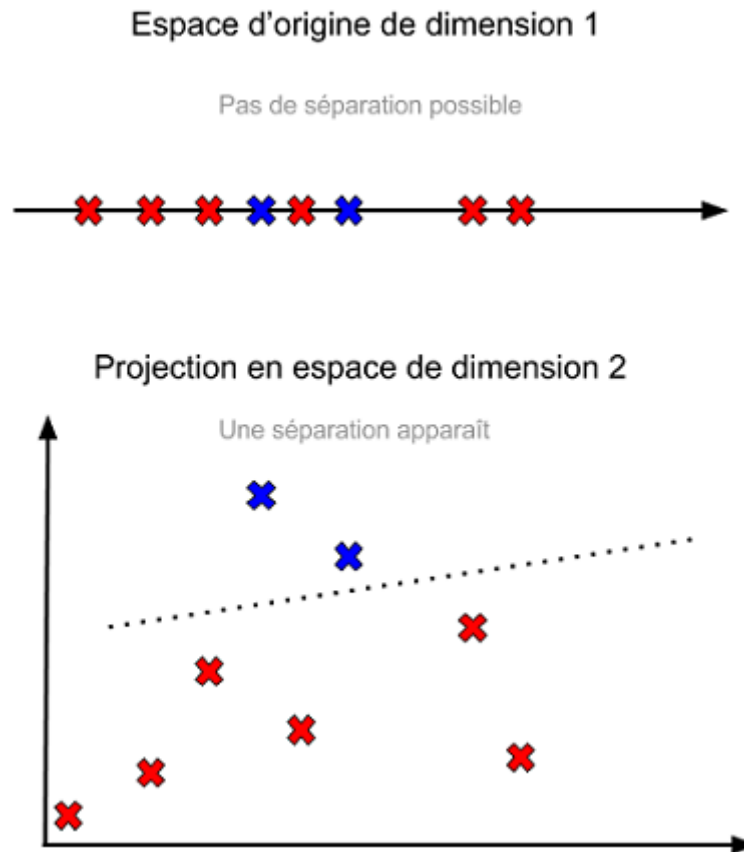


Figure 11 – Exemple de projection dans un espace de plus grande dimension réalisé par SVM

d'entrée dont la sortie doit être déterminée et les k échantillons d'apprentissage dont l'entrée est la plus proche.

Une fois ces k plus proche voisins trouvés, il suffit de trouver quelle classe est majoritairement présente pour trouver la sortie associée et donc à quelle classe appartient le nouvel échantillon. Si jamais il y a un nombre égale d'échantillon de plusieurs classes (par exemple avec $k=3$ on obtient 1 échantillon de 3 différentes classes), on choisi alors aléatoirement la classe d'appartenance.

Un exemple vous est présenté en [Figure 12](#) dans lequel 3 classes ont été appris via la phase d'apprentissage et où k est fixé à 3 ; quatre échantillons doivent être classés dont un se trouvant en zone d'indécision.

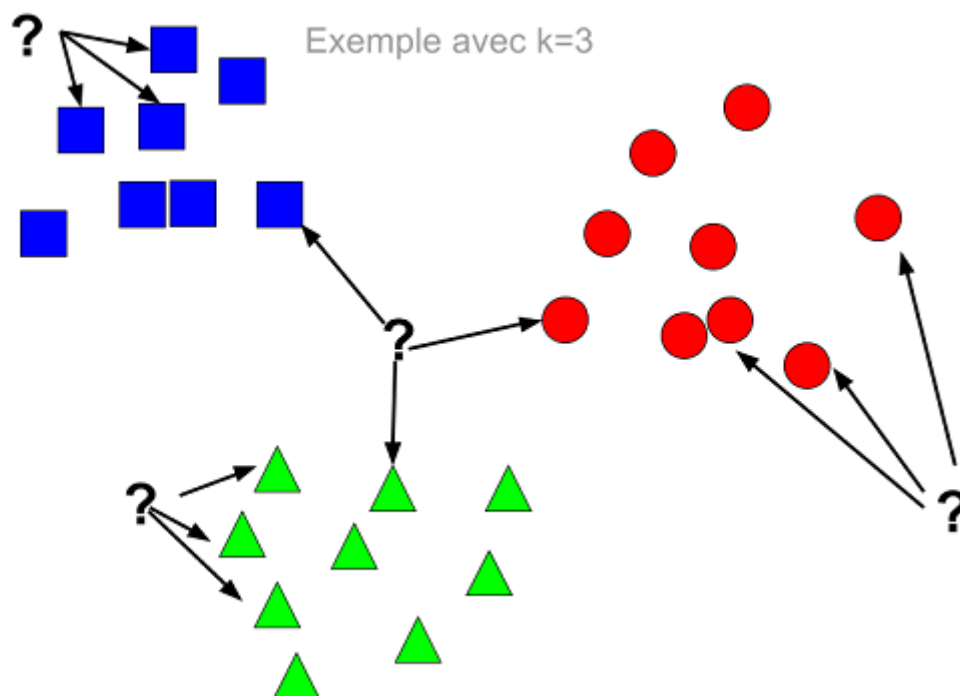


Figure 12 – Fonctionnement du kPPV avec $k=3$, 3 classes et 4 échantillons dont un indéci

5

Nouvelle approche

Toutes les recherches que je vous ai présenté pour l'instant existent et son implanter dans la communauté scientifique depuis plusieurs années, elles ne présentent donc pas vraiment d'innovations possibles.

Sur la demande de Mr Conte, j'ai dû étudier un article écrit par Jonathan Vitale, Mary-Anne Williams, Benjamin Johnston et Giuseppe Boccignone, ce dernier étant un collègue de Mr Conte.

Affective facial expression processing via simulation : A probabilistic model [29] décrit une toute nouvelle approche permettant de reconnaître des expressions faciales **via un modèle probabiliste**.

Pour aller de paire avec cet article, les auteurs ont également crée un programme Matlab mettant en pratique leurs idées.

Le but de ce projet étant également de créer une application innovante en travaillant avec des personnes handicapées, nous nous sommes donc mis d'accord avec mes tuteurs académiques pour utiliser cet article et le programme associé pour développer notre application. J'aurais donc pour mission dans la phase de développement d'améliorer le programme existant et d'y ajouter des fonctionnalités qui seront propres à l'utilisation que nous ferions de l'application.

Un entretien Skype est prévu avec Mr Boccignone pour parler du programme et avoir certaines précisions à propos des notions abordées dans l'article. L'écriture d'un article en collaboration avec les auteurs originaux est également envisagée.

Je vais maintenant vous présenter ce modèle probabiliste permettant de représenter les émotions ainsi que le travail déjà réalisé via le programme Matlab.

1 Contexte de simulation

Pour comprendre les études menées dans l'article, il faut d'abord poser le problème et donc le contexte de simulation de la détection d'émotions.

A un certain moment, un individu, appelé *acteur*, éprouvera un certain état interne, noté X_{act} . Cet état interne X_{act} peut soit être déclenché par un événement extérieur soit induit (remémoration d'un certain souvenir).

X_{act} va déclencher un comportement correspondant, noté Y_{act} , qui peut être par exemple une expression facial, une posture particulière, un changement du rythme cardiaque, etc.

Un second individu, appelé *observateur*, va maintenant utiliser Y_{act} pour trouver l'état interne X_{obs} lui permettant d'être dans l'état mental M_{obs} correspondant à celui de l'acteur.

Cette situation de simulation est illustrée en **Figure 1** (figure tirée de l'article [29]).

Plusieurs manières existent pour traduire cette simulation en langage compréhensible par un ordinateur, dont 3 sont intéressantes.

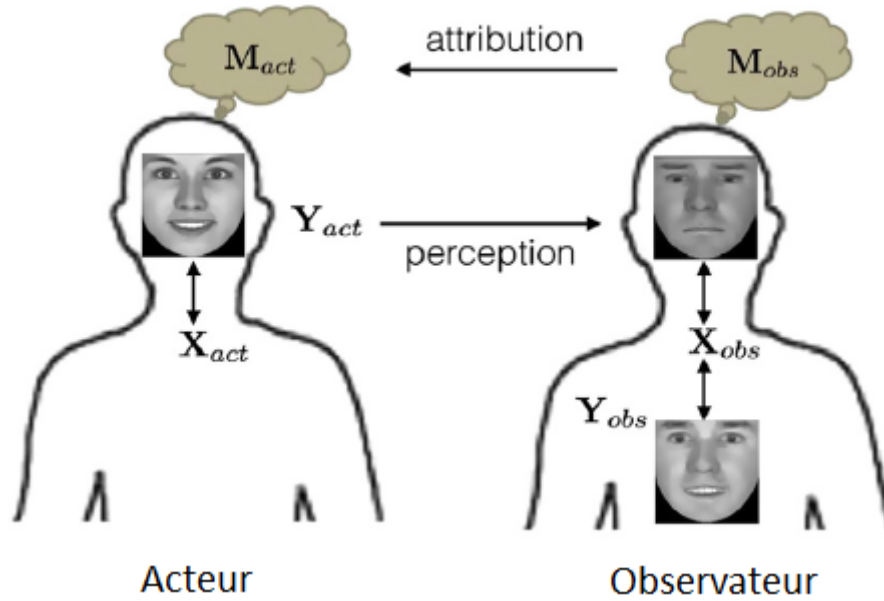


Figure 1 – Situation de simulation pour la détection d'émotions

Générer-et-tester

L'observateur suppose que l'acteur ressent X_{act} , il va donc faire en sorte de ressentir la même chose X_{obs} . En fonction de X_{obs} , l'observateur obtient Y_{obs} . Il compare ensuite son état extérieur Y_{obs} à celui de l'acteur, Y_{act} , et si se sont les mêmes, alors ce que ressent l'observateur avec cette émotion, M_{obs} , est également ce que ressent l'acteur, M_{act} .

En langage mathématique, cela peut se traduire de la façon suivante :

$$\text{Soit l'hypothèse } X_{act} \simeq X_{obs} : \text{SI } X_{obs} \mapsto Y_{obs} \simeq Y_{act} \text{ ALORS } X_{obs} \mapsto M_{obs} \simeq M_{act}$$

Simulation inverse

L'observateur imite l'état extérieur Y_{act} de l'acteur pour obtenir Y_{obs} . Puis, en fonction de l'état copiée Y_{obs} , l'observateur va ressentir X_{obs} . En supposant $X_{obs} \simeq X_{act}$, l'observateur obtient un état mental M_{obs} censé être le même que celui de l'acteur, M_{act} .

En langage mathématique, cela peut se traduire de la façon suivante :

$$\text{Soit } Y_{obs} \simeq Y_{act} : Y_{obs} \mapsto X_{obs} \mapsto M_{obs} \simeq M_{act}$$

Simulation inverse utilisant une boucle "as if"

Similaire au modèle précédent sauf qu'ici l'observateur ne copie pas l'état extérieur. Il se contente de se demander (intervention du "as if") ce que ressent l'acteur en fonction de Y_{act} . Il va donc ressentir X_{obs} qui va le mener à M_{obs} et donc M_{act} .

En langage mathématique, cela peut se traduire de la façon suivante :

$$Y_{act} \mapsto X_{obs} \mapsto M_{obs} \simeq M_{act}$$

2 Modèle réalisé

Pour la réalisation du modèle probabiliste, plusieurs probabilités ont été définies :

- $P(Y_{obs}/Y_{act})$: représente la probabilité pour l'observateur de ressentir intérieurement Y_{obs} quand l'acteur affiche Y_{act} . Cette probabilité est appelé *transcodage*.
- $P(X_{obs}/Y_{obs})$: représente la probabilité d'être dans un état interne X_{obs} sachant Y_{obs} . Cette probabilité est appelé *correspondance inverse*.

- $P(Y_{obs}/X_{obs})$: représente la probabilité que l'observateur génère un état Y_{obs} sachant l'état interne X_{obs} . Cette probabilité est appelé *correspondance vers l'avant*.

Une fonction de décision $\mathcal{D}(\bullet)$ est également nécessaire et permettra de comparer l'état interne Y_{act} de l'acteur avec un état simulé $\check{Y}_{obs}$.

Ces probabilités nous permette d'obtenir les variables suivantes :

$$y_{obs} \sim P(Y_{obs}/Y_{act} = y_{act}) \quad (1)$$

qui permet de définir le processus de transcodage transformant une instance de l'expression faciale de l'acteur y_{act} en une instance de représentation interne de l'observateur y_{obs} ,

$$x_{obs} \sim P(P(X_{obs}/Y_{obs} = y_{obs}) \quad (2)$$

qui décrit le processus de correspondance inverse donc la détection de l'état x_{obs} à partir de l'expression faciale (interne) y_{obs} , et

$$\check{y}_{obs} \sim P(Y_{obs}/X_{obs} = x_{obs}) \quad (3)$$

qui décrit le processus de correspondance vers l'avant et donc la correspondance de l'expression simulé par l'observateur $\check{y}_{obs}$ quand il se trouve dans l'état interne x_{obs} .

La fonction $\mathcal{D}(\bullet)$ compare le y_{obs} transcodé de l'équation (1) à l'expression générée intérieurement $\check{y}_{obs}$ de l'équation (3), et est également utilisé via l'équation (2) pour contrôler quand le processus de comparaison aura converger vers la solution la plus probable.

Les processus de transcodage, de correspondance inverse et de correspondance vers l'avant se produisent dans 2 espaces latents distincts :

- l'espace latent d'auto-projection, noté Z
- l'espace latent du premier ordre *phenomenologique*

L'espace Z est défini de la façon suivante :

$$P(Z) = \mathcal{N}(0, I_L) \quad (4)$$

qui permet de donner la priorité des points dans l'espace Z de dimension L,

$$P(y_{act}/z) = \mathcal{N}(W_{act}z + \mu_{act}, \sigma_{act}^2 I_D) \quad (5)$$

qui permet d'avoir la probabilité d'obtenir l'expression faciale de l'acteur y_{act} , avec $D \gg L$, sachant $z \in Z$, et

$$P(y_{obs}/z) = \mathcal{N}(W_{obs}z + \mu_{act}, \sigma_{obs}^2 I_D) \quad (6)$$

qui permet d'avoir la probabilité d'obtenir l'expression faciale de l'observateur y_{act} , avec $D \gg L$, sachant $z \in Z$, avec :

$\mathcal{N}(\bullet)$: distribution Gaussienne ; W_{act} et W_{obs} : paramètres de correspondance de l'acteur et de l'observateur ; μ : moyenne ; σ : variance ; I_L et I_D : matrices identités de dimensions L et D.

On peut également définir les variables suivantes :

$$W = \begin{pmatrix} W_{act} \\ W_{obs} \end{pmatrix}, \mu = \begin{pmatrix} \mu_{act} \\ \mu_{obs} \end{pmatrix} \text{ et } \Phi = \begin{pmatrix} \sigma_{act}^2 I_D & 0 \\ 0 & \sigma_{obs}^2 I_D \end{pmatrix}.$$

Vu que le modèle est gaussien, y_{act} et y_{obs} suivent une distribution gaussienne de la forme $\mathcal{N}(\mu, \Sigma)$ avec $\Sigma = \Phi + WW^T$ correspondant à la matrice de covariance.

On obtient donc :

$$P(y_{obs}|y_{act}) \sim \mathcal{N}(\hat{\mu}_{obs}, \hat{\Sigma}_{obs}) \quad (7)$$

où $\hat{\mu}_{obs} = \mu_{obs} + \Sigma_c^T \Sigma_a^{-1}$ et $\hat{\Sigma}_{obs} = \Sigma_b - \Sigma_c^T \Sigma_a^{-1} \Sigma_c$ est le complément de Schur de Σ réécrit sous la forme du

$$\text{bloc } \Sigma = \begin{pmatrix} \Sigma_a & \Sigma_c \\ \Sigma_c^T & \Sigma_b \end{pmatrix}.$$

L'équation (7) nous retourne la probabilité de correspondance de l'expression faciale de l'observateur

sachant une expression faciale similaire de l'acteur. Pour faire cela, elle utilise W_{act} et W_{obs} en distribution normale multivariée.

Pour résoudre $\check{y}_{obs}$, on passe par l'utilisation du GPLVM (Gaussian Process Latent Variable Model) et l'on obtient

$$P(Y_{obs}|X_{obs}) = \frac{1}{\sqrt{(2\pi)^{ND}|K^D|}} \exp\left(-\frac{1}{2}tr(K^{-1}Y_{obs}Y_{obs}^T)\right) \quad (8)$$

avec Y_{obs} correspondant à l'ensemble d'apprentissage et K à la matrice de noyau dont les éléments sont définis par $(K)_{i,j} = \Phi(x_{i,obs}, x_{j,obs})$.

Une fois que $P(Y_{obs}/X_{obs})$ est appris, il est très facile d'obtenir $P(X_{obs}/Y_{obs})$ par GPLVM.

3 Fonctionnement

Le fonctionnement de ce modèle probabiliste, utilisant les notions présentées dans la section précédente, est décrit dans la Figure 2.

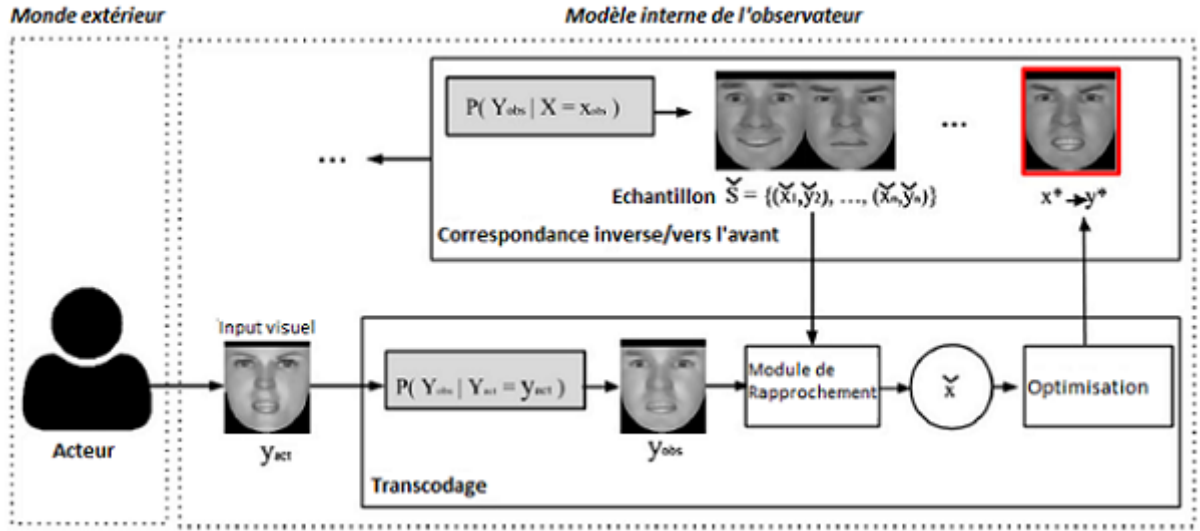


Figure 2 – Fonctionnement du modèle probabiliste

y_{act} est transcodé via l'espace latent d'auto-projection Z en y_{obs} .

Au même moment, un ensemble $\check{S}$ d'échantillons d'expressions de l'observateur est généré via l'équation (8). Avec $\mathcal{D}$, une mesure de similarité est utilisée pour évaluer la vraisemblance entre les échantillons de $\check{S}$ et de y_{obs} . L'état initial $\check{x}$ est sélectionné de cette manière :

$$\check{x} \in \check{S} | \check{x} \mapsto \check{y}_{obs} \wedge \check{y}_{obs} = \arg_y \max \mathcal{D}(\check{y}_{obs}, y)$$

En utilisant HGP-LVM (Hierarchical Gaussian Process Latent Variable Model), il est possible de créer un vocabulaire $S : X \mapsto \{Y_{obs}^j\}_{j=1}^{\infty}$ qui génère un ensemble infini d'expressions faciales.

$\check{S}$ est un sous-ensemble de S .

Chaque point de l'espace latent représente un état interne phénoménologique x . Une telle représentation est de dimension q et dans le cas de cet article et du programme Matlab, $q=2$.

4 Comparaison

Pour fonctionner, le module de rapprochement (cf Figure 2) utilise la mesure **SSIM** (Structural SIMilarity). Cette mesure est effectuée sur plusieurs "parties" de l'image, appelées fenêtres, et ensuite une moyenne

est calculée pour obtenir le résultat final. Le mesure entre 2 fenêtres x et y se calcule comme suit :

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

avec μ_x et μ_y : moyennes de x et y ; σ_x^2 et σ_y^2 : variances de x et y ; σ_{xy} : covariance de x et y ; c_1 et c_2 : variables de stabilisation de la division quand celle-ci à un dénominateur faible.

5 Programme Matlab

Le programme associé à cette article permet de mettre en application le modèle proposé.

Au lancement, la phase d'apprentissage s'enclenche. Elle consiste en l'apprentissage de 60 images en niveaux de gris d'une personne réalisant plusieurs expressions faciales différentes. Ces images sont tout d'abord récupérées depuis un fichier *.mat* où elles sont écrit sous forme d'une ligne composé de 551 pixels, autrement dit une matrice de pixels de taille 1*551.

Ces images seront ensuite redimensionnées à la taille 29*19 pixels pour obtenir de "vraies" images.

Ces 60 images sous forme de lignes sont stockées dans une variable Y qui sera donc de taille 60*551.

Le méthode PCA (Principal Component Analysis) est ensuite appliquée à Y. Cela va permettre de décorrélérer les variables entre elles dans le but d'obtenir de nouvelles variables appelées "composantes principales". Cela va réduire le nombre de variables et réduire la redondance de l'information.

Le GPLVM est ensuite lancé.

Les coordonnées x et y de chaque image dans l'espace latent sont calculées au travers de plusieurs itérations (ici 15 itérations) pour obtenir des coordonnées précises grâce à la mise à jour à chaque itération.

Une fois que les coordonnées des 60 images sont calculées, le programme fini par afficher l'espace latent. Sur cette espace se trouvent les positions précédemment calculées, comme montré sur la **Figure 3**.

En parallèle de l'affichage de cette espace s'affiche également une fenêtre contenant une image.

La principale fonctionnalité de ce programme est de pouvoir se "déplacer" dans l'espace latent avec la souris et de visualiser en temps réel à quelle image, donc émotion, correspond la position de la souris sur cet espace. C'est donc dans cette deuxième fenêtre contenant une image que va s'effectuer la mise à jour en temps réel.

Pour réaliser cela, le programme a défini un déclencheur sur l'évènement *move* de la souris lorsque cette dernière se déplace. Une fois déclenché, le programme va récupérer les positions de la souris et des axes dans la fenêtre pour ensuite normaliser la position du point par rapport à celles des axes, ce qui va permettre d'obtenir la position x et y de la souris dans le repère.

En fonction de cela, l'image apprise lors de la phase d'apprentissage la plus proche et dont l'expression est la plus probable est retournée.

Pour finir, l'image est ensuite affichée dans la fenêtre. Des exemples des images pouvant être retournées sont présentées en **Figure 4**.

6 Modification à apporter

Pour adapter ce programme à notre problématique, il va falloir le modifier pour qu'il puisse classer une image d'un visage réalisant une expression, prise en entrée, et retourner l'émotion correspondante. Pour cela, une première piste serait de faire passer l'image en entrée dans les mêmes fonctions que les images du set d'apprentissage ce qui nous permettrait d'obtenir ses coordonnées dans l'espace latent. Puis avec ces coordonnées, procéder de la même façon que lorsque l'on se déplace sur l'espace latent avec la souris et donc déduire de quelle émotion ces coordonnées sont la plus proche.

Pour l'instant, cette idée n'est qu'une hypothèse et aucun travail n'a été effectué dans le but de corroborer ou non cette première piste. Ce sera le travail que j'aurai à effectuer durant la phase Développement de ce projet.

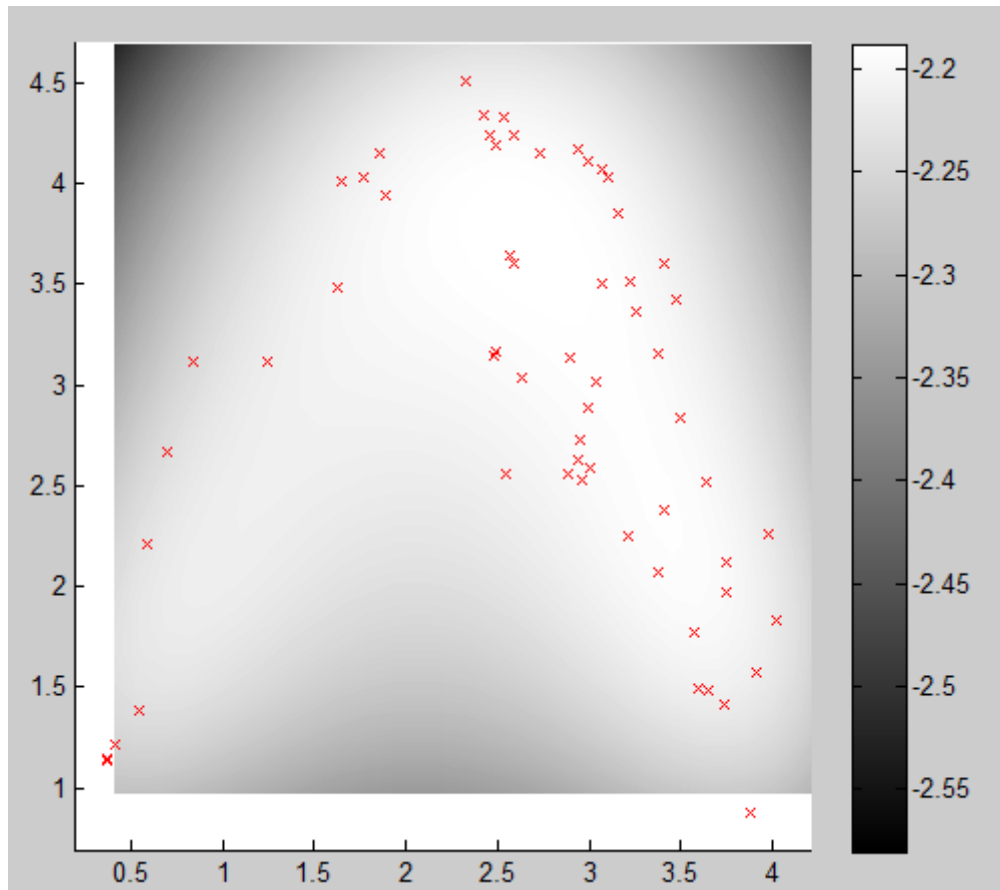


Figure 3 – Espace latent représenté grâce au programme Matlab



Figure 4 – Exemples d'images retournées lorsque l'on se déplace sur l'espace latent

6

Spécifications de l'application

1 Utilisateurs

Il n'y aura qu'un seul type d'utilisateur qui manipulera (indirectement) cette application : **les personnes handicapées**.

Elles vagueront à leurs occupations normalement et l'application se chargera de repérer leurs émotions. Ces personnes n'ont donc aucunement besoin d'avoir des connaissances en informatique.

2 Fonctions

A l'heure où j'écris ce rapport, toutes les spécifications concernant l'ensemble des fonctions ne sont pas encore connues.

Je vais néanmoins présenter les spécifications qui ont déjà été établi pour certaines fonctions.

Fonction F4 : Reconnaître l'expression/Apprentissage

Cette fonction sera la même que celle déjà utilisée par Vitale et al. dans leur programme Matlab.

Fonction F5 : Reconnaître l'expression/Extraction des *features* du visage

Étant donné que nous utiliserons le modèle de Vitale et al., cette fonction n'est pas nécessaire puisqu'aucune extraction de features n'est requise.

Fonction F6 : Reconnaître l'expression/Classification

Cette fonction devra utiliser le programme Matlab de Vitale et al. déjà existant. A partir des travaux déjà réalisés, cette fonction viendra se greffer sur le programme et permettra, grâce à une image donnée en entrée, de trouver sur l'espace latent, recensant les échantillons appris lors de la phase d'apprentissage, l'émotion la plus proche et donc la plus probable.

Pour l'instant, une fonctionnalité du programme est de pouvoir naviguer dans cette espace avec la souris de l'ordinateur et de voir en temps réel l'émotion associée au point désigné par la souris. Il suffira donc de rajouter une méthode permettant de placer un point fictif sur cet espace (correspondant à l'échantillon à déterminer) et d'utiliser la fonction précédemment utilisée dans la navigation pour obtenir l'émotion associée la plus probable.

7

Planning et outils utilisés

1 Planning du projet

Pour la réalisation de ce projet, j'ai divisé mon travail à réaliser en plusieurs tâches, chaque tâche ayant une date limite pour que le projet soit fini en temps et en heure. Je vais donc maintenant vous présenter ce planning avec ces tâches sous forme de tableaux car plus simple à lire qu'une petite image avec un diagramme de Gantt.

Table 1 – Planning du projet

Partie Recherche (réel)				
Nom	Date début	Durée	Date Fin	Livrable
Comparaison 2D VS 3D	16/09/15	1 semaine	24/09/15	Compte-rendu
Architecture	23/09/15	1 semaine	01/10/15	Powerpoint
Étude techniques extraction de <i>features</i>	30/09/15	6 semaines	12/11/15	Powerpoint
Étude bases de données	28/10/15	1 semaine	05/11/15	Powerpoint
Étude article/programme Vitale et al.	19/11/15	7 semaines	07/01/16	
Rédaction rapport de recherche	08/10/15	13 semaines	05/01/16	Rapport L ^A T _E X
Préparation soutenance	05/01/16	1 semaine	14/01/16	Powerpoint
Partie Développement (prévision)				
Nom	Date début	Durée	Date Fin	Livrable
Identification des modifications	14/01/16	2 semaines	02/02/16	
Réalisation des modifications	03/02/16	4 semaines	03/03/16	
Rédaction rapport développement	27/01/16	9 semaines	24/03/16	Rapport L ^A T _E X
Préparation soutenance	24/03/16	1 semaine	01/04/16	Powerpoint

La tâche "Comparaison 2D VS 3D" correspond aux recherches que j'ai effectués pour faire un début d'état de l'art sur les technologies 3D et 2D permettant de reconnaître des émotions. Cette recherche a permis de choisir que notre application se ferait avec des données 2D après le compte-rendu que j'ai réalisé auprès de mes tuteurs académiques.

La tâche "Architecture" correspond aux recherches m'ayant permis de définir les différentes étapes nécessaires pour construire un système de reconnaissance d'émotions : détection du visage, extraction des *features* et classification.

La tâche "Étude techniques extraction de *features*" correspond aux recherches que j'ai menées sur les

filtres de Gabor et sur les filtres de Haar ainsi que d'autres techniques permettant d'extraire les features d'un visage.

La tâche "Étude bases de données" correspond à l'étude des bases de données Cohn-Kanade, MMI, HCI-Tagging et MHI-Mimicry ,ainsi qu'à la prise de connaissance des données contenues dans les bases Cohn-Kanade et HCI-Tagging.

Ces 3 dernières tâches ont amenées la création d'une présentation Powerpoint que j'ai présenté à mes tuteurs académiques.

La tâche "Étude article/programme Vitale et al." correspond à la compréhension de l'article écrit par Vitale et al. et dont je vais me baser dessus pour la réalisation de l'application. Cela correspond également au temps passé à essayer de faire marcher le programme sous Octave (un Matlab-like gratuit) ainsi qu'à l'étude du code pour arriver à recouper l'article avec ce dernier.

La tâche "Identification des modifications" correspond au travail que je vais réaliser à la suite de l'étude de l'article de Vitale et al. et qui va permettre d'identifier quelles parties du programme peuvent être améliorées et les fonctions à rajouter pour pouvoir reconnaître l'émotion contenue dans une image passée en entrée du programme.

La tâche "Réalisation des modifications" consiste à réaliser les modifications qui auront été identifiées dans la tâche précédente.

2 Outils

2.1 Versioning

Pour pouvoir gérer les différentes versions de mon application, j'utilise **Git** ([consulter mon profil GitHub](#)).

Pour l'instant il ne me sert qu'à versionner mon rapport de projet mais dès que je commencerai le développement, il me permettra également de versionner mon application.

2.2 Gestion de projet

Pour gérer les différentes tâches que j'ai à faire dans ce projet, j'utilise **Trello** (<https://trello.com/>) qui permet de créer des tâches, de les classer sous forme de listes de tâches, d'affecter des statuts (A faire, En cours, Fait). Il est possible d'affecter des personnes à chacune des tâches, de mettre des commentaires et de partager des fichiers.

Ce Trello n'est que pour mon utilisation personnelle et n'est pas partagé avec mes tuteurs académiques.

Deuxième partie

Développement

8

Identifications des modifications

Pour réaliser notre application de reconnaissance d'émotions, mes tuteurs pédagogiques et moi-même avons donc choisi de reprendre le programme réalisé par Vitale et al. présenté précédemment. Grâce à un entretien Skype réalisé avec Giuseppe Boccignone, un des auteurs de l'article [29], nous avons pu identifier les modifications et obtenir certaines pistes concernant leurs implémentations.

Ces modifications sont au nombre de 4 :

- utiliser des images de la base de données HCI Tagging pour créer la base d'apprentissage ainsi que la base de test
- changer la représentation des données, passer d'une représentation pixellique à une représentation vectorielle
- utiliser l'algorithme FGPLVM (Faster GPLVM) pour remplacer l'algorithme GPLVM et donc obtenir des résultats plus rapides
- comparer les trajectoires des différentes émotions dans l'espace latent pour trouver la plus probable

Une migration vers le langage Python a également été proposée mais par manque de temps et de compétences dans ce langage, j'ai préféré continuer le projet en Matlab.

1 Changement de la base d'apprentissage

Dans le programme d'origine, la base d'apprentissage permettant de construire l'espace latent se composait de 60 images de taille 29*19 pixels.

Du fait de la résolution de ces images, elles étaient difficilement exploitables pour mon projet.

Avec mes encadrants, nous avons décidé de nous servir de la base HCI-Tagging, présentée précédemment (cf [subsubsection 2.2.5](#) (Chapitre 3)), pour récupérer de nouvelles images. Ces images auront donc une résolution plus importante puisqu'elles seront de taille 780*580 pixels.

Il a également été décidé d'extraire plusieurs images consécutives pour chaque émotion plutôt qu'une seule, ceci dans le but de pouvoir représenter l'évolution d'une émotion au cours du temps grâce à la représentation d'une trajectoire dans l'espace latent.

2 Changement de la représentation des données

Une fois la nouvelle base d'apprentissage définie, je me suis intéressé à la représentation des données et donc de l'information portée par les images de la base.

Dans le programme d'origine, chaque pixel de chaque image était analysé et via l'algorithme de **Principal Component Analysis** (PCA, [9]), les pixels représentant un réel apport d'information étaient retournés.

C'était ensuite ces pixels qui étaient analysés pour pouvoir construire l'espace latent.

Cependant l'algorithme du PCA est très chronophage et c'est donc pour cela que les images de la base d'apprentissage d'origine étaient de petite taille (29*19 pixels). Il était bien entendu impossible d'utiliser cette technique avec des images de taille 780*580 pixels comme les nôtres car cela représenterait 452 400 pixels à faire analyser par le PCA.

Il devenait alors évident d'abandonner cette représentation pixelique pour une autre plus adaptée.

Après concertation avec mes encadrants, nous avons décidé d'utiliser la position des *features* des visages et de faire analyser le vecteur des positions de chaque point à notre programme.

3 Utilisation de l'algorithme F-GPLVM

Le programme d'origine utilisait l'algorithme GPLVM développé par Neil Lawrence [11] [WWW5] en 2004, et il a donc été cité la possibilité de passer à sa version supérieur, c'est-à-dire le FGPLVM [WWW6], également développé par Lawrence en 2007.

Cela vous me permettre, comme son nom l'indique, d'avoir des résultats plus rapidement mais également de pouvoir rajouter des contraintes ainsi que du dynamisme sur mes variables latentes pour pouvoir obtenir des trajectoires plus lisses dans l'espace latent.

4 Représentation des données sous formes de trajectoires

Comme introduit dans les sections précédentes, nous allons utiliser des trajectoires pour représenter les émotions dans l'espace latent plutôt que de simples points.

Cela est dû à une raison principalement : cela nous permet d'avoir l'évolution de l'émotion au cours du temps ce qui est beaucoup plus fidèle à la réalité.

En effet, personne ne passe directement d'un état neutre à un grand sourire, il y a une progression dans son expression et c'est donc cette progression que permet de définir les trajectoires.

9

Rédaction de l'article scientifique pour la conférence "Handicap 2016" de l'IFRATH

Après avoir identifié les modifications nécessaires à apporter au programme, j'ai écrit en collaboration avec Donatello Conte, Mohamed Slimane et Giuseppe Boccignone l'article *Une nouvelle approche à la détection d'émotions pour l'aide à l'interaction homme-machine chez les personnes handicapées*.

Cet article a été soumis à validation pour la conférence "Handicap 2016" de l'IFRATH le 10 février et nous obtiendrons la réponse, si oui ou non il sera publié, le 8 avril.

Vous pouvez retrouver l'article en annexe de ce rapport ([Annexe A](#)).

10

Réalisations des modifications

1 Création de la nouvelle base d'apprentissage

Pour créer la base d'apprentissage, j'ai donc utilisé des images issues des vidéos se trouvant dans la base HCI-Tagging [24][12]. Pour chaque émotion, j'ai extrait 15 frames consécutives (1 frame extraite toutes les 50 frames) de la vidéo ce qui va nous permettre de représenter l'évolution de l'émotion au cours du temps. Les vidéos utilisées ainsi que l'émotion associée se trouve dans le [Table 1](#).

Table 1 – Données utilisées dans notre base d'apprentissage

	N° sujet	N° session	Émotion
1	1	6	joie
2	3	264	dégoût
3	1	28	tristesse
4	6	662	colère
5	3	290	surprise

Ces images sont de taille 780*580 pixels.

J'ai décidé de ne pas utiliser l'émotion de la peur car je n'ai trouvé aucune données réellement exploitables dans la base HCI-Tagging pour cette émotion. En effet, très peu de vidéos représentant la peur se trouve dans cette base et les réactions des sujets soumis à cette émotion sont imperceptibles par rapport à leur état neutre. Donc pour ne pas "fausser" mon application, j'ai décidé de ne pas l'utiliser.

Les frames que j'ai utilisé pour représenter l'émotion de la joie se trouve en [Figure 1](#).

2 Représenter les données via la position des *features*

Vu que je vais utiliser la position des *features* des visages, j'ai tout d'abord dû rechercher un programme permettant de trouver leurs positions en Matlab. Après en avoir testé plusieurs, j'ai choisi le programme Matlab réalisé par Akshay Asthana dans le cadre de son article [1]. Ce programme permet d'extraire les coordonnées des positions de 66 features d'un visage. Il permet aussi d'extraire des données sur l'orientation du visage, même si cela ne nous sera pas utile pour ce projet.

Les 66 points du visage trouvés via ce programme se trouve en [Figure 2](#).

Ce programme est très polyvalent car il nous permet, si nous le souhaitons, de changer l'algorithme de détection de visage ainsi que des features du visage. Pour l'instant, 2 méthodes sont présentes permettant

de récupérer ces points, une méthode basé sur des arbres (qui est très lente) et une méthode utilisant une fonctionnalité innée à Matlab (beaucoup plus rapide). Cette fonctionnalité permet, directement via Matlab sans utiliser de plugins supplémentaires, de détecter les visages dans une image via l'algorithme de Viola-Jones dont j'ai déjà parlé précédemment (cf [subsection 1.1](#) (Chapitre 4)) et ensuite de détecter les différentes parties d'un visage (yeux, bouche, nez, etc). C'est cette méthode que j'ai choisi d'utiliser pour extraire les coordonnées des positions des 66 features.

En plus de ces coordonnées, nous avons également discuté de la possibilité d'utiliser la distance entre ces points plutôt que leurs coordonnées pour représenter ces données. Plus clairement, calculer la distance entre un point et tous les autres pour ensuite récupérer les K distances les plus petites, qui vont donc correspondre en toute logique aux K voisins les plus proches du point choisi au départ.

Cette représentation permettrait de représenter le déplacement des points d'un visage par rapport à ses voisins et non pas seulement leur déplacement dans l'espace.

Après plusieurs tests de construction de l'espace avec ce type de données en changeant la valeur de K, nous sommes arrivés à la conclusion que K=10 était la meilleure solution pour la représentation des données sur l'espace latent.

Le code Matlab réalisant ces opérations est le suivant :

```

1 \label{codeDetection}
2 bbox_method = 1; % Permet de choisir entre les différents modes de détection
3 % 0: Méthode basée sur les arbres, 1: Détecteur de visage de Matlab (ou détecteur externe), 2: Utilisation de boîtes prédéfinies
4 visualize=0; % 1 : visualiser l'image avec les 66 points placés, 0 : ne pas visualiser l'image
5
6 %-----%
7 % Chargement des images au format numeroEmotion.numeroFrame.png
8 index=0; % Indice qui va permettre d'écrire les données dans data
9 for j=1:5 % Pour chaque émotion ...
10     for i=1:15 % Pour chaque frame ...
11         data(index+i).name = strcat(num2str(j),'. ',num2str(i),'.png'); % Récupération du nom
12         data(index+i).img = im2double(imread(strcat(num2str(j),'. ',num2str(i),'.png'))); % Récupération de l'image
13         data(index+i).points = []; % Va contenir les coordonnées des 66 points
14         data(index+i).pose = []; % Va contenir l'orientation du visage [Pitch;Yaw;Roll]
15     end
16     index=index+15; %Fin des 15 frames d'une émotion donc incrémentation de 15 de l'indice pour ne pas réécrire sur les données précédentes
17 end
18 %-----%
19
20 load('model/DRMF_Model.mat'); % Chargement du modèle permettant la détection
21 data=DRMF(c1m_model,data,bbox_method,visualize); % Lancement de la détection des 66 points
22
23 %-----%
24 for i=1:size(data,2) % Pour chaque image, toute émotion confondue ...
25     dist = calcKDistance(10,data(i).points); % Calcul des distances des 10 points les plus proches
26     datas(i)= struct('points',data(i).points,'dist',dist); % Ajout à la structure datas les coordonnées et les distances
27 end
28
29 save pointsAllK10.mat datas; % Sauvegarde de la structure datas dans le fichier pointsAllK10
30 disp('DONE');
```

Au final, nous arrivons donc à deux types de représentations différentes : les coordonnées des points et les distances entre les 10 plus proches voisins. Ces données sont sauvegardés dans une structure Matlab que j'ai appelé **datas** et qui se trouve dans le fichier *pointsAllK10.mat* qui sera chargé via le programme de FGPLVM. Les données à l'intérieur de la structure **datas** sont séparés en 2 colonnes : la colonne **points** correspondant aux coordonnées (x,y) des 66 features, et la colonne **dist** correspondant aux distances de

chaque points avec ses 10 plus proches voisins. Cette représentation est illustré en [Figure 3](#).

3 Utilisation de l'algorithme F-GPLVM (Faster GPLVM) pour créer l'espace latent

Là où le programme d'origine utilisait l'algorithme GPLVM développé par Neil Lawrence en 2004, je vais utiliser sa version améliorée, FGPLVM, également développé par Lawrence en 2007.

Ce programme est très volumineux et utilise plus de 15 bibliothèques annexes, également toutes créées par Lawrence. Le programme complet avec toutes les bibliothèques peut être trouvé à cette adresse [[WWW4](#)].

Je vais maintenant vous expliquer son fonctionnement globale (dans le cas où l'on se sert des coordonnées des 66 *features*), sans rentrer dans des détails bien trop compliqués.

Tout d'abord, nous allons charger les données sauvegardées précédemment dans le fichier *pointsAllK10.mat*, via la fonction `lvmLoadData(nom_dataset)`. Cette fonction prend en paramètres le nom des données à charger et va renvoyer 2 choses :

- Y : correspondra à nos coordonnées des 66 points de chaque image
- lbls : les labels correspondant aux émotions pour les classifier

Le code chargeant les données est le suivant :

```
1 load pointsAllK10.mat; % chargement des données contenues dans le fichier
2 N=size(datas,2); % nombres de données (images), ici 75
3 L=size(datas(1).points,1); % nombres de points, ici 66
4 C=size(datas(1).points,2); % dimension des coordonnées, ici 2
5 data = zeros(N,L*C);
6 lbls={'joie','degout','tristesse','colere','surprise'}; % labels correspondant aux émotions
7 for i=1:N % Pour chaque images...
8     data(i,:)=reshape(datas(i).points,L*C,1); % on redimensionne ses coordonnées 66*2 en 132*1
9 end
10 Y=data; % taille finale = 75*132
```

Nous allons ensuite créer le modèle de données que le programme utilisera tout au long de la construction de l'espace grâce à ce que nous venons de récupérer. La fonction `fgplvmCreate(q, d, Y, options)` permet de nous renvoyer le modèle initialisé et prend en paramètres :

- q : la dimension souhaitée de l'espace latent, dans notre cas 2
- d : le nombre d'observations (images), ici 75
- Y : les données récupérées via `lvmLoadData()`
- options : une structure contenant les options par défaut pour le type d'approximation choisi, ici "ftc"

Une fois le modèle créé, nous y ajoutons du dynamisme. Cela va permettre de lisser les courbes obtenues et d'avoir des résultats plus précis. Pour cela il nous faut définir des options supplémentaires, tel que le type de noyau qui va être utilisé.

```
1 % Add dynamics model.
2 options = gpOptions('ftc');
3 options.kern = kernCreate(model.X, {'rbf', 'white'});
4 options.kern.comp{1}.inverseWidth = 0.2;
5 % This gives signal to noise of 0.1:1e-3 or 100:1.
6 options.kern.comp{1}.variance = 0.1^2;
7 options.kern.comp{2}.variance = 1e-3^2;
8 model = fgplvmAddDynamics(model, 'gp', options);
```

Une fois cela fait, il ne nous reste plus qu'à optimiser le modèle pour obtenir la position finale de nos points dans l'espace latent. Il nous suffit de choisir le nombre d'itération maximal qui va être utilisé pour l'optimisation. Je me suis aperçu, au cours des différents tests que j'ai fait, qu'au dessus de 100 itérations, les différences observées étaient minimales. J'ai donc choisi de bloquer le nombre d'itérations à 100 pour le reste de mon projet.

```

1 % Optimise the model.
2 iters = 100;
3 display = 1;
4 model = fgplvmOptimise(model, display, iters);

```

Je fini ensuite par sauvegarder le modèle obtenu, ainsi que les labels des émotions, dans un fichier **model.mat** qui va donc contenir toutes les données de mon modèle optimisé permettant de construire l'espace latent.

4 Représenter et comparer des trajectoires dans l'espace latent

Maintenant que nous avons nos données, il ne nous reste plus qu'à les utiliser pour construire l'espace latent.

Construction et affichage de l'espace latent

Pour cela, seul l'appel à une fonction est nécessaire : *lvmVisualise(model, YLbIs, visualiseFunction, visualiseModify, varargin)*. Ses paramètre sont notre modèle, nos labels d'émotions, la fonction de visualisation utilisée et la fonction de modification de la visualisation utilisée. Dans mon cas, j'appelle donc la fonction de cette façon :

```

1 lvmVisualise(model, lbls, ['vector' 'Visualise'], ['vector' 'Modify']);

```

Cette fonction va donc faire le nécessaire pour créer l'espace latent, avec entre autres le bon dimensionnement des axes. La partie qui nous intéresse se trouve dans le fichier **lvmTwoDPlot.m** et est la suivante :

```

1 labelNo=1;
2 deb=1;
3 for i = 1:(size(X, 1)/15)
4     returnVal = [returnVal; plot(X(deb:15*i, 1), X(deb:15*i, 2), symbol{labelNo}, ←
5         'linewidth',1)];
6     if labelsString
7         textReturnVal = [textReturnVal; text(X(15*i, 1), X(15*i, 2), ['    ' lbl{i}])];
8     end
9     labelNo = labelNo+1;
10    deb=(15*i)+1;
11 end

```

C'est dans cette boucle que je vais afficher les trajectoires correspondnats aux émotions de la base d'apprentissage.

size(X, 1)/15 va nous retourner le nombre d'émotions contenu dans notre base d'apprentissage, car chaque émotion se compose de 15 images et donc de 15 points à afficher.

Je stocke ensuite les courbes à tracer dans la variable **returnVal** grâce à la commande *plot()* qui permet de tracer des courbes en Matlab. Je lui passe comme paramètres les coordonnées en x des 15 points de l'émotion, les coordonnées en y, le symbol avec lequel s'affichera la trajectoire ainsi qu'un paramètre spécifiant l'épaisseur du trait, ici défini à 1.

La variable **textReturnVal** va me permettre d'afficher le nom de l'émotion à côté de sa courbe, plus précisément à côté de son dernier point, d'où le fait du choix *15*i* correspondant au dernier point de la courbe. Le nom de l'émotion est donc récupéré depuis *lbl* qui contient tous les labels des émotions.

Je fini par incrémenter correctement mes variables pour que je puisse bien parcourir toutes les données présentes dans ma base d'apprentissage.

Au final, en utilisant les coordonnées des 66 points comme données pris en compte pour la création du modèle, l'espace latent résultant est présenté en **Figure 4**.

Maintenant que l'espace est construit, il faut encore que j'arrive à afficher une trajectoire supplémentaire correspondant à des données prises dans une base de tests.

Affichage d'une trajectoire à classifier dans l'espace latent

Il n'est pas possible de faire passer ces données par le même parcours qu'on suivit celles de l'apprentissage puisqu'il n'est plus question maintenant de construire l'espace mais seulement de projeter des données dessus.

Après avoir passé plusieurs semaines bloqués sur ce point, pour avoir la solution à ce soucis, j'ai demandé conseil à Giuseppe Boccignone. Grâce à son aide précieuse, j'ai pu découvrir exactement la fonction qui résolvait ce problème : *fgplvmOptimisePoint(model, x, y, display, iters)*. Elle prend en paramètre le modèle ayant permis de construire le modèle, des coordonnées d'initialisation *x*, les données pour lesquels nous souhaitons trouver les coordonnées dans l'espace, un booléen pour l'affichage ou non des itérations, ainsi que le nombre d'itération pour optimiser les valeurs.

La particularité de cette fonction est le fait de devoir lui donner des coordonnées d'initialisation pour obtenir la position des données à optimiser *y* dans l'espace.

Pour récupérer cette position d'initialisation, j'ai créé la fonction suivante :

```

1 function initXPos = getNearestValue(Ymodel, Y)
2     ytemp = 0;
3     dist = Inf(1);
4     for i=1:size(Ymodel,1)
5         D = sqrt(sum((Ymodel(i,:) - Y).^2));
6         if(D < dist)
7             dist = D;
8             ytemp = i;
9         end
10    end
11    initXPos = ytemp;
12 end

```

Elle prend donc en paramètre les données brutes ayant servi à construire l'espace *Ymodel* et les données à classifier *Y*. Je vais donc chercher à laquelle des données d'*Ymodel* se rapproche le plus notre donnée *Y*. Je calcule donc simplement la distance euclidienne et à la fin de la fonction je retourne l'indice de la donnée la plus proche.

Cette indice va ensuite me permettre de récupérer, dans le modèle de l'espace, les coordonnées du point correspondant dans l'espace latent et ces donc ces coordonnées qui vont servir ensuite de point d'initialisation pour la fonction *fgplvmOptimisePoint()*.

En appliquant cette dernière aux 15 données d'une émotion de test prise en entrée, j'obtiens les 15 couples de coordonnées correspondant à leur position dans l'espace latent.

Pour vérifier le bon fonctionnement du programme, j'ai choisi de reprendre les données correspondant à l'émotion de la joie qui ont servi pour construire l'espace latent. Selon toute logique, les points étant les mêmes, ils devraient se superposer. Le résultat de ce test est en [Figure 5](#).

Représenté par des triangles bleus clairs, nous pouvons nous apercevoir que la superposition avec l'émotion de la joie apprise lors de la construction n'est pas totale. Cependant, et cela est le plus important, la forme de la trajectoire est sensiblement la même. En effet, nous ne souhaitons pas forcément que les points se superposent mais plutôt que les 2 trajectoires aient la même forme dans le même espace.

Classification des trajectoires

Je n'ai malheureusement pas eu le temps d'implémenter une solution permettant de comparer 2 trajectoires basé sur leur forme.

Cependant, j'ai tout de même mené plusieurs recherches sur le sujet et trouvé quelques algorithmes pouvant être intéressants.

Tout d'abord Srivastava et al. présente en 2011 [25] une distance entre trajectoires permettant de détecter efficacement des formes similaires mais elle impose à ces dernières d'être de la même longueur. Elle est cependant indépendante de la distance physique entre les 2 trajectoires.

Ensuite, Lin et Su en 2005 [15] ont créé la distance OWD, One-Way-Distance, qui elle ne se base que sur la forme des trajectoires pour les comparer.

Besse et al. [2] ont présenté une modification de la distance OWD de Lin et Su. Elle se base sur la distance physique entre les 2 trajectoires, sur leur forme et sur leur "indépendance temporelle".

Au final, la distance qui m'a paru la plus adaptée pour notre problème est celle introduite par Lin et Su en 2005 : **OWD**. Vu qu'elle ne se base que sur la forme des trajectoires, elle s'adapte parfaitement à notre classification d'émotion. En effet, les émotions que nous essayons de classer ne sont pas forcément à la même position que celles apprises lors de la création de l'espace, néanmoins elles sont censées avoir la même forme, ou du moins une forme similaire.



Figure 1 – Images de notre base d'apprentissage pour l'émotion de la joie

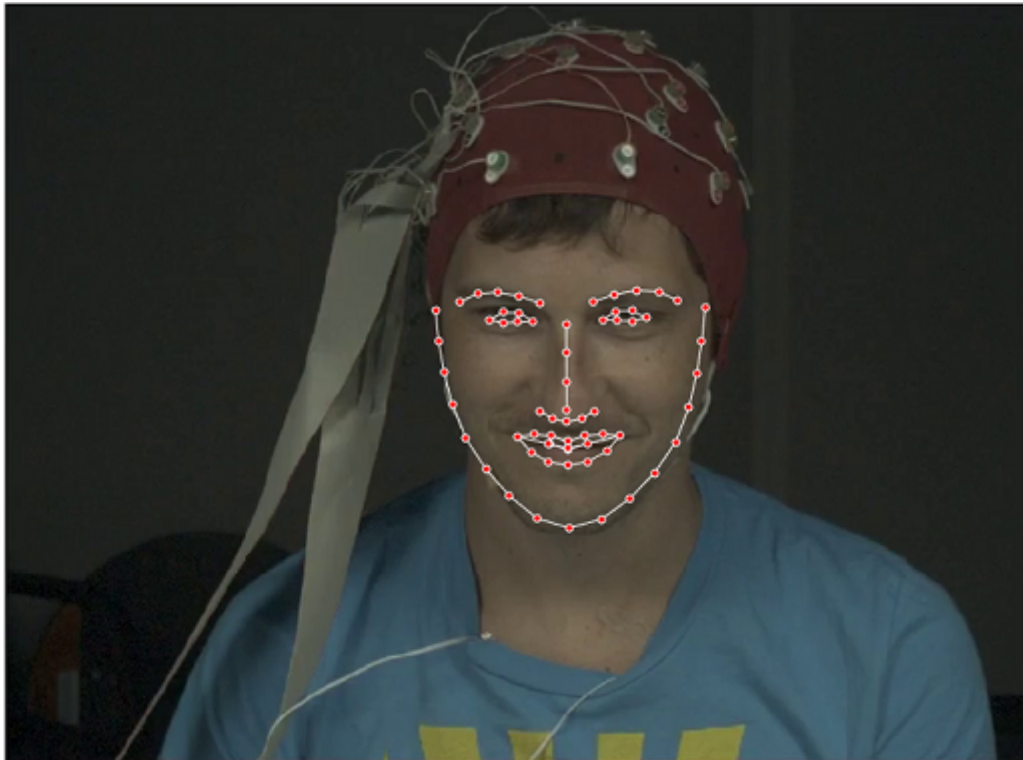


Figure 2 – 66 features du visage trouvé grâce au programme de A .Asthana

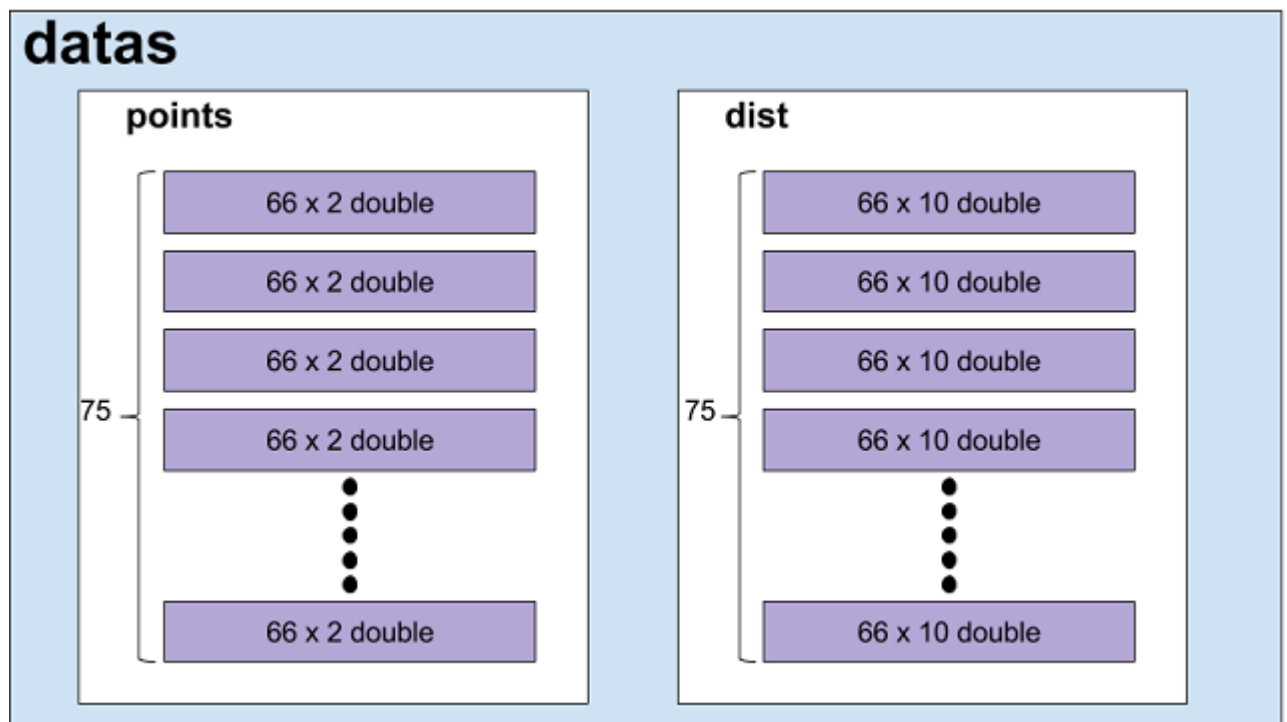


Figure 3 – Structure utilisée pour stocker les données

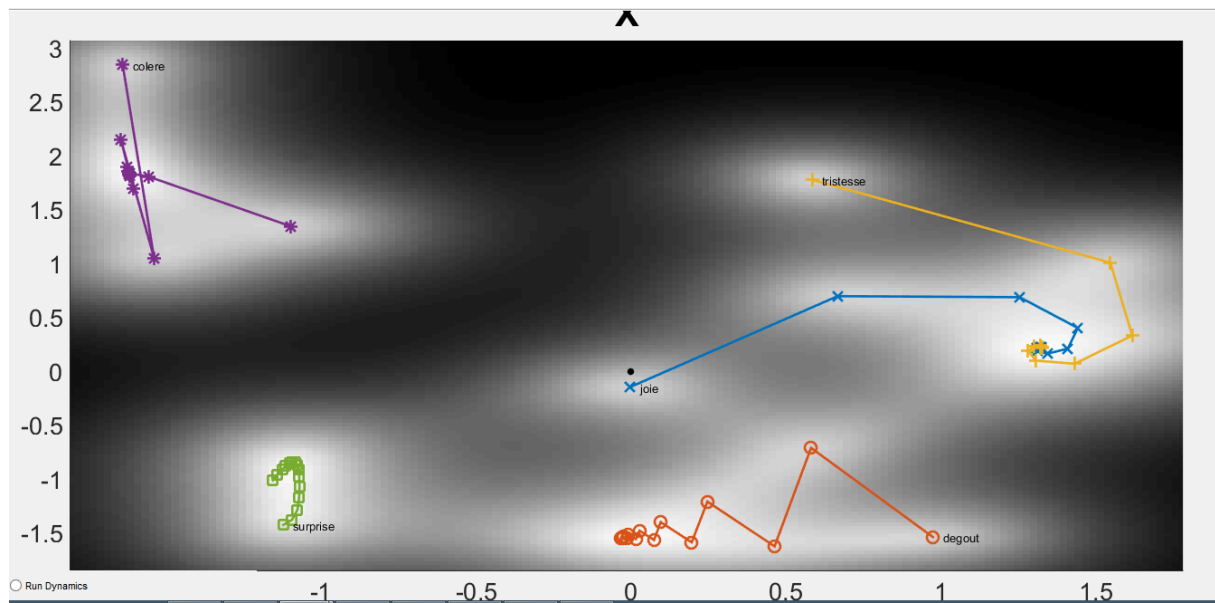


Figure 4 – Espace latent permettant de représenter les émotions.

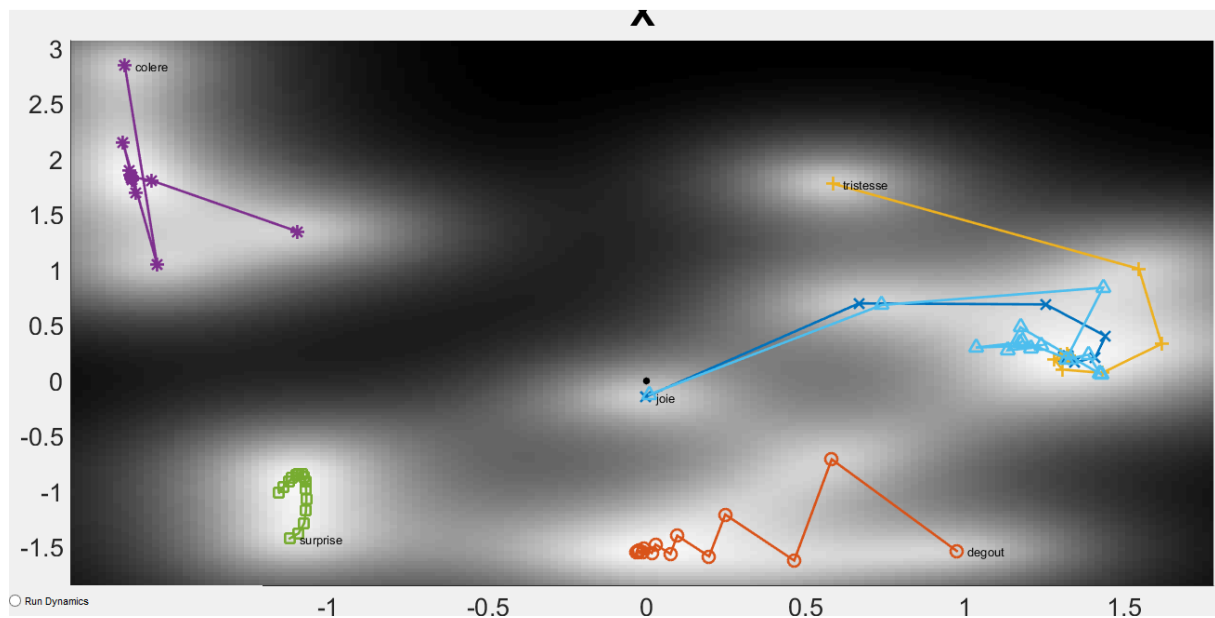


Figure 5 – Test : représentation de l'émotion de la joie utilisée lors de la construction de l'espace latent

11

Tests et résultats

Comme je l'ai dit précédemment je n'ai malheureusement pas eu le temps d'implémenter une fonction permettant de classifier les émotions en fonction de la forme de leur trajectoire.

Cependant, j'ai pris le temps de faire la base de test qui va ensuite permettre de réaliser une matrice de confusion décrivant l'efficacité de mon système de reconnaissance faciale d'émotions.

Ma base de test est définie dans le [Table 1](#)

Table 1 – Données utilisées dans ma base de test

	N° sujet	N° session	Émotion
1	1	2	dégoût
2	3	274	dégoût
3	6	652	dégoût
4	7	792	dégoût
5	14	1720	dégoût
6	1	8	joie
7	1	12	joie
8	6	654	joie
9	7	782	joie
10	14	1692	joie
11	1	4	surprise
12	1	22	surprise
13	1	24	tristesse
14	1	32	tristesse
15	3	262	tristesse
16	7	790	tristesse
17	14	1704	tristesse
18	14	1702	colère

J'ai également mesuré le temps total mis pour afficher l'espace latent avec des données de tests. Matlab dispose d'une fonction permettant de mesurer automatiquement ce temps, ainsi que le temps passé dans chacune des fonctions appelées. Les résultats obtenus sont présentés en [Figure 1](#).

Le temps total de cette partie de l'application est donc d'environ 3.5 secondes, ce qui est correct. Il manque cependant le temps de récupération des données des *features* du visage ainsi que le temps de

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
displayLatentSpace	1	3.462 s	0.016 s	
lvmVisualise	1	3.446 s	0.098 s	
lvmScatterPlot	2	3.260 s	0.111 s	
lvmSetPlot	1	2.867 s	0.032 s	
lvmTwoDPlot	2	2.406 s	0.031 s	
fgplvmOptimisePoint	15	2.374 s	0.000 s	
scg	15	2.374 s	0.173 s	
cmpndKernCompute	1849	2.122 s	0.261 s	
fgplvmPointGradient	697	1.748 s	0.007 s	
fgplvmPointLogLikeGradient	697	1.741 s	0.137 s	
gpPosteriorMeanVar	1144	1.185 s	0.185 s	
kernCompute	7396	1.068 s	0.205 s	
gpPosteriorGradMeanVar	697	0.786 s	0.167 s	
modelPosteriorVar	4	0.711 s	0.000 s	
gpPosteriorVar	4	0.711 s	0.333 s	
fgplvmPosteriorVar	4	0.711 s	0.000 s	
rbfKernCompute	1849	0.460 s	0.189 s	

Figure 1 – Temps total passé dans chaque fonction lors de l’affichage de l’espace latent avec des données de test

comparaison de trajectoires, non implémenté à l’heure actuelle.

12

Notice d'utilisation de mon application

Pour que mon projet puisse être repris par la suite pour être amélioré, je vais maintenant présenter un mode d'emploi permettant de l'utiliser.

J'ai découpé ce mode d'emploi en 3 parties :

- L'extraction des coordonnées des *features* des visages
- La génération de l'espace latent grâce aux données de la base d'apprentissage
- L'affichage de l'espace latent avec des données de test supplémentaires

1 Extraction des coordonnées des *features* des visages

L'extraction de coordonnées se passe dans le fichier *extractFeaturesCoordinates.m*.

Il vous faut tout d'abord spécifier le type de détecteur de visage que vous souhaitez utiliser via la variable **bbox_method**. Les possibilités sont les suivantes :

- 0 : Détecteur de visage basé sur des arbres. Très précis et adapté pour rechercher des visages dans des images chargés mais extrêmement lent.
- 1 : Détecteur de visage Matlab ou votre propre détecteur de visage. Pour appliquer votre propre détecteur de visage, il vous faudra modifier le fichier *External_Face_Detector.m* se trouvant dans le dossier *features extraction*.
- 2 : L'utilisation de boîtes précalculées.

Vous devez ensuite renseigner si vous souhaitez ou non visualiser sur chaque image la détection des features avec **visualize**.

Une fois ces paramètres choisis, vous devrez spécifier le nombre de différentes images par émotions (**numberImages**, 15 dans mon cas) ainsi que le nombre d'émotions qui vont être analysé (**numberEmotions**, 5 dans mon cas).

Également à spécifier, le nombre des K plus proches distances entre chaque points que vous souhaitez calculer. Si vous ne souhaitez pas calculer cela, mettez simplement le paramètre **computeNearestDistances** à 0.

Une fois tous les paramètres renseignés, lancer le programme. Lorsqu'il aura fini, le message "DONE" s'affichera et vous pourrez trouver toutes les données qui ont été calculées dans le fichier *coordinates.mat*.

2 Génération des données de construction l'espace latent

La génération des données de construction l'espace latent se passe dans le fichier *genLatentSpace.m*.

Il vous faut tout d'abord donner le nom du fichier dans lequel les données précédemment récupérées ont été sauvegardé. Si vous n'y avait pas touché dans l'étape précédente, laissez la valeur par défaut de **coordinates.mat**.

Vous devez ensuite définir les valeurs de labels des émotions qui vont être apprises à l'intérieur de la variable **Ibls**, en faisant attention à l'ordre de ces labels. Ils doivent être définis entre simples guillemets séparés par des virgules, le tout compris entre accolades.

Également à définir, le nombre d'itérations que vous souhaitez utiliser pour optimiser les valeurs, via **iters**. Par défaut, cette valeur est 100.

Pour finir, il vous faut spécifier, grâce au booléen **getKDist**, avec quelle donnée vous souhaitez générer l'espace latent. En mettant cette variable à 1, vous générerez l'espace avec les K plus proches distances pour chaque points et en la mettant à 0, vous générerez l'espace grâce aux coordonnées de chaque points. Par défaut, cette valeur est 0.

Information importante à préciser : vous ne devez pas changer le nom *model.mat* qui va permettre de sauvegarder ces données. Si jamais vous souhaitez réellement changer ce nom de fichier, vous devrez également aller le changer dans le fichier *FGPLVM/mltools/lvmTwoDPlot*, à la ligne 61.

3 Affichage de l'espace latent ainsi que des données de test

L'affichage de l'espace latent ainsi que des données de test se passe dans le fichier *displayLatentSpace.m*.

Vous devez tout d'abord renseigner le nom de l'objet dans lequel les données de l'émotion à tester sont sauvegardées. En effet, pour tester une émotion, il faudra également la faire passer par la phase *extractFetauresCoordinates.m*.

Par défaut, cette valeur est **pointsTest.mat** mais vous devrez bien évidemment la changer par le nom adéquat.

Le second paramètre à spécifier est **getKDist** et est similaire à celui se trouvant dans le fichier *genLatentSpace.m*. En le mettant à 1, vous utiliserez les valeurs des K plus proches distances de chaque point pour tester l'émotion de test et en la mettant à 0, vous utiliserez les coordonnées.

Bien évidemment, il faut que cette variable est la même valeur dans *genLatentSpace.m* et *displayLatentSpace.m* pour obtenir des résultats cohérents.

13

Gestion de projet et planning final

1 Méthodologie

La méthodologie que j'ai utilisé est la méthodologie agile. Je me suis fixé des objectifs consécutifs à atteindre avec des retours hebdomadaires à mes encadrants.

2 Gestion de version

Pour versionner mon projet, j'ai utilisé Git ainsi qu'un dépôt GitHub public [WWW11]. J'ai également utilisé un dépôt GitHub pour versionner ce rapport [WWW12].

3 Planning final

Je vous avais présenté le planning prévisionnel que j'avais prévu à la fin de la section de recherche de mon projet (cf [Section 1](#) (Chapitre 7)). Voici donc le planning réel final :

Table 1 – *Planning du projet*

Partie Recherche				
Nom	Date début	Durée	Date Fin	Livrable
Comparaison 2D VS 3D	16/09/15	1 semaine	24/09/15	Compte-rendu
Architecture	23/09/15	1 semaine	01/09/15	Powerpoint
Étude techniques extraction de <i>features</i>	30/09/15	6 semaines	12/11/15	Powerpoint
Étude bases de données	28/10/15	1 semaine	05/11/15	Powerpoint
Étude article/programme Vitale et al.	19/11/15	7 semaines	07/01/16	
Rédaction rapport de recherche	08/10/15	13 semaines	05/01/16	Rapport L ^A T _E X
Préparation soutenance	05/01/16	1 semaine	14/01/16	Powerpoint
Partie Développement)				
Nom	Date début	Durée	Date Fin	Livrable
Identification des modifications	14/01/16	2 semaines	02/02/16	
Rédaction article	20/01/16	3 semaines	10/02/16	
Modification : Changement de la base d'apprentissage	27/01/16	1 semaine	04/02/16	
Modification : Changement de la représentation des données	10/02/16	3 semaines	02/03/16	
Modification : Prise en main FGPLVM	10/02/16	2 semaines	25/02/16	
Modification : Représentation des trajectoires (apprentissage + test)	18/02/16	4 semaines	23/03/16	
Rédaction rapport développement	17/02/16	5 semaines	23/03/16	Rapport L ^A T _E X
Préparation soutenance	24/03/16	1 semaine	31/03/16	Powerpoint



Conclusion

Avoir eu l'occasion de passer près de 4 mois à faire de la recherche pour ce projet a été très intéressant. Faire un état de l'art des différentes techniques et technologies disponibles pour faire de la reconnaissance faciale d'émotions m'a permis de m'ouvrir sur un monde totalement différent de ce que je connaissais jusqu'à présent.

La première partie de mon projet de recherche et développement m'a également montré ce que cela pouvait être de faire de la recherche dans le cadre universitaire dans le but de faire de nouvelles découvertes ou de concevoir des applications concrètes pouvant aider au quotidien.

J'ai ensuite pu, pendant 3 mois environ, développer mon application en mettant en œuvre certaines des recherches que j'ai pu faire.

Le fait de récupérer un programme existant et de le modifier, en accord avec un des auteurs d'origine, a également été bénéfique. Cela m'a permis de travailler en collaboration avec une personne spécialiste dans ce domaine.

Contrairement à ce qu'il avait été prévu dans le planning prévisionnel, je n'ai pas pu finir l'application à 100%.

En effet, à cause d'une difficulté que j'ai pu rencontrer et qui m'a paralysé plusieurs semaines, je n'ai pas eu le temps d'implémenter un algorithme de comparaison de trajectoire basé sur la forme. J'ai cependant pu mener des recherches sur cette partie qui vont, je l'espère, permettre une implémentation rapide et efficace.

Hormis l'implémentation d'un algorithme de comparaison de trajectoire basé sur la forme, plusieurs points peuvent être améliorés dans ce programme.

Tout d'abord, augmenter la base d'apprentissage et prendre plusieurs set d'images de différentes personnes pour représenter une émotion en faisant la moyenne entre chaque frame correspondantes.

Cela pourrait permettre un apprentissage moins dépendant de l'expression d'un sujet pour une émotion et donc plus performant, en prenant en compte différentes expressions pour représenter la même émotion.

Un second point qui pourrait être amélioré est la normalisation des données. En effet, pour l'instant nous utilisons les coordonnées des *features* des visages ou les K plus proches distances entre ces points. Cependant, ces valeurs sont dépendantes de la position du visage dans l'image et également de la forme du visage de la personne.

Pour remédier à cela et obtenir des valeurs moins dépendantes de la position et de la forme des visages, une normalisation des valeurs peut être effectué, en ramenant toutes les valeurs entre 0 et 1 par exemple.

Pour finir, je tiens à remercier mes tuteurs académiques, Mr Mohamed SLIMANE et Mr Donatello CONTE, de m'avoir fait confiance pour la réalisation de ce projet ainsi que pour leur aide et disponibilité tout au long de ce projet. Je remercie également Mr Giuseppe Boccignone pour l'aide qu'il a pu m'apporter

quand j'en avais besoin.

Annexes



Article Handicap 2016

Veillez trouver ci-après l'article soumis à la conférence Handicap 2016 de l'IFRATH.

Une nouvelle approche à la détection d'émotions pour l'aide à l'interaction homme-machine chez les personnes handicapées

Florian Tissier*, Donatello Conte*, Giuseppe Boccignone[†] et Mohamed Slimane*

*Université de Francois Rabelais, Laboratoire LI EA 6300, Tours, France

Email : florian.tissier@etu.univ-tours.fr, {donatello.conte, mohamed.slimane}@univ-tours.fr

[†]Dipartimento di Informatica, Università degli Studi di Milano, Milano 20135, Italy

Email : giuseppe.boccignone@unimi.it

Résumé—Dans cet article une nouvelle approche à la détection d'émotions, par analyse de vidéos, est présentée. Dans le domaine de recherche de l'analyse de vidéos, la reconnaissance d'émotions est une application qui intéresse de plus en plus les chercheurs, pour améliorer l'interaction entre les hommes et les machines. La recherche est encore tout à fait ouverte, car aucune approche présente dans la littérature scientifique a obtenu de bonnes performances, surtout si on considère des vidéos non contrôlées et non construites *ad hoc* pour la détection d'émotion. La problématique est encore plus difficile en présence de personnes handicapées dans la vidéo. Récemment des approches probabilistes ont montrés leur efficacité. L'approche présentée dans cet article s'inspire de ces travaux et propose de nouvelles modifications afin d'avoir de meilleures performances de détection.

I. INTRODUCTION

L'interaction entre les hommes et les machines a toujours été un enjeu de taille. Arriver à faire communiquer un ordinateur avec un être humain est un défi de tous les jours et est de plus en plus présent dans notre quotidien.

C'est dans cette optique de facilitation du quotidien, et plus précisément de facilitation du quotidien de personnes handicapées, grâce à l'interaction avec une machine, que nos travaux prennent place.

Cet article va décrire les travaux que nous sommes en train de réaliser pour construire un système permettant, à partir d'un flux vidéo acquis grâce à une caméra, de détecter l'expression actuelle du visage d'une personne handicapée. Cette détection d'expressions, et donc d'émotions, se fait dans le but de réaliser des actions, comme par exemple changer la couleur de la lumière de la pièce, jouer de la musique, diffuser un certain parfum relaxant, etc., pouvant améliorer le bien-être de la personne.

Nous allons tout d'abord vous présenter un état de l'art des différentes approches pour un système de reconnaissance faciale d'expressions.

Dans un second temps, sachant que la détection d'émotions chez des personnes handicapées est difficile et que de nouvelles méthodes sont nécessaires, nous vous introduiront une nouvelle approche permettant de reconnaître des émotions via un modèle probabiliste, introduit par Vitale et al. [1]. A partir

de ce modèle nous avons introduit des améliorations pour augmenter les taux de reconnaissance de notre application. Nous finirons par présenter les expérimentations et les résultats obtenues grâce aux modifications que nous aurons apporté. Ces travaux ont été réalisés dans le cadre d'un projet de recherche et développement au sein de l'école d'ingénieur Polytech Tours, dans le Laboratoire d'Informatique de l'Université de Tours.

II. ÉTAT DE L'ART

Dans cette section, nous allons voir un état de l'art des différentes parties qui composent l'architecture d'un système de reconnaissance faciale d'expressions, ainsi que plusieurs bases de données permettant de réaliser l'apprentissage d'un tel système.

Nous ferons ensuite une critique d'un tel système et de ses limitations dans sa forme actuelle par rapport à notre problématique.

Mais tout d'abord, nous allons définir les émotions basiques que nous allons reconnaître via notre système de reconnaissance.

A. Émotions universelles

Définies par Paul Ekman et Wallace Friesen en 1971 dans leur article [2], les 7 émotions universelles sont les suivantes :

- la neutralité
- la joie
- la tristesse
- la colère
- la surprise
- la peur
- le dégoût

Ces émotions sont dites universelles car tout individu est capable de les exprimer mais est également capable de les reconnaître et de les interpréter chez autrui.

Ces 7 émotions sont utilisées dans la plupart des systèmes réalisant de la reconnaissance d'émotions à l'heure actuelle et c'est également celles-ci que nous allons chercher à reconnaître.

À noter également qu'une expression se divise en 3 phases :

l'**onset** (correspondant au début de l'expression), l'**apex** (le pic de l'expression) et l'**offset** (la fin de l'expression).

Ekman et Friesen ont également créé en 1978 une norme de description des mouvements faciaux permettant de décomposer les expressions. Cette norme se nomme FACS, Facial Action Coding System [4]. Elle se compose de 46 Action Units (AU) permettant de représenter la contraction ou décontraction des muscles faciaux. La composition de plusieurs AUs permet donc de décrire une expression.

B. Architecture d'un système de reconnaissance faciale d'expressions

Un tel système se compose de 3 parties qui sont : détection du visage dans une image, extraction des *features* (points clés) de ce dernier et la classification des *features* pour obtenir l'émotion associée.

1) *Détection du visage*: Il existe 2 types de détecteurs de visages, les absolus et les différentiels.

Tout d'abord les détecteurs absolus. Ils permettent de déterminer la position d'un visage sur chaque *frame* d'un flux vidéo, indépendamment des *frames* précédentes. Le premier détecteur absolu a été inventé par Paul Viola et Mickaël Jones en 2001 [3] et il est encore à l'heure actuelle la référence de ce type de détecteur d'objets.

Les détecteurs absolus quant à eux vont repérer un visage en utilisant sa position dans la *frame* précédente. Si la position d'un visage est connue à l'instant t , le détecteur différentiel va se servir de cette position pour trouver celle à l'instant $t + 1$. Bien sûr, il faut initialiser la position à l'instant 0 pour faire fonctionner ces détecteurs. Pour cela il est possible d'utiliser un détecteur absolu sur la toute première *frame* du flux vidéo. L'avantage de ces détecteurs est leur grande rapidité et précision. Cependant, l'inconvénient est que l'accumulation de petites erreurs sur les positions peut mener à la "dérive" et donc la perte du visage.

2) *Extraction des features*: La deuxième partie d'un système de reconnaissance faciale d'émotions est l'extraction des *features* du visage précédemment repéré. Ces features sont les points clés d'un visage et se composent par exemple de la position du coin des yeux, des lèvres, du nez, des joues, de la position de la pupille, etc.

Pour extraire ces points et leurs positions, plusieurs méthodes existent.

Nous pouvons par exemple citer les filtres de Gabor qui appliquent un filtre linéaire à chaque pixel d'une image et dont la réponse est une sinusoïde modulée grâce à une fonction gaussienne. Un filtre de Gabor possède plusieurs paramètres lui permettant de pouvoir s'adapter à tous les types de problèmes de détection d'objets ou de textures dans une image.

Une autre méthode existante est celle des composantes pseudo-Haar. Elle a été introduite par Viola et Jones lors du développement de leur détecteur de visage [3]. Elle consiste à appliquer un masque en forme de "boîte" (composé d'une

partie noire et d'une partie blanche) à toutes les positions possibles sur l'image, tout d'abord de petite taille (20*20 pixels par exemple) puis en les agrandissant. A chaque position, on soustrait la somme des pixels contenus dans la partie noire à la somme des pixels contenus dans la partie blanche. Le résultat d'une caractéristique à une certaine position est donc un nombre réel qui va coder les variations du contenu pixelique. Cette méthode est d'une grande rapidité lorsqu'elle est utilisée avec des images intégrales ([3]).

D'autres méthodes existent pour réaliser la détection d'objet dans une image, telle que les motifs binaire locaux ou l'utilisation de flots optiques.

3) *Classification*: Une fois les features extraites, il ne reste plus qu'à les analyser pour les classer.

Deux types de classificateurs existent : les systèmes basés sur des règles d'experts (*rule-based expert systems*) et les classificateurs avec apprentissage (*machine learning classifiers*). Les premiers sont de moins en moins répandus car ils sont beaucoup plus compliqués à utiliser que les autres. C'est donc sur les classificateurs avec apprentissage que nous allons nous concentrer. Parmi ces classificateurs, nous pouvons les classer en 2 catégories : les binaires et les multi-classes.

Comme l'indique son nom, un classificateur binaire ne permet de classer une instance qu'entre 2 classes (ex : sourire VS pas de sourire). Le classificateur binaire le plus connu à l'heure actuelle est le SVM [5] (Support Vector Machine, ou Séparateurs à Vaste Marge en français).

Contrairement aux binaires, les classificateurs multi-classes permettent de classer une instance selon un nombre de classes supérieur à 2. Un des classificateur multi-classes le plus connu est le k-PPV [6] (k Plus Proche Voisin) qui consiste à chercher les k échantillons (appris lors de la phase d'apprentissage) les plus proches d'un point pour ensuite retourner la classe majoritairement représentée par ces k voisins.

C. Base de données

Pour réaliser un système tel que nous l'avons présenté précédemment, une phase d'apprentissage est nécessaire pour la classification. Cette phase va donc utiliser des images se trouvant dans des bases de données de visages.

Nous avons regroupé plusieurs bases de données ainsi que leurs caractéristiques principales dans le tableau I.

Malheureusement, aucune base de données existante ne concerne les personnes handicapées.

D. Critiques

A l'heure actuelle, tout système de reconnaissance faciale d'émotions tel que présenté précédemment ([10], [11], [12]) a atteint ses limites. Tout d'abord du fait que pour obtenir des résultats convenables, les expressions devant être analysées doivent être posées et donc exagérées, ce qui diffère catégoriquement de la vie réelle. Ensuite la plupart des systèmes permettant la détection d'émotions se basent sur des techniques et base de données, telle que les FACS et Cohn-Kanade [7],

TABLE I
TABLEAU RÉCAPITULATIF DE BASES DE DONNÉES DE VISAGES 2D

Nom	P/S	Taille	Contenu	Label
CK	P	97 sujets (65% femmes, 15% afro-américains et 3% asiatiques ou sud américain) de 18 à 30 ans	486 séquences vidéos des 6 émotions basiques	FACS
CK+	P/S	+ 27%	+ 22%	FACS + émotion
MMI	P/S	75 sujets (48% de femmes, européen, africain et sud américain) de 19 à 62 ans	2900 vidéos des 6 émotions basiques ou d'AUs spécifiques	FACS
MHI Mimicry	S	40 sujets (28 hommes, 12 femmes) entre 18 et 40 ans	54 vidéos (sessions)	Auto-évaluation
HCI Tagging	S	27 sujets (11 hommes, 16 femmes) de 19 à 40 ans	Partie 1 : 20 vidéos ; Partie 2 : 28 images et 14 vidéos	Auto-évaluation
SAL	S	24 sujets	10 heures de vidéo : les sujets parle à une IA ayant différentes personnalités	Feeltrace
JAFFE	P	10 femmes japonaises	213 images des 7 émotions basiques	Noté selon 6 adjectifs d'émotion par 60 sujets Japonais

qui commencent à se faire vieille.

Dans la but d'innover et répondre au mieux à notre problématique de facilitation du quotidien de personnes handicapées, nous avons décidé d'utiliser des images de la base de données HCI-Tagging ([8]), dont les expressions présentées sont spontanées, pour l'apprentissage ainsi qu'un modèle probabiliste qui nous permettra de représenter les émotions sur un espace latent de dimension finie.

C'est maintenant ce modèle que nous allons vous présenter.

III. MÉTHODE PROPOSÉE

Dans un soucis d'innovation, et pour améliorer les performances, nous avons décidé d'utiliser un modèle probabiliste qui va nous permettre de représenter les émotions sur un espace fini en dimension finie. Ce modèle a été défini par Vitale et al. dans l'article *Affective Facial Expression Processing Via Simulation : A Probabilistic Model* [1] paru en 2014. Nous allons tout d'abord expliciter ce modèle probabiliste, puis nous introduirons les différentes modifications que nous avons apporté à ce modèle pour l'adapter à notre problématique de détection d'émotions sur les personnes handicapées.

A. Contexte

Pour comprendre les études menées dans l'article [1], poser le problème, et donc le contexte de la simulation de détection d'émotions, est nécessaire.

À un certain moment, un individu, appelé *acteur*, éprouvera un certain état interne, noté X_{act} . Cet état interne X_{act} peut soit être déclenché par un événement extérieur soit induit (remémoration d'un certain souvenir).

X_{act} va déclencher un comportement correspondant, noté

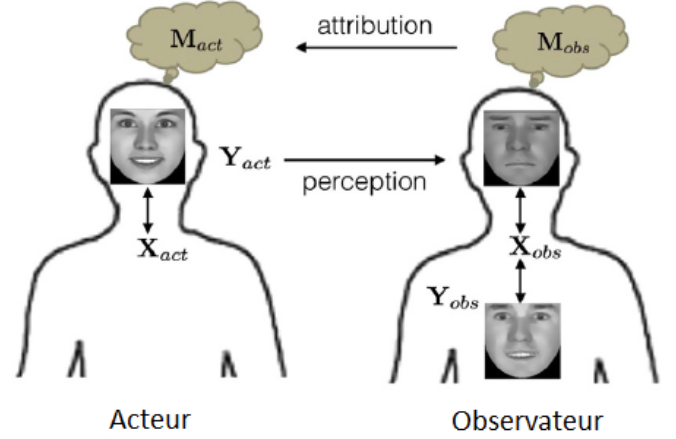


FIGURE 1. Contexte de la simulation du modèle probabiliste

Y_{act} , qui peut être par exemple une expression faciale, une posture particulière, un changement du rythme cardiaque, etc. Un second individu, appelé *observateur*, va maintenant utiliser Y_{act} pour trouver son état interne X_{obs} lui permettant d'être dans le même état mental M_{obs} que celui de l'acteur M_{act} . Si l'observateur est dans le même état mental que l'acteur, cela signifie que l'observateur a réussi à trouver à quoi correspondait Y_{act} et donc l'émotion de l'acteur.

Cette situation de simulation est illustrée en Figure 1 (figure tirée et traduite de [1]).

B. Modèle

Pour la réalisation du modèle probabiliste, plusieurs probabilités ont été définies :

- $P(Y_{obs}/Y_{act})$: représente la probabilité pour l'observateur de montrer l'état externe (expression faciale) Y_{obs} quand l'acteur affiche Y_{act} . Cette probabilité est appelée *transcodage*.
- $P(X_{obs}/Y_{obs})$: représente la probabilité d'être dans un état interne X_{obs} sachant Y_{obs} . Cette probabilité est appelée *correspondance inverse*.
- $P(Y_{obs}/X_{obs})$: représente la probabilité que l'observateur génère un état externe Y_{obs} sachant l'état interne X_{obs} . Cette probabilité est appelée *correspondance vers l'avant*.

Une fonction de décision $\mathcal{D}(\bullet)$ est également nécessaire et permettra de comparer l'état externe Y_{act} de l'acteur avec un état simulé $\tilde{Y}_{obs}$.

Ces probabilités nous permettent d'obtenir les variables suivantes :

$$y_{obs} \sim P(Y_{obs}/Y_{act} = y_{act}) \quad (1)$$

qui permet de définir le processus de transcodage transformant une instance de l'expression faciale de l'acteur y_{act} en une instance de représentation interne de l'observateur y_{obs} ,

$$x_{obs} \sim P(P(X_{obs}/Y_{obs} = y_{obs}) \quad (2)$$

qui décrit le processus de correspondance inverse donc la détection de l'état x_{obs} à partir de l'expression faciale (interne) y_{obs} , et

$$\tilde{y}_{obs} \sim P(Y_{obs}/X_{obs} = x_{obs}) \quad (3)$$

qui décrit le processus de correspondance vers l'avant et donc la correspondance de l'expression simulé par l'observateur $\tilde{y}_{obs}$ quand il se trouve dans l'état interne x_{obs} .

La fonction $\mathcal{D}(\bullet)$ compare le y_{obs} transcodé de l'équation (1) à l'expression générée intérieurement $\tilde{y}_{obs}$ de l'équation (3), et est également utilisé via l'équation (2) pour contrôler quand le processus de comparaison aura convergé vers la solution la plus probable.

Les processus de transcodage, de correspondance inverse et de correspondance vers l'avant se produisent dans l'espace latent d'auto-projection, noté Z , défini de la façon suivante :

$$P(Z) = \mathcal{N}(0, I_L) \quad (4)$$

qui permet de donner la priorité des points dans l'espace Z de dimension L ,

$$P(y_{act}/z) = \mathcal{N}(W_{act}z + \mu_{act}, \sigma_{act}^2 I_D) \quad (5)$$

qui permet d'avoir la probabilité d'obtenir l'expression faciale de l'acteur y_{act} , avec $D \gg L$, sachant $z \in Z$, et

$$P(y_{obs}/z) = \mathcal{N}(W_{obs}z + \mu_{act}, \sigma_{obs}^2 I_D) \quad (6)$$

qui permet d'avoir la probabilité d'obtenir l'expression faciale de l'observateur y_{act} , avec $D \gg L$, sachant $z \in Z$, avec :

$\mathcal{N}(\bullet)$: distribution Gaussienne; W_{act} et W_{obs} : paramètres de correspondance de l'acteur et de l'observateur; μ : moyenne; σ : variance; I_L et I_D : matrices identités de dimensions L et D .

On peut également définir les variables suivantes :

$$W = \begin{pmatrix} W_{act} \\ W_{obs} \end{pmatrix}, \mu = \begin{pmatrix} \mu_{act} \\ \mu_{obs} \end{pmatrix} \text{ et } \Phi = \begin{pmatrix} \sigma_{act}^2 I_D & 0 \\ 0 & \sigma_{obs}^2 I_D \end{pmatrix}.$$

Vu que le modèle est gaussien, y_{act} et y_{obs} suivent une distribution gaussienne de la forme $\mathcal{N}(\mu, \Sigma)$ avec $\Sigma = \Phi + WW^T$ correspondant à la matrice de covariance.

On obtient donc :

$$P(y_{obs}|y_{act}) \sim \mathcal{N}(\hat{\mu}_{obs}, \hat{\Sigma}_{obs}) \quad (7)$$

où $\hat{\mu}_{obs} = \mu_{obs} + \Sigma_c^T \Sigma_a^{-1}$ et $\hat{\Sigma}_{obs} = \Sigma_b - \Sigma_c^T \Sigma_a^{-1} \Sigma_c$ est le complément de Schur de Σ réécrit sous la forme du bloc $\Sigma = \begin{pmatrix} \Sigma_a & \Sigma_c \\ \Sigma_c^T & \Sigma_b \end{pmatrix}$.

L'équation (7) nous retourne la probabilité de correspondance de l'expression faciale de l'observateur sachant une expression faciale similaire de l'acteur. Pour faire cela, elle utilise W_{act} et W_{obs} en distribution normale multivariée.

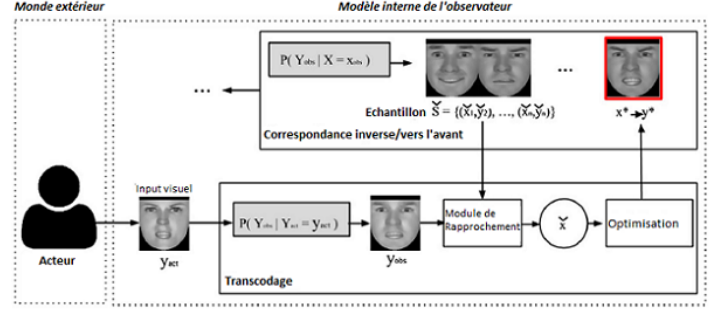


FIGURE 2. Fonctionnement du modèle probabiliste

Pour résoudre $\tilde{y}_{obs}$, on passe par l'utilisation du GPLVM (Gaussian Process Latent Variable Model) et l'on obtient

$$P(Y_{obs}|X_{obs}) = \frac{1}{\sqrt{(2\pi)^{ND} |K^D|}} \exp\left(-\frac{1}{2} \text{tr}(K^{-1} Y_{obs} Y_{obs}^T)\right) \quad (8)$$

avec Y_{obs} correspondant à l'ensemble d'apprentissage et K à la matrice de noyau dont les éléments sont définis par $(K)_{i,j} = \Phi(x_{i,obs}, x_{j,obs})$.

Une fois que $P(Y_{obs}/X_{obs})$ est appris, il est très facile d'obtenir $P(X_{obs}/Y_{obs})$ par GPLVM.

1) Fonctionnement: Le fonctionnement de ce modèle probabiliste, utilisant les notions précédentes, est décrit dans la Figure 2 ([1]). y_{act} est transcodé via l'espace latent d'auto-projection Z en y_{obs} .

Au même moment, un ensemble $\tilde{S}$ d'échantillons d'expressions de l'observateur est généré via l'équation (8). Avec $\mathcal{D}$, une mesure de similarité est utilisée pour évaluer la vraisemblance entre les échantillons de $\tilde{S}$ et de y_{obs} . L'état initial $\tilde{x}$ est sélectionné de cette manière :

$$\tilde{x} \in \tilde{S} | \tilde{x} \mapsto \tilde{y}_{obs} \wedge \tilde{y}_{obs} = \arg_y \max \mathcal{D}(\tilde{y}_{obs}, y)$$

2) Comparaison: Pour fonctionner, le module de rapprochement (cf Figure 2) utilise la mesure **SSIM** (Structural SIMilarity).

Cette mesure est effectuée sur plusieurs "parties" de l'image, appelées fenêtres, et ensuite une moyenne est calculée pour obtenir le résultat final. Le mesure entre 2 fenêtres x et y se calcule comme suit :

$$SSIM(x, y) = \frac{(2\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

avec μ_x et μ_y : moyennes de x et y ; σ_x^2 et σ_y^2 : variances de x et y ; σ_{xy} : covariance de x et y ; c_1 et c_2 : variables de stabilisation de la division quand celle-ci à un dénominateur faible.

C. Modifications

Pour aller de pair avec l'article [1], les auteurs ont développé un programme qui permet de visualiser les émotions sur l'espace latent Z . Cet espace est créé grâce

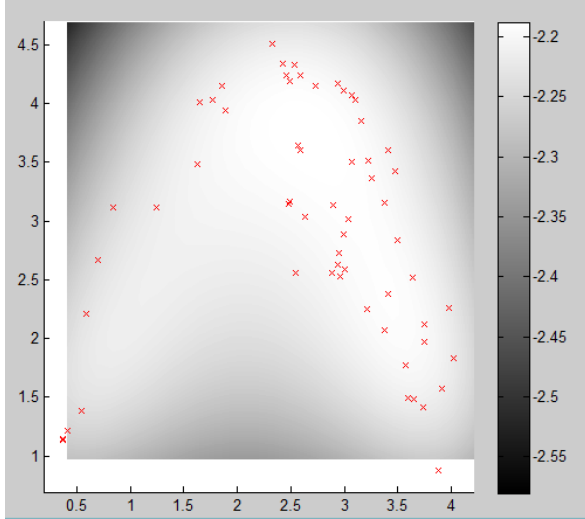


FIGURE 3. Espace latent Z représentant les émotions

à une base d'apprentissage de 60 images de taille 29*19 pixels. Cet espace est présenté en Figure 3. La principale fonctionnalité de ce programme est de pouvoir se déplacer sur l'espace Z et de voir en temps réel l'émotion associée.

Le modèle présentée est très intéressant et efficace, mais il présente encore quelque limitation. Dans cet article nous présentons les modifications apportées afin de l'améliorer.

Tout d'abord le modèle de représentation des données. Le modèle original traite les pixels d'une image et nous souhaiterions plutôt utiliser une représentation vectorielle qui représentera les mouvements du visage. Cela permettra d'être moins dépendante des pixels mais également du sujet présent sur l'image. Cela permettra, en plus, de ne pas utiliser l'algorithme PCA qui était utilisé dans le modèle original pour réduire le nombre de variables et qui est extrêmement chronophage.

Une deuxième amélioration consiste à analyser un ensemble d'images pour détecter une émotion, plutôt qu'une seule. En effet une émotion est composé, généralement, de plusieurs expressions faciales, et analyser plusieurs images d'une vidéo à la place d'une seule, nous permettrait d'être plus précis dans la reconnaissance. Dans notre modèle, cela se traduit à représenter une émotion, non plus par un point dans l'espace latent Z , mais plutôt à travers des trajectoires dans cet espace. Cela nous permettra aussi d'utiliser des modèles tels que les chaînes de Markov cachés (HMM [9]) pour la phase de reconnaissance.

IV. EXPÉRIMENTATION

A l'heure où nous écrivons la méthode n'as pas encore été testée. On a, cependant, défini le protocole de notre expérimentation et en particulier les données à utiliser pour l'apprentissage et pour les tests. Pour cela nous avons choisi 3 sujets de la base de données HCI Tagging [8] : les sujets 1,

TABLE II
LES SUJETS ET LES ÉMOTIONS CONSIDÉRÉES DANS NOTRE EXPÉRIMENTATION (EXTRAITS DE LA BASE HCI-TAGGING).

	N° sujet	N° session	Émotion
1	1	6	joie
2	3	264	dégoût
3	1	28	tristesse
4	6	662	colère
5	3	290	surprise

3 et 6. Pour chacun de ces sujets, nous avons choisi plusieurs vidéos correspondantes à différentes émotions : la joie, le dégoût, la surprise, la tristesse et la colère. Pour chacune de ces vidéos, nous en avons extrait les *frames* (1 toutes les 75) et en avons sélectionné 15 consécutives permettant de voir l'évolution d'une émotion au cours du temps et donc de construire la trajectoire. Les 15 *frames* consécutives ont été choisi de telle sorte que l'on puisse observer l'onset, l'apex et l'offset de l'émotion .

Le tableau II permet de présenter quels sujets nous avons choisi pour une émotion donnée, avec le numéro de la session correspondante dans la base de données HCI Tagging.

Une autre partie des mêmes vidéos utilisées pour l'apprentissage, sera employée pour les tests de détection. A l'heure actuelle nous sommes en train de terminer les expérimentations pour analyser ensuite les résultats.

V. CONCLUSION

Les systèmes de reconnaissance faciale d'expressions majoritairement utilisés à l'heure actuelle commencent à atteindre leurs limites, notamment lorsqu'ils doivent être utilisés avec des personnes handicapées.

De nouvelles méthodes et approches ont donc dû être mises en place et notamment la méthode probabiliste introduite par Vitale et al. [1]. C'est en se basant sur cette méthode et sur les travaux réalisés par ces même auteurs que nous avons commencé à construire notre application de reconnaissance d'émotions chez les personnes handicapées.

Plusieurs modifications ont été réalisées dans le but d'obtenir une reconnaissance des plus fidèles. La construction de trajectoires sur l'espace latent va nous permettre de voir l'évolution d'une émotion au cours du temps et donc de comparer plus fidèlement différentes émotions. De plus l'utilisation d'images de la base de données HCI Tagging nous permet de travailler avec des expressions spontanées et donc d'être au plus proche de la réalité, même si il aurait été préférable d'utiliser une base contenant directement des images de personnes handicapées, ce qui n'existe pas à l'heure actuelle malheureusement.

Dans le futur nous envisageons de créer des bases de données de vidéos avec des personnes handicapés pour pouvoir tester le système et apporter éventuellement d'ultérieures améliorations.

RÉFÉRENCES

- [1] J. Vitale, M-A. Williams, B. Johnston et G. Boccignone, *Affective Facial Expression Processing Via Simulation : A Probabilistic Model*, Biologically Inspired Cognitive Architectures, 2014.
- [2] P. Ekman et W. V. Friesen, *Constants across culture in the face and emotion*, Journal of Personality and Social Psychology, 1971.
- [3] P. Viola et M. Jones, *Robust Real-Time Face Detection*, 2001.
- [4] P. Ekman, *Facial Action Coding System (FACS) : Manual*, Palo Alto : Consulting Psychologists Press, 1978.
- [5] C. Cortes, V. Vapnik, *Support-vector networks*, Machine Learning, Vol. 20-3, pp. 273-297, 1995.
- [6] N. S. Altman, *An introduction to kernel and nearest-neighbor nonparametric regression*. The American Statistician, Vol. 46-3, pp. 175–185, 1992.
- [7] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar et I. Matthews, *The Extended Cohn-Kanade Dataset (CK+) : A complete expression dataset for action unit and emotion-specified expression*. Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010), San Francisco, USA, pp. 94-101. 2010.
- [8] M. Soleymani, J. Lichtenauer, T. Pun et M. Pantic., *A multimodal database for affect recognition and implicit tagging*, IEEE Transactions on Affective Computing, Vol. 3-1, pp. 42 - 55, 2012.
- [9] L. E. Baum et T. Petrie, *Statistical Inference for Probabilistic Functions of Finite State Markov Chains*. The Annals of Mathematical Statistics, Vol. 37-6, pp. 1554–1563. 1966.
- [10] M. Soleymani, S. Asghari-esfeden, M. Pantic et Y. Fu, *Continuous Emotion Detection using EEG Signals and Facial Expressions*, Proceedings of IEEE Int'l Conf. Multimedia and Expo (ICME'14), Juillet 2014.
- [11] Tingfan Wu, M.S Bartlett et J.R. Movellan, *Facial expression recognition using Gabor motion energy filters*, Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference, pp. 42-47, 13-18 Juin 2010
- [12] J. Whitehill, M.S. Bartlett et J.R. Movellan, *Automatic Facial Expression Recognition*, Social Emotions in Nature and Artifact, 2013.



Comptes rendus hebdomadaires

Compte rendu n°1 du 17/09/2015

Découverte de 2 grandes méthodes de description des mouvements du visage existent : le FACS (Facial Action Coding System) mis en place par P. Ekman et W. Friesen en 1978 et le FAPU (Facial Animation Parameter Units) introduit par la norme de codage vidéo MPEG-4.
Recherche sur les types d'acquisitions d'images en 2D ou en 3D avec le matériel nécessaires à chaque fois ainsi que les algorithmes disponibles.

Compte rendu n°2 du 24/09/2015

Recherche sur comment se décompose un bon système de reconnaissance facial d'émotions.
Décomposition en 4 parties (récupération du visage, normalisation, extraction des points clés, classification) et recherche plus poussée sur les 2 premières parties.

Compte rendu n°3 du 01/10/2015

Continuation des recherches sur les 2 dernières parties du système.
Recherche également sur les différentes bases de données 2D publics disponibles à l'utilisation.

Compte rendu n°4 du 08/10/2015

Recherche plus approfondies sur les filtres de Gabor. Présentation de mes recherches à Messieurs Conte et Slimane.
Commencement de l'écriture du rapport.

Compte rendu n°5 du 15/10/2015

Continuation des recherches sur les filtres de Gabor et leur fonctionnement. J'ai essayé de comprendre le fonctionnement des filtres et l'impact des différents paramètres. Grâce à un simulateur que j'ai trouvé en ligne ([[WWW8](#)]) et aux instructions associées ([[WWW7](#)]), j'ai pu constater l'effet qu'ont les différents paramètres sur le résultat final.

Compte rendu n°6 du 22/10/2015

Étude approfondi des bases de données MMI Mimicry et HCI Tagging.

Compte rendu n°7 du 05/11/2015

Documentation sur les caractéristiques pseudo-Haar et leur fonctionnement.
Recherche de techniques permettant de placer les points clés d'un visage sans FACS (ASM, AAM ...)

Compte rendu n°8 du 12/11/2015

Réunion avec Mrs Conte et Slimane : décision de l'arrêt de la phase état de l'art pour commencer le développement ; prise de décision sur les spécifications de notre système.
Rédaction du rapport.

Compte rendu n°9 du 19/11/2015

Documentation plus poussée sur ASM, récupération d'un programme Matlab d'analyse d'émotions réalisé par des collègues italiens à Mr Conte et tentative de le faire fonctionner sous Octave vu que nous ne possédons pas de licence Matlab.
Commencement de la prise en main de la librairie C++ OpenCV mais suite à un entretien avec Mr Conte, la décision a été prise de changer les spécifications de notre programme pour continuer le travail qui a déjà été réalisés par ses collègues italiens.

Compte rendu n°10 du 26/11/2015

Étude approfondi de l'article écrit par Vitale et al. ([29]) et rendez vous avec Mr Conte pour faire fonctionner le programme Matlab, presque fonctionnel au final.

Compte rendu n°11 du 03/12/2015

Fin de l'étude approfondi de l'article de Vitale et al.
Travail sur le programme Matlab pour le faire fonctionner à 100%.

Compte rendu n°12 du 10/12/2015

Rédaction du rapport.
Téléchargement de Matlab pour faire fonctionner le programme car il est impossible de le faire fonctionner avec Octave.
RDV avec Mr Conte pour vérifier la compréhension de l'article, des zones d'ombre persistent.

Compte rendu n°13 du 17/12/2015

Étude du programme Matlab fonctionnel et comparaison de l'article au programme pour trouver à quelle partie du programme correspond chaque partie de l'article dans le but de mieux le comprendre.

Compte rendu n°14 du Vacances de Noël

Fin de rédaction du rapport

Compte rendu n°15 du 07/01/2016

Création du PowerPoint pour la soutenance et répétitions.
Modification du programme Matlab pour qu'il prenne une image en entrée et l'affiche sur l'espace latent.
Création nouvelle base d'apprentissage plus adaptée, avec des images plus grandes.

Compte rendu n°16 du 14/01/2016

Soutenance de la partie Recherche
Continuation de la création de la base d'apprentissage en réduisant la taille des images (les images étaient trop grandes et faisaient crasher mon ordinateur)

Compte rendu n°17 du 21/01/2016

Rédaction d'un article scientifique sur les travaux réalisés lors de ce projet. L'article sera soumis à la conférence Handicap 2016. Rédaction de l'introduction à 90%, de l'état de l'art à 15% et de la méthode

proposée à 100%

Au niveau du programme Matlab :

- création nouvelle base d'apprentissage : 31 images de taille 50*81 pixels
- prise d'une image en entrée
- association d'une émotion à une image

Mais classification peu efficace pour l'instant.

Compte rendu n°18 du 28/01/2016

Rédaction de l'article scientifique.

Conférence Skype avec Giuseppe Boccignone pour nous aider dans nos travaux et nous donner de nouvelles pistes. Plusieurs améliorations ont été dégagées de cette conférence :

- changer la représentation des données, passer d'une représentation pixellique à une représentation vectorielle, ce qui nous permettrait de nous affranchir de l'algorithme PCa qui est très chronophage dès qu'une image est un peu grande
- changer la base d'apprentissage avec des images de grandes tailles
- représenter sur l'espace latent la trajectoire dans le temps d'une émotion plutôt que seulement la position d'une image
- utiliser l'algorithme FGPLVM (Faster GPLVM) plutôt que l'algorithme GPLVM utilisé précédemment
- migrer le programme de Matlab vers Python

Tentative de migration sous Python mais sans succès et donc prise en main du FGPLVM sous Matlab.

Compte rendu n°19 du 04/02/2016

Fin de la rédaction de l'article scientifique.

Création de la nouvelle base d'apprentissage (5 émotions * 15 frames) pour générer les trajectoires.

Compte rendu n°20 du 11/02/2016

Soumission de l'article scientifique à la conférence Handicap 2016 [\[WWW3\]](#).

Recherche d'une détection des features avec Matlab pour utiliser la position de ces features comme données pour construire l'espace latent

Création de l'espace latent avec toutes les trajectoires.

Compte rendu n°21 du Vacances de Février

Création de l'espace latent avec les coordonnées ou les distances des K plus proches voisins de chaque point (avec K=1,3,5,10).

Traçage des trajectoires correspondantes sur l'espace latent.

Compte rendu n°22 du 25/02/2016

Tentative d'ajouter sur l'espace latent des images passées en entrée représentant l'évolution d'une émotion : sans succès.

Compte rendu n°23 du 03/03/2016

Envoi d'un mail à Giuseppe Boccignone pour lui demander des conseils et de l'aide. Rédaction de la partie Développement du rapport

Compte rendu n°24 du 10/03/2016

Rédaction du rapport.

Réception de la réponse de G. Boccignone, tentative d'application de ce qu'il a dit mais sans succès donc renvoi d'un second mail

Compte rendu n°25 du 17/03/2016

Rédaction du rapport.

Réception de la seconde réponse de G. Boccignone et ce coup-ci ses conseils nous ont permis d'afficher

une trajectoire de test sur l'espace latent comme il faut.
Recherche d'un algorithme permettant de comparer des trajectoires basé sur leurs formes.
Création d'une base de test pour tester la précision de notre système.

Compte rendu n°26 du 24/03/2016

Fin de rédaction du rapport.
Préparation de la soutenance.

Webographie

- [WWW1] AFFECTIVA. *Demonstration de la reconnaissance d’emotion via la webcam par la societe Affectiva*. URL : <https://labs.affectiva.com/superbowl/affdexweb.html> (visité le 03/12/2015).

ANNOTATION: Exemple de reconnaissance d’emotion via la webcam de notre ordinateur realise par la societe Affectiva

- [WWW2] EMOTIENT. *Demonstration de la reconnaissance d’emotion via la webcam par la societe Emotient*. URL : <http://emotient.com/livecam-demo/> (visité le 03/12/2015).

ANNOTATION: Exemple de reconnaissance d’emotion via la webcam de notre ordinateur realise par la societe Affectiva

- [WWW3] IFRATH. *Site officiel de la conference Handicap 2016*. URL : <http://ifrath.fr/handicap2016/> (visité le 03/03/2016).

ANNOTATION: Site officiel de la conference Handicap 2016

- [WWW4] Neil LAWRENCE. *Programme complet (FGPLVM + librairies) realise par N.Lawrence*. URL : <https://github.com/SheffieldML/GPmat> (visité le 10/03/2016).

ANNOTATION: Github du programme Matlab realisant le FGPLVM avec toute les librairies annexes

- [WWW5] Neil LAWRENCE. *Github du GPLVM de Neil Lawrence*. 2005. URL : <https://github.com/lawrennd/gplvm> (visité le 03/03/2016).

ANNOTATION: Depot Github du programme Matlab de GPLVM utilise dans le programme d’origine que j’ai ensuite modifie

- [WWW6] Neil LAWRENCE. *Github du FGPLVM de Neil Lawrence*. 2007. URL : <https://github.com/lawrennd/fgplvm> (visité le 03/03/2016).

ANNOTATION: Depot Github du programme Matlab de FGPLVM que j’ai utilise pour mon programme

- [WWW7] N.PETKOV. *Instructions pour le simulateur de filtres de Gabor*. Sous la dir. d’University of GRONINGEN. URL : http://matlabserver.cs.rug.nl/edgedetectionweb/web/edgedetection_params.html.

ANNOTATION: Decrit chaque parametre du simulateur de filtres de Gabor et leurs impacts et explique comment l’utiliser au mieux

- [WWW8] N.PETKOV. *Simulateur de filtres de Gabor*. Sous la dir. d'University of GRONINGEN. URL : <http://matlabserver.cs.rug.nl/edgedetectionweb/web/index.html>.
 ANNOTATION: Site permettant de simuler le comportement des filtres de Gabor sur une image de notre choix. Tous les parametres des filtres peuvent etre modifies.
- [WWW9] *Site officiel de SAL*. URL : <http://emotion-research.net/toolbox/%20toolboxdatabase.2006-09-26.5667892524> (visité le 16/12/2015).
 ANNOTATION: Site decrivant la base de donnees de visage SAL
- [WWW10] Visage TECHNOLOGIES. *MPEG-4 Face and Body Animation (MPEG-4 FBA) : An overview*. URL : <http://www.visagetechnologies.com/uploads/2012/08/MPEG-4FBAOverview.pdf> (visité le 01/11/2015).
 ANNOTATION: Description du systeme de FAPU introduit par la norme de codage MPEG-4
- [WWW11] Florian TISSIER. *Github de mon PRD*. 2016. URL : <https://github.com/flo3693/Prog-PRD> (visité le 17/03/2016).
 ANNOTATION: GitHub de mon programme realise lors de ce PRD
- [WWW12] Florian TISSIER. *Github de mon rapport de PRD*. 2016. URL : <https://github.com/flo3693/Rapport-PRD>.
 ANNOTATION: GitHub de mon rapport realise lors de ce PRD

Bibliographie

- [1] Akshay ASTHANA, Stefanos ZAFEIRIOU, Shiyang CHENG et Maja PANTIC. « Robust Discriminative Response Map Fitting with Constrained Local Models ». In : *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference* (2013).
ANNOTATION: Article présentant les travaux qui ont permis a la realisation du programme Matlab que j'utilise pour extraire 66 features d'un visage sur une image
- [2] Philippe BESSE, Brendan GUILLOUET, Jean-Michel LOUBES et Francois ROYER. « Classification non Supervisee de Trajectoires ». In : ().
ANNOTATION: Presentation d'une adaptation de la distance OWD. Celle-ci se base sur la distance physique entre deux trajectoires, la forme des trajectoires (orientation, longueur) et "l'independance temporelle".
- [3] S. BILAKHIA, S. PETRIDIS, A. NIJHOLT et M. PANTIC. « The MAHNOB Mimicry Database - a database of naturalistic human interactions ». In : *Pattern Recognition Letters, vol. 66, pp. 52-61* (2015).
ANNOTATION: Article decrivant la base de donnees de visage MHI Mimicry
- [4] D. COSKER, E. KRUMHUBER et A. HILTON. « A FACS valid 3D dynamic action unit database with applications to 3D dynamic morphable facial modeling ». In : *IEEE InternationalConference on Computer Vision (ICCV)* (2011).
ANNOTATION: Article decrivant la base de donnees de visage D3DFACS
- [5] Charles DARWIN. *L'Expression des émotions chez l'homme et les animaux*. 1872.
ANNOTATION: Ce livre de Charles DARWIN decrit les similitudes entre les hommes et les animaux en matiere d'émotions
- [6] Paul EKMAN. « Facial Action Coding System (FACS) : Manual ». In : *Palo Alto : Consulting Psychologists Press* (1978).
ANNOTATION: Description et manuel d'utilisation du systeme FACS cree par Paul EKMAN
- [7] Paul EKMAN et Wallace V. FRIESEN. « Constants across culture in the face and emotion. » In : *Journal of Personality and Social Psychology* (1971).

- ANNOTATION: Article écrit par Ekman après son voyage en Papouasie-Nouvelle-Guinée et décrivant pour la première fois les 7 émotions universelles (neutre, joie, peur, colère, dégoût, tristesse et surprise)
- [8] HARWOOD, OJALA, PIETKINEN, KELMAN et DAVIS. « Texture classification by center-symmetric auto-correlation, using Kullback discrimination of distributions ». In : (1993).
ANNOTATION: Article introduisant pour la première fois l'utilisation des motifs locaux binaires pour mesurer le contraste d'une image
- [9] I.T. JOLLIFE. *Principal Component Analysis (Second Edition)*. 2002.
ANNOTATION: Livre expliquant le fonctionnement du Principal Component Analysis
- [10] Takeo KANADE, Jeffrey F. COHN et Yingli TIAN. « Comprehensive Database for Facial Expression Analysis ». In : *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition* (2000).
ANNOTATION: Article décrivant la base de données de visage Cohn-Kanade (CK)
- [11] Neil LAWRENCE. « Gaussian process latent variable models for visualisation of high dimensional data ». In : (2004).
ANNOTATION: Papier correspondant au programme Matlab de l'algorithme GPLVM
- [12] Jeroen LICHTENAUER et Mohammad SOLEYMANI. « MAHNOB-HCI-TAGGING DATABASE ». In : (2012).
ANNOTATION: Manuel d'utilisation de la base de données de visage HCI Tagging
- [13] Jeroen LICHTENAUER, Michel VALSTAR, Xiaofan SUN, Anton NIJHOLT et Maja PANTIC. « MAHNOB HMI IBUG MIMICRY DATABASE (MHI-MIMICRY) ». In : (2011).
ANNOTATION: Manuel d'utilisation de la base de données de visage MHI Mimicry
- [14] Rainer LIENHART et Jochen MAYDT. « An Extended Set of Haar-like Features for Rapid Object Detection ». In : *IEEE ICIP* (2002).
ANNOTATION: Extensions des composantes de Haar, ajout de composantes orientées à 45 degrés permettant d'augmenter les performances de la méthode des composantes pseudo-Haar dans la détection de contours ou d'objets
- [15] Bin LIN et Jianwen SU. « Shapes Based Trajectory Queries for Moving Objects ». In : *Proceedings of the 13th annual ACM international workshop on Geographic information systems*, (2005).
ANNOTATION: Description d'une distance appelée OWD, One-Way-Distance, qui se base uniquement sur la forme des trajectoires.
- [16] Patrick LUCEY, Jeffrey F. COHN, Takeo KANADE, Jason SARAGIH, Zara AMBADAR et Iain MATTHEWS. « The Extended Cohn-Kanade Dataset (CK+) : A complete dataset for action unit and emotion-specified expression ». In : (2010).
ANNOTATION: Article décrivant la 2ème version de la base de données de visage Cohn-Kanade (CK+)
- [17] Michael J. LYONS, Shigeru AKAMATSU, Miyuki KAMACHI et Jiro GYOBA. « Coding Facial Expressions with Gabor Wavelets ». In : *Proceedings, Third IEEE International Conference on Automatic Face and Gesture Recognition* (1998).

- ANNOTATION: Article décrivant la base de données de visage JAFFE
- [18] A. MORENO et A. SANCHEZ. « Gavabdb : a 3D face database ». In : (2004).
ANNOTATION: Article décrivant la base de données de visage Gavabdb
- [19] OJALA, PIETIKINEN et HARWOOD. « A comparative study of texture measures with classification based on feature distributions ». In : *Pattern Recognition* 29 (1996).
ANNOTATION: Article expliquant l'utilisation des motifs binaires locaux dans l'analyse de texture d'une image.
- [20] Maja PANTIC, Michel VALSTAR, Ron RADEMAKER et Ludo MAAT. « Web-based database for facial expression analysis ». In : (2005).
ANNOTATION: Article décrivant la base de données de visage MMI
- [21] Constantine P. PAPAGEORGIOU, Michael OREN et Tomaso POGGIO. « A General Framework for Object Detection ». In : *International Conference on Computer Vision* (1998).
ANNOTATION: Travaux sur lesquels Viola et Jones se sont basés pour réaliser leur détecteur de visage et décrivant des caractéristiques conçues à partir des ondelettes de Haar
- [22] Georgia SANDBACH, Stefanos ZAFEIRIOU, Maja PANTIC et Lijun YIN. « Static and dynamic 3D facial expression recognition : A comprehensive survey ». In : *IMAGE AND VISION COMPUTING* (2012).
ANNOTATION: Etat de l'art sur les techniques et matériels utilisés dans la reconnaissance d'expressions en 3D. Description des différentes techniques utilisées par différents matériels (ex : Kinect) pour récupérer des visages en 3D et les analyser ensuite
- [23] Arman SAVRAN, Nese ALYUZ, Hamdi DIBEKLIOGLU, Oya CELIKTUTAN, Berk GOKBERK, Bulent SANKUR et Lale AKARUN. « Bosphorus database for 3D face analysis ». In : *The First COST 2101 Workshop on Biometrics and Identity Management* (9 mai 2008).
ANNOTATION: Article décrivant la base de données de visage Bosphorus
- [24] Mohammad SOLEYMANI, Jeroen LICHTENAUER, Thierry PUN et Maja PANTIC. « A Multimodal Database for Affect Recognition and Implicit Tagging ». In : *IEEE Transactions on Affective Computing*. 3 : pp. 42 - 55, Issue 1 (2012).
ANNOTATION: Article décrivant la base de données de visage HCI Tagging
- [25] Anuj SRIVASTAVA, Eric KLASSEN, Shantanu H. JOSHI et Ian H. JERMYN. « Shape analysis of elastic curves in euclidean spaces ». In : *Issue No.07 - July (2011 vol.33)* (2011).
ANNOTATION: Présentation d'une distance entre 2 trajectoires qui permet de détecter très efficacement des formes similaires, mais impose à celles-ci d'être de même longueur et est indépendante de la distance physique entre les deux trajectoires.
- [26] Giota STRATOU, Abhijeet GHOSH, Paul DEBEVEC et Louis-Philippe MORENCY. « Effect of Illumination on Automatic Expression Recognition : A Novel 3D Relightable Facial Database ». In : *9th International Conference on Automatic Face and Gesture Recognition* (2011).
ANNOTATION: Article décrivant la base de données de visage ICT-3DRFE
- [27] Michel F. VALSTAR et Maja PANTIC. « Induced Disgust, Happiness and Surprise : an Addition to the MMI Facial Expression Database ». In : *Proceedings of Int'l Conf. Language Resources and Evaluation, Workshop on EMOTION* (2010).
ANNOTATION: Article décrivant la base de données de visage MMI

- [28] Paul VIOLA et Mickael JONES. « Robust Real-Time Face Detection ». In : (2001).
 ANNOTATION: Cet article decrit l'algorithme cree par Viola et Jones. Ce fut l'un des premiers algorithmes a detecter efficacement en temps reel des visages dans une image et il est maintenant utilise dans un grand nombre de detecteurs.
- [29] Jonathan VITALE, Mary-Anne WILLIAMS, Benjamin JOHNSTON et Giuseppe BOCCIGNONE. « Affective facial expression processing via simulation : A probabilistic model ». In : *Biologically Inspired Cognitive Architectures* (2014).
 ANNOTATION: Article decrivant une nouvelle approche permettant d'analyser des emotions via une methode probabilistique
- [30] Jacob WHITEHILL, Marian Stewart BARTLETT et Javier R. MOVELLAN. « Automatic Facial Expression Recognition ». In : (2014).
 ANNOTATION: Article ecrit par les fondateurs de l'entreprise Emotient et faisant un etat de l'art des differentes parties d'un systeme de reconnaissances d'emoions ainsi que de plusieurs techniques et bases de donnees
- [31] Lijun YIN, Xiaochen CHEN, Yi SUN, Tony WORM et Michael REALE. « A High-Resolution 3D Dynamic Facial Expression Database ». In : *The 8th International Conference on Automatic Face and Gesture Recognition* (19 sept. 2008).
 ANNOTATION: Article decrivant la base de donnees de visage BU4DFE
- [32] Lijun YIN, Xiaozhou WEI, Yi SUN, Jun WANG et Matthew J. ROSATO. « A 3D Facial Expression Database For Facial Behavior Research ». In : *The 7th International Conference on Automatic Face and Gesture Recognition* (12 avr. 2006).
 ANNOTATION: Article decrivant la base de donnees de visage BU3DFE

Application d'aide à l'interaction homme/machine pour les personnes handicapées

Florian Tissier

Encadrement : Mohamed Slimane et Donatello Conte

Objectif

Repérer le visage d'une personne handicapée grâce à une caméra puis analyser son émotion actuelle.
En fonction de l'émotion récupérée, une action spécifique sera réalisée (ex: changement de couleur de la lumière).



Première approche

Utilisation de normes de description des mouvements faciaux, telle que la norme **FACS**, qui permet de définir 7 émotions (neutre, joie, tristesse, peur, surprise, colère et dégoût) en terme d'**Action Units** représentant des mouvements faciaux.
Approche de moins en moins adaptée aux problèmes d'aujourd'hui.

AU1 Inner brow raiser	AU2 Outer brow raiser	AU4 Brow Lowerer	AU5 Upper lid raiser	AU6 Cheek raiser
AU7 Lid tighten	AU9 Nose wrinkle	AU12 Lip corner puller	AU15 Lip corner depressor	AU17 Chin raiser
AU23 Lip tighten	AU24 Lip presser	AU25 Lips part	AU27 Mouth stretch	

Nouvelle approche

Via un modèle probabiliste, il est possible de représenter les émotions sur un espace fini en 2 dimensions
En fonction de la position d'un point sur cet espace, on peut donc en déduire l'émotion associée.



Application d'aide à l'interaction homme/machine pour les personnes handicapées

Résumé

Dans le cadre de ce projet, j'ai pour mission de réaliser une application, destinée aux personnes handicapées, qui permettra de reconnaître leurs émotions et de réaliser certaines actions en conséquences (ex: changement de couleur de la lumière).

Dans la partie Recherche de ce projet, j'ai réalisé un état de l'art sur les différentes techniques permettant de reconnaître des émotions (filtres de Gabor, composantes pseudo-Haar,...) et sur comment construire un système de reconnaissance faciale d'expressions fiable et performant. Je me suis également documenté sur une nouvelle approche permettant de représenter des émotions sur un espace fini via un modèle probabiliste.

Mots-clés

détection, émotion, filtres de Gabor, composantes pseudo-Haar, reconnaissance facial d'expression, FACS, bases de données de visages, modèle probabiliste

Abstract

During this project, I have the mission to realize an application, for disabled people, which will recognize their emotions and do specific actions in consequences (e.g: change the color of the lights).

In the Research part of this project, I carried out a state of the art on the different techniques allowing emotion recognition (Gabor filters, Haar-like features,...) and on how to build a reliable and efficient facial expression recognition system. I also documented myself about a new way that allows us to represent emotion on a finite space via a probabilistic model.

Keywords

detection, emotion, Gabor filters, Haar-like features, facial expression recognition, FACS, faces databases, probabilistic model

Tuteurs académiques

Mohamed SLIMANE
Donatello CONTE

Étudiants

Florian TISSIER (DI5)