Master Informatique

« Suivi de visage et détection de saillances »

Rapport
en vue de la validation de
l'UE Initiation à la recherche
2018 - 2019

Étudiants:
Alaa Eddine BENKARRAD
Abdelaziz CHERIFI
Walid HAFIANE

Encadrants :
Monsieur Amine BOUMAZA
Monsieur Alain DUTECH
Monsieur Yann BONIFACE

Soutenu le 20/05/2019

Décharge de responsabilités

L'Université de Lorraine n'entend donner ni approbation ni improbation aux opinions émises dans ce rapport, ces opinions devant être considérées comme propres à leur auteur

Remerciements

En tout premier lieu, nous souhaitons remercier l'ensemble des enseignants de l'université de Lorraine ayant contribué à notre formation.

Nous sommes également très reconnaissants envers nos encadrants, Monsieur Amine Boumaza, Monsieur Alain Dutech et Monsieur Yann Boniface, pour le temps et la confiance qu'ils nous ont accordés ainsi que pour leurs précieux conseils, leur constante bonne humeur, et leurs moments très agréables passés avec eux.

Bien sûr, nous n'oublions pas de remercier nos familles et nos amis respectifs pour leur présence et leur soutien inconditionnel.

Table des matières

Table des Figures i
Table des Symboles ii
Introduction générale 1
Chapitre 1: Présentation du projet 2
Introduction . . . 3
1.1 Projet PsyPhINe . . . 3
1.1.1 Idée de base du projet . . . 3
1.1.2 Présentation du projet . . . 3
1.1.3 Projet similaire - Pinokio . . . 4
1.2 Étude de la lampe . . . 4
1.2.1 Description de la lampe . . . 5
1.2.2 Travaux réalisés . . . 6
1.3 Problématique . . . 7
Conclusion . . . 8
Chapitre 2: État de l'art 9
Introduction . . . 10
2.1 Méthodes de détection de visage . . . 10
2.1.1 Détection avec la méthode de Viola et Jones . . . 10
2.1.2 Détection avec l'Histogrammes de gradients orientées (HOG) . . . 11
2.1.3 Autre méthodes de détection . . . 13
2.2 Methode d'extraction des points de saillances [10] . . . 13
2.3 Clustering . . . 14
2.3.1 K-means [16] . . . 15
2.3.2 Carte auto-organisatrice (SOM) . . . 16
Conclusion . . . 17
Chapitre 3: Analyse et réalisation 18
Introduction . . . 19
3.1 Appoche globale . . . 19
3.2 Phase de préparation . . . 20
3.2.1 Prétraitement . . . 20
3.2.2 Détection du visage . . . 20
3.2.3 Éxtraction des points de saillances . . . 21
3.3 Phase d'extraction des VCC . . . 21
3.3.1 Éxtraction et encodage des caractéristiques . . . 22
3.3.2 Vecteurs caractéristiques des changements d'expressions . . . 24

3.4 Phase de décision 25
3.4.1 K-Means 26
3.4.2 Carte auto-organisatrice dynamique 26
Conclusion 28
Chapitre 4: Implémentation et Tests **29**
Introduction 30
4.1 Implémentation 30
4.1.1 Outils utilisés 30
4.1.2 Interfaces 30
4.2 Tests 33
4.2.1 K-Means 33
4.2.2 DSOM 36
Conclusion 38
Conclusion générale **39**
Bibliographie **40**
Annexe **42**

Table des figures

1.1 Expreriences du projet Psyphine [3] 4
1.2 Pinokio la lampe de bureau rebotisée 5
1.3 La lampe de Psyphine [3] 5
1.4 Les moteurs de la lampe[3] 6

2.1 Caractéristiques de Haar 10
2.2 construction du gradient orienté 12
2.3 Processus de la construction de l'histogramme des gradients 12
2.4 L'ensemble des 68 points détectés par Dlib pré-entraîné 14
2.5 Clustering 15
2.6 K-means clustering 15
2.7 Réduction de la dimensionnalité dans la SOM 17

3.1 Structure globale du systèùme 19
3.2 Prétraitement 20
3.3 Détection de visage 20
3.4 Éxtraction des points de saillances 21
3.5 Limites de la méthode d'extraction des points de saillances. 22
3.6 Éxtraction des caractéristiques 22
3.7 Exemple du problème de représentation des distances sans normalisation . 23
3.8 Calcul des vecteurs caractèristiques des changements 25
3.9 Exemples VCC 25
3.10 Structure de la DSOM utilisée 27

4.1 Fenetre du flux vidéo 31
4.2 Fenetre du plan 2D du modèle 31
4.3 Fenetre des dimensions des clusters 32
4.4 Fenetre des entrées-sorties 32
4.5 Test 1 - cas général 33
4.6 Test 2 - taille de la base - plan 2D 34
4.7 Test 2 - taille de la base - dimensions des clusters 35
4.8 Test 2 - taille de la base - dimensions des clusters 36
4.9 Test avec le paramètre Init_Methode 36
4.10 Test avec le paramètre d'élacticité 37
4.11 test avec le paramèrte n 38

Liste des symboles

BMU Best Matching Unit

ERT Ensemble of Regression Trees

OpenCV Open Computer Vision

PsyPhINe Psychologie, Philosophie, Informatique et Neuroscience

RNA réseau de neurones artificiels

ROI Region Of Interest

VCC Vecteurs Caracteristiques des Changements

Introduction générale

L'expansion rapide de l'industrie de l'intelligence artificielle et de la robotique est un facteur important qui influence et transforme divers aspects de notre vie quotidienne. Aujourd'hui, les machines intelligentes sont omniprésentes et elles sont devenues indispensables. Nous sommes de plus en plus souvent en relation avec des robots et des machines, que ce soit à des fins pratiques (thérapeutiques, professionnelles, scientifiques, quotidien ménager) ou ludiques. Par conséquent, ces relations, entre l'homme et la machine, entrainent de nombreuses questions et plus particulièrement sur l'**attribution d'intentions**, d'**intelligence** voire de **conscience** à un objet robotisé non humanoïde. Autrement dit, l'aspect humanoïde de la machine est-il nécessaire pour que nous soyons enclins à lui prêter des états mentaux ? Afin d'étudier ces interactions et dans le but de répondre à ces questions liées au sujet, le projet Psyphine a été mis en place.

Le projet Psyphine regroupe plusieurs disciplines, à savoir, la psychologie, la philosophie, l'informatique et la neuroscience. Il s'interroge sur les **interactions** homme-robot et cherche à répondre à ces interrogations à travers plusieurs expériences. Pour ce faire, il utilise un **prototype robotisé** qui se présente sous la forme d'une lampe « La lampe Psyphine ». Cette dernière est un modèle unique qui a été construit et développé par le groupe.

Puisque le but du projet est d'étudier les **réactions** et les **comportements** des gens à travers des expériences dans laquelle les personnes seront en interaction non verbale avec la lampe rebotisé, notre objectif sera d'affecter à la lampe, la **capacité d'interagir** (de se comporter) conformément au **changement d'expressions du visage** de la personne dans l'expérience. La question à laquelle nous essayons de répondre est comment concevons-nous un système qui permet à la lampe d'exécuter des comportements suivant les différentes réactions des gens perçues par le biais de sa webcam ?

Pour répondre à cet objectif, nous avons décidé d'organiser notre rapport en quatre chapitres, tels que :

- Le premier chapitre sera une présentation du projet, du prototype utilisé et des différents travaux réalisés. A la fin de ce chapitre nous allons détailler les différents aspects de notre problématique.
- Le second chapitre regroupe plusieurs notions, méthodes et techniques existantes dans l'état de l'art et spécifiquement celles que nous utiliserons dans notre méthode.
- Nous abordons, dans le troisième chapitre, les détails de notre solution dans laquelle nous expliquerons nos différents choix conceptuels et théoriques.
- Le dernier chapitre sera consacré à l'implémentation et aux divers tests.

Chapitre 1

Présentation du projet

« ...la robotique et d'autres combinaisons rendront le monde assez fantastique comparé à aujourd'hui. ». B. Gates.

Sommaire

Introduction **3**
1.1 Projet PsyPhINe **3**
1.1.1 Idée de base du projet 3
1.1.2 Présentation du projet 3
1.1.3 Projet similaire - Pinokio 4
1.2 Étude de la lampe **4**
1.2.1 Description de la lampe 5
1.2.2 Travaux réalisés 6
1.3 Problématique **7**
Conclusion **8**

Introduction

Premièrement et avant d'expliquer notre méthode, il est nécessaire d'introduire le contexte du projet et les moyens logiciels et matériels existants. Au cours de ce chapitre, nous présentons le projet Psyphine et d'autres projets similaires. Nous décrivons la structure physique de la lampe robotisée, ses caractéristiques, ainsi que les travaux réalisés dans le cadre du même projet. Nous clôturons ce chapitre par l'établissement de notre problématique sur laquelle nous construisons notre solution.

1.1 Projet PsyPhINe

1.1.1 Idée de base du projet

L'idée fondatrice du projet Psyphine est partie de l'interaction entre l'humaine et le robot de telle sorte que le robot peut se comporter de la même façon que l'humain en laissant le robot apprendre à travers son interaction avec l'humain et son environnement. Le but donc est que le robot tente de percevoir les diffèrents comportements de l'être-humain avec lequel il interagit à partir d'indices multiples et d'expressions variées. Ainsi, cette perception s'étend aux interactions non verbales c'est-à-dire aux interactions non seulement avec des êtres humains capable de parler ou entre les humains non verbalement, mais aussi avec des choses non humaines (animaux, objets). Pour cela il va falloir attribuer des comportements d'intentions, une certaine forme de volonté, et même des émotions dans certains cas à ce robot qui est programmé par l'homme. La diversité des différentes tâches du projet Psyphine a met en relation de nombreux chercheurs de différentes disciplines, tels que des psychologues, philosophes, roboticiens, cogniticiens, sociologues, et neuroscientifiques d'où le nom « PsyPhINe » qui est l'abréviation de « Psychologie, Philosophie, Informatique et neuroscience » [17].

1.1.2 Présentation du projet

Le projet PsyPhINe, d'après [17], est lancé officiellement le premier avril 2016 par des chercheurs de l'université de lorraine, il vise à confronter et articuler les apports de différentes disciplines à la question de l'attribution d'intentionnalité, d'intelligence, de cognition voire d'émotions, à des entités naturelles ou des dispositifs artificiels. Il s'agit d'appréhender les questions naturellement posées par l'interaction homme/robot, à savoir celles liées à l'interprétation du comportement du robot jusqu'à la confiance qui peut lui être ou non accordée. Le projet vise notamment à explorer la gradation des attributions d'intelligence ou d'intentionnalité, quand on passe par exemple d'une mouche à un chat, en faisant l'hypothèse que l'intersubjectivité ainsi que notre tendance naturelle à l'anthropomorphisme jouent des rôles centraux : on projette dans l'autre énormément de notre propre cognition. La perspective générale du groupe de recherche pluridisciplinaire est d'aboutir à la définition d'un test de Turing non verbal qui permette d'appréhender l'intelligence artificielle en évitant certains écueils de la formulation d'origine dudit test. Dans le cadre du présent projet, le groupe PsyPhINe conduira des expérimentations à large échelle à l'aide d'un dispositif d'interaction basé sur un prototype de lampe robo-

tisée. La mise en place des protocoles expérimentaux, des questionnaires et des analyses de vidéos, doublées d'analyses « profanes », constituent le support et le lieu des confrontations et réflexions interdisciplinaires sur la cognition. Les résultats attendus en sont principalement une clarification d'un champ conceptuel riche et complexe, l'ébauche d'un langage commun et la production d'un référentiel des comportements donnant lieu à des interprétations intentionnelles. Comme présente la figure 1.1 suivante, la valorisation des travaux sera assurée par l'organisation d'ateliers interdisciplinaires et par l'édition d'un ouvrage collectif [17].

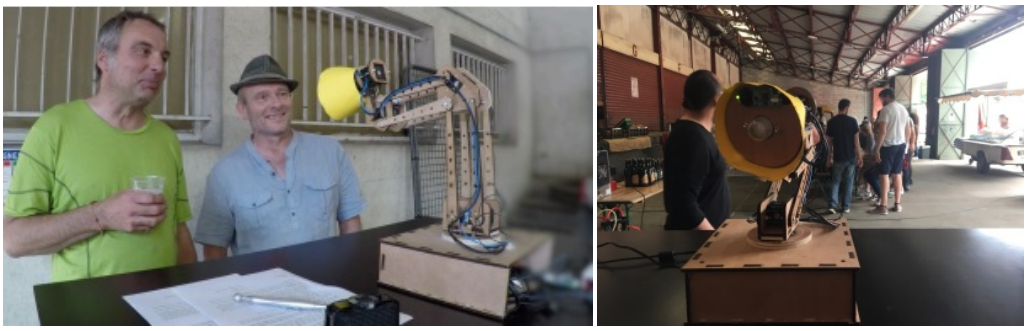


FIGURE 1.1 – Expreriences du projet Psyphine [3]

1.1.3 Projet similaire - Pinokio

Le nom Pinokio est venu de la lampe Luxo Jr, l'adorable petite lampe qui apparait dans le logo Disney Pixar, illustre comment les animateurs peuvent donner vie à des objets inanimés banals. Le trio de l'Université Victoria de Wellington, en Nouvelle-Zélande se sont inspiré de la lampe Luxo pour fabriquer leurs lampe robotisée grâce à une combinaison de robotique facilement accessible et de technologie de fabrication automatisée, combinée à des logiciels libres. Le trio qui comporte un programmeur Shanshan Zhou, un ingénieur mécanique Adam Ben-Dor qui travaillait sur les détails mécaniques de la lampe et un designer Joss Dogget qui s'est chargé du design et tout l'esthétique de la forme de la lampe, ce trio a donc décidé en 2012 d'embellir une lampe de bureau avec un peu de personnalité appelée Pinokio (présente dans la figure 1.2).

La lampe Pinokio est composée dans sa structure générale de son corps d'une tête, d'un bras pliant et capable d'étirer et de rétrécir, une base sur laquelle le bras tourne et 6 servo-moteurs. Elle est également dite une lampe active (qui se déplace), ses actions sont principalement pilotées par Arduino et le logiciel de traitement d'image OpenCV, qui recherche le visage dans les images à partir de sa webcam. Lorsqu'il trouve un visage, il tente de le suivre comme s'il essayait de maintenir un contact visuel.

1.2 Étude de la lampe

Afin de répondre aux questions liées aux interactions homme/machine et l'attribution de conscience dans le cadre du projet, le groupe Psyphine a construit et a développé

FIGURE 1.2 – Pinokio la lampe de bureau rebotisée

un prototype robotisé qui se présente sous la forme d'une lampe dite « La lampe de Psyphine ». La structure et le modèle de fonction de cette lampe est inspiré de la lampe Pinokio décrite précédement.

1.2.1 Description de la lampe

Lampe de Psyphine, comme la montre la figure 1.3, est construite de contreplaqué léger dont les pièces ont été découpées au laser. Ces dernières sont assemblées autour de cinq moteurs. L'abat-jour a été découpé dans un carton jaune et contient une ampoule centrale et une petite caméra située juste au dessus de l'ampoule [3].

FIGURE 1.3 – La lampe de Psyphine [3]

1.2.1.1 Moteurs

La lampe est équipée de cinq moteurs de la marque Robotis Dynamixel en deux modèles, AX-12 et AX-18. Ces moteurs permettent cinq articulations (illustré par la

figure 1.4) :

- **Moteur 1 :** la base sur un axe vertical
- **Moteur 2 :** le premier bras pour avancer ou éloigner la lampe
- **Moteur 3 :** le deuxième bras pour monter ou descendre l'abat-jour
- **Moteur 4 :** pour incliner l'abat-jour vers le bas ou vers le haut
- **Moteur 5 :** pour tourner l'abat-jour à droite ou à gauche

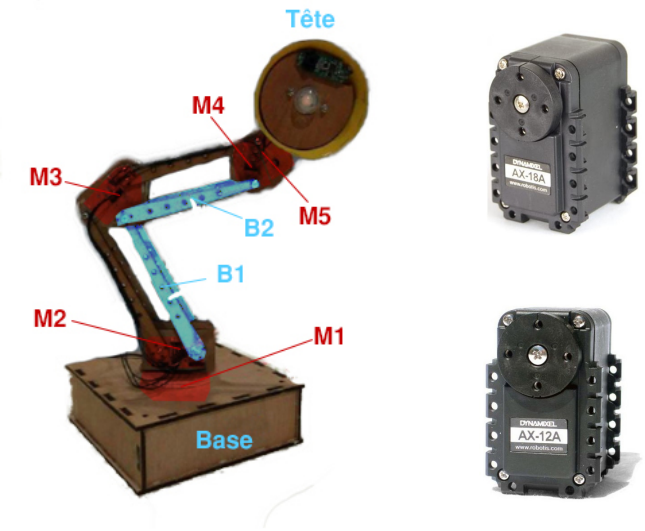


FIGURE 1.4 – Les moteurs de la lampe[3]

1.2.1.2 Système

La lampe comporte un système qui permet de faire fonctionner les moteurs. Cependant, la totalité des traitements couteux sont exécutés sur un ordinateur auquel les moteurs sont raccordés sur un port USB ou en wifi par l'intermédiaire d'un Raspberry Pi. Cet ordinateur s'occupe de commander la lampe rebotisée dont chaque mouvement de cette dernière consiste en une séquence de positions pour chacun des moteurs.

1.2.2 Travaux réalisés

Dans le cadre du projet, une panoplie de travaux ont été réalisés par des étudiants. Nous en citons :

Le travail [7] réalisé par D. SAMY et al. sur plusieurs axes. En premier, ils ont mis en place un système pour contrôler les moteurs de la lampe à distance. Ensuite, ils ont réalisé un suivi de visage dont la détection est basée sur la méthode Viola et Jones implémentée dans la bibliothèque OpenCV. Finalement, leur travail a révélé plusieurs problèmes comme la fluidité de mouvement du suivi, qui a été réglé en modifiant le temps d'exécution, ainsi que le problème persistant de réchauffement des moteurs.

S. BARBILLON et al. ont construit, dans [4], une nouvelle architecture à l'aide de la bibliothèque Poppy pour faciliter la définition des nouveaux mouvements (interface de

haut niveaux). Ils ont adapté la même méthode que celle du travail précédent pour le suivi de visage. Ainsi, ils ont consacré une grande partie de leur travail à la cinématique inversée pour les mouvements de la lampe.

Un autre travail est celui de A. Eva et G. Augustin [6]. Ce travail s'est concentré sur le suivi du visage et la détection des changements d'expressions. La méthode de Viola et Jones a été appliquée pour la détection du visage avec des amélioration au niveau de la détection de la bouche. Cette phase a été suivi par l'application de la régression linéaire pour détecter l'ouverture de la bouche.

R. BUISINE et O. RIOU ont exploré, dans leur rapport [18], un axe de travail différents des autres travaux. Il s'agit de la classification des émotions. Initialement, ils ont utilisé la bibliothèque (dlib) pour détecter les visage et extraire des points de saillances. Ensuite, ils ont classifé les émotions en trois classes (joie, colère et surprise) à l'aide du model bayésien. Ce dernier a été validé par la validation croisée à k-plis avec une précision de 82%, 97% et 31% pour la surprise, la joie et la colère respectivement.

Le projet le plus récent est celui de J. NOWAK et S. RIMLINGER [9]. Il traite la détection du visage et l'extraction des points de saillances dont une grande partie était consacrée à l'amélioration des performances et l'évaluation de la détection, de la bibliothèque dlib, sur les différentes contraintes (l'intensité de lumière, l'angle de rotation...). Ils ont clôturé leur réalisation par la mise en place d'une méthode de détection d'ouverture de la bouche.

1.3 Problématique

Plusieurs sujets ont été abordé dans les travaux précédents du projet Psyphine. Cependant, la diversité et la richesse des disciplines du projet Psyphine ont rendu difficile de travailler sur tous les axes du projet. En effet, une panoplie des thématiques peuvent être traitées dans le cadre du projet Psyphine. Celles proposées sont, la détection des objets, la détection des émotions, la reconnaissance du visage, le problème du réchauffement des moteurs, le raffinement du mouvement de la lampe rebotisée, etc. Cependant, nous voulions travailler sur l'ensemble des trois premières thématiques mentionnées. Néanmoins, puisque le but du projet est d'étudier les *réactions des gens*, la détection des objets ne peut pas apporter une grande valeur au but du projet. De plus, les études sont effectuées à travers des expériences d'une *durée maximale de 10 minutes* dans les différents salons et évènement scientifiques, ce qui implique que les participants, des différentes expériences, seront en interaction non verbale avec la lampe rebotisée qu'*une seul fois*. Donc, la reconnaissance du visage ne sera pas très utile à ces études. Par conséquent, la détection des émotions semble la plus intéressante des thématiques. Mais, le but du projet, est de permettre à la lampe de réagir aux différentes expressions, des participants, *sans avoir besoin de les interpréter*.

Tout cela nous à permet d'en sortir avec la problématique que nous allons traiter. Notre contribution est donc d'affecter à la lampe la capacité d'interagir (de se comporter) conformément au changement d'expressions du visage de la personne en face. En effet, la problématique principale est comment peut-on concevoir un système qui permet à la lampe d'exécuter des comportements en fonction des différentes réactions des gens

aperçues par le biais de sa webcam ?

Nous pouvons diviser cette problématique en trois sous problématiques dont la première sera le choix de la méthode de détection du visage et d'extraction des points de saillances. La deuxième, c'est le choix de la représentation des comportements des gens et la dernière sera le regroupement (clustering) de ces comportements afin de décider le comportement approprié.

Conclusion

Ce chapitre a servi comme introduction au projet et à notre problématique sur laquelle notre contribution sera basée. Dans le chapitre suivant nous plongeons dans les notions techniques des méthodes existantes dans l'état de l'art.

Chapitre 2

État de l'art

« ...ce que nous voulons, c'est une machine qui peut apprendre de l'expérience ». A. Turing.

Sommaire

Introduction 10
2.1 Méthodes de détection de visage 10
2.1.1 Détection avec la méthode de Viola et Jones 10
2.1.2 Détection avec l'Histogrammes de gradients orientées (HOG) . . 11
2.1.3 Autre méthodes de détection 13
2.2 Methode d'extraction des points de saillances [10] 13
2.3 Clustering 14
2.3.1 K-means [16] 15
2.3.2 Carte auto-organisatrice (SOM) 16
Conclusion 17

Introduction

Dans le but de faciliter la compréhension de notre solution dans le chapitre prochain, nous introduisons dans ce chapitre les differentes notions de base employé, ainsi, nous présentons les méthodes existantes pour les differents phases, de la détection du visage jusqu'à la classification non supervisé.

2.1 Méthodes de détection de visage

La détection d'objet est d'identifier quels vobjets se trouvent à l'intérieur d'une image et où ils se trouvent. C'est un type d'application classé dans la technologie de vision par ordinateur, et aussi le processus au cours duquel des algorithmes sont développés et formés pour localiser correctement les visages ou les objets. Celles-ci peuvent être en tems réel à partir d'une caméra vidéo. Afin de détecter la position des visages dans les images de sorte à obtenir une région d'intérêt sur laquelle l'extraction des vecteurs de caractéristiques pourra être accomplie il y a bien de nombreuses méthodes, dans ce qui suit nous allons présenter les méthodes les plus utilisées dans la détection.

2.1.1 Détection avec la méthode de Viola et Jones

Les cascades de Haar est une méthode utilisée pour la détection d'objet de classificateurs en cascade basés sur les fonctionnalités Haar, c'est une méthode efficace de détection d'objets proposée par Paul Viola et Michel Jones en 2001 [20]. Il s'agit d'une approche basée sur l'apprentissage automatique.

La fonction cascade est formée à partir de nombreuses images positives et négatives. Il est par la suite utilisé pour détecter des objets dans d'autres images. Cette méthode travaille avec la détection de visage. Initialement, l'algorithme nécessite beaucoup d'images positives (image de visage) et d'images négatives (images sans visages) pour former le classifieur. Ensuite, extraire les caractéristiques, et pour cela, les fonctionnalités de Haar présentées dans la figure ci-dessous sont utilisées. Chaque caractéristique est une valeur unique obtenue par la soustraction de la somme des pixels sous le rectangle blanc de la somme des pixels sous le rectangle noir.

(a) Caractéristiques du bord (b) Caractéristiques de ligne (c) Caractéristiques quatre rectangles

FIGURE 2.1 – Caractéristiques de Haar

Cette méthode consiste à parcourir l'intégralité de l'image pour extraire un certain nombre de caractéristiques comme ça permet de réduire le nombre de calculs relatif à un

pixel donné à une opération ne nécessitant que quatre pixels. Parmi les caractéristiques calculées, il y'en a celles qui sont sans importance, pour sélectionner que le nombre de caractéristiques importantes Haar Cascade utilise l'algorithme d'apprentissage Adaboost [1] pour à la fin obtenir un résultat efficace des classifieurs.

Une fois le visage détecté, il faut en extraire les saillances. L'utilisation des caractéristiques rectangulaires (caractéristiques calculées par la différence des sommes des pixels de deux ou plusieurs zones rectangulaires adjacentes) de cette méthode permet de détecter les yeux ou la bouche sur un visage, et cette fonction est disponible dans OpenCV. Cette méthode des Haar a résolu beaucoup de problèmes rencontrés auparavant dans la détection de visage tel que, cette méthode fonctionne presque en temps réel sur le processeur, elle en dispose d'une architecture simple et elle détecte les visages à différentes échelles. Cependant cette méthode reste imprécise, elle donne beaucoup de fausses prédictions, elle ne fonctionne pas sur les images non frontales et elle ne fonctionne pas sous occlusion.

2.1.2 Détection avec l'Histogrammes de gradients orientées (HOG)

L'Histogramme des Gradients Orientés (HOG) est un descripteur de caractéristiques utilisé dans la vision pas ordinateur et le traitement d'images pour la détection d'objets. Il s'agit d'un modèle de détection de visage largement utilisé, inventé en 2005 par N. Dalal et B.Triggs [5], basé sur SVM (Support Vector Machine).

Dans le descripteur de caractéristiques HOG, la distribution (histogrammes) des directions de gradients (gradients orientés) est utilisé comme caractéristique. Le calcul de l'histogramme des gradients se déroule en différentes étapes.

Tout abord, nous allons mettre l'image en noir et blanc, nous enlevons toutes les autres couleurs car nous n'avons pas besoins de couleurs pour trouver un visage. Ensuite, sélectionner un patch d'image de l'image initiale et redimensionner ce patch de l'image à une taille plus petite dans le but de regarder chaque pixel de notre image. Pour chaque pixel, nous voulons regarder les pixels qui l'entourent directement. Le but est de déterminer l'obscurité du pixel courant par rapport aux pixels qui l'entourent. Ensuite dessiner une flèche pour montrer dans quelle direction l'image devient plus sombre comme le montre la figure ci-dessous :

En répétant ce processus, chaque pixel sera remplacé par une flèche. Ces flèches sont appelées gradients et elles montrent l'écoulement de la lumière à l'obscurité sur toute l'image.

La raison majeure pour laquelle cette méthode remplace les pixels par des flèches est pour avoir une représentation plus « robuste » aux conditions des images. En analysant directement les pixels, pour la même personne, les images vraiment sombres et les images vraiment claires auront des valeurs de pixels totalement différentes. En revanche, en considérant seulement la direction dans laquelle la direction change, nous arrivons à une seule représentation finale.

Cependant, sauvegarder le gradient pour chaque pixel nous donne beaucoup de détails et cela fait beaucoup de données, pour cela, nous allons découper l'image en petits carrés de 16x16 pixels chacun. Et pout chaque case, on comptera combien de pentes dans chaque direction principale (vers le haut, haut droit, haut gauche, etc....). Ensuite, nous remplaçons ce carré dans l'image par la direction des flèches. A la fin, on arrive à transformer

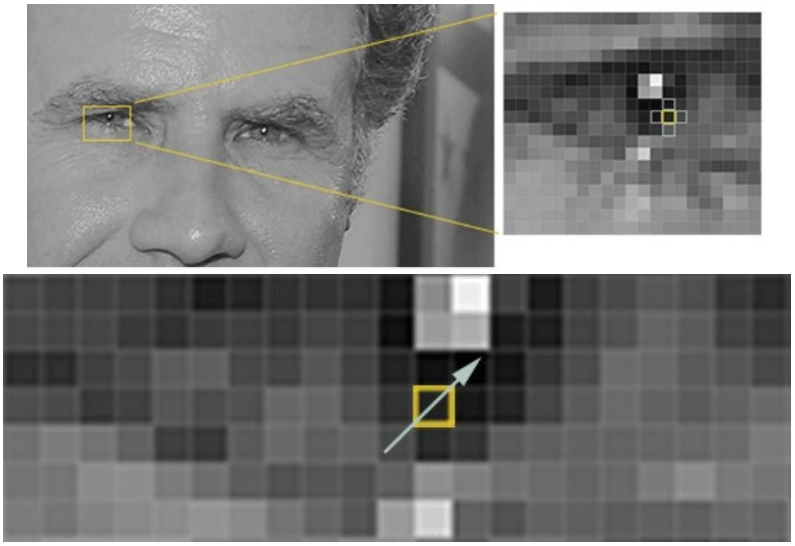


FIGURE 2.2 – construction du gradient orienté

l'image originale en une représentation très simple qui capture la structure de base d'un visage de manière très simple.

Enfin, on calcule le vecteur de caractéristiques HOG, ce dernier est un grand vecteur qui concatène plusieurs vecteurs de caractéristique de l'ensemble de l'image pour chaque carré de cette image comme il est illustré dans la figure 2.3 ci-aprés.

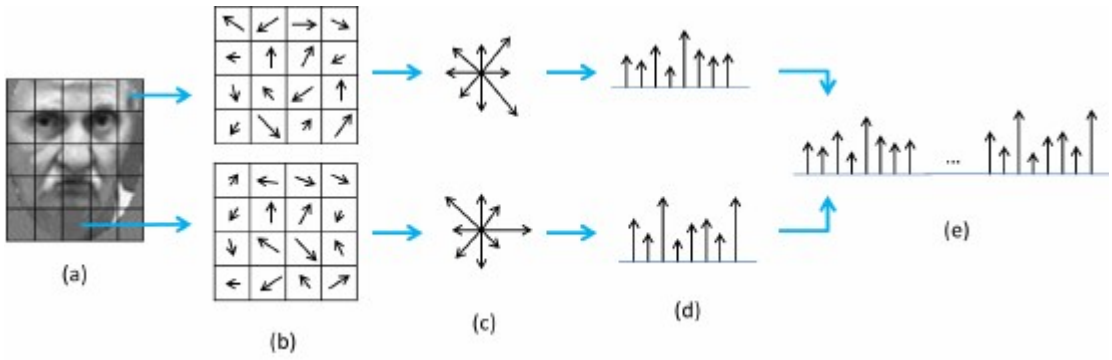


FIGURE 2.3 – Processus de la construction de l'histogramme des gradients

La dernière étape consiste à calculer le vecteur de caractéristique HOG, pour calculer le dernier vecteur de caractéristique pour l'ensemble du bloc d'image, les vecteurs de 36 x 1 sont concaténés en un seul vecteur géant.

La méthode HOG est la méthode la plus rapide sur le processeur d'après [8], elle fonctionne très bien pour les faces frontales et légèrement pour les non frontales et elle fonctionne sous petit occlusion. Cependant, l'inconvénient majeur de cette méthode est qu'elle ne détecte pas les petits visages, car elle est conçue pour une taille minimale de 80 x 80, la région d'interêt exclut souvent une partie du front et même une partie du menton

parfois, elle ne fonctionne pas très bien sous occlusion importante et elle ne fonctionne pas pour les faces latérales et les faces extrêmes non frontales, comme par exemple regarder vers le bas ou vers le haut.

2.1.3 Autre méthodes de détection

Il existe d'autres méthodes de détection de visage parmis ces méthodes, la méthode de Single Shot MultiBox Detector (SSD) proposé par W. Liu et all. dans [15].

Ce descripteur surmonte tous les défauts de la détection avec la méthode à cascade de Haar selon [8], sans compromettre aucun avantage fourni par Haar. En plus de ça, c'est la méthode la plus précise dans la détection, elle fonctionne en temps réel sur le processeur ainsi elle détecte les visages à différentes échelles (visage grand et petit) et elle fonctionne pour les différentes orientations du visage (haut, bas, gauche, droite, côté, etc) voire même sous occlusion importante. Actuellement le méthode SSD ne présente aucun inconvénient majeur par rapport à toutes les méthodes de détection de visage dans OpenCV sauf qu'elle est plus au moins lente que celle basée sur Dlib HOG avec les CNN.

Un autre méthode aussi impourante dans la détection de visage, la méthode Max Margin Object Detection (MMOD) proposé par Davis E. King dans [11]. Son modèle est le plus rapide sur les GPUs (45 ms par images), il est aussi robuste à l'occlusion et son processus de formation est très facile. En revanche, ce dernier, reste lent sur les processus (370 ms pour traiter une seule image), il est entrainé pour une taille minimale de 80 x 80, ce qui fait qu'il ne détecte pas les petits visages et sa boite englobante est encore plus petite que le détecteur HOG avec SVM.

.

2.2 Methode d'extraction des points de saillances [10]

Le processus qui est capable d'explorer un ensemble de points clés à partir d'une image de visage donnée, est appelée Localisation du repère du visage ou Face Landmark Localization en anglais (alignement du visage).

Les repères (points clés) qui nous intéressent sont ceux qui décrivent la forme des attributs du visage comme : les yeux, les sourcils, le nez, la bouche, et le menton. Ces points ont donné un excellent aperçu de la structure faciale analysée, qui peut être très utile pour un large éventail d'applications.

De nombreuses méthodes permettent de détecter ces points : certaines d'entres elles atteignent une précision et une robuste supérieures en analysant un modèle de visage 3D extrait d'une image 2D, d'autres s'appuient sur la puissance des CNN et d'autres utilisent des fonctions simples (mais rapides) pour estimer l'emplacement des points.

L'Algorithme de détection des repères de visage proposé par Dlib est l'implémentation de l'Ensemble of Regression Trees (ERT) présenté en 2014 par Kazemi et Sullivan [10]. Cette technique utilise une fonction simple et rapide (différence d'intensité des pixels) pour estimer directement la position des points de repère. Ces positions estimées sont ensuite affinées au moyen d'un processus itératif effectué par une cascade de variables explicatives. Les régresseurs produisent une nouvelle estimation à partir de la précédente, en essayant de réduire l'erreur d'alignement des points estimés à chaque itération. L'algorithme est

très rapide, en fait, il faut environ 1 à 3 ms pour détecter un ensemble de 68 points de repère sur un visage donné. La figure suivante présente l'ensemble des points de visage extrait par l'algorithme :

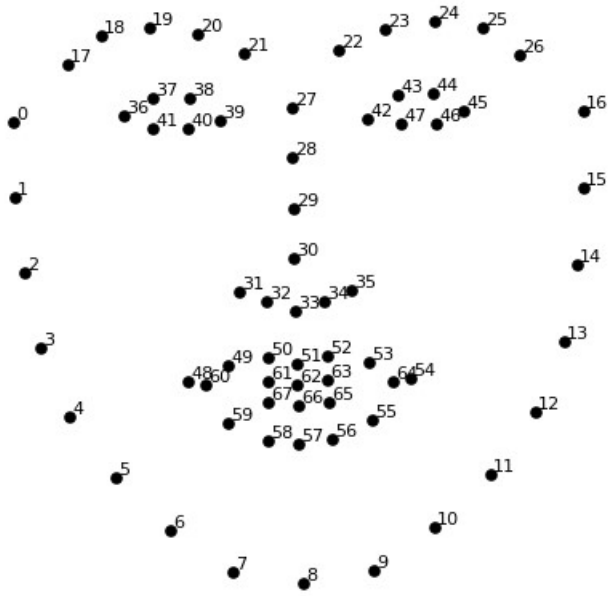


FIGURE 2.4 – L'ensemble des 68 points détectés par Dlib pré-entraîné

La région du visage peut être facilement obtenue par n'importe quel algorithme de détection de visage (OpenCV Haar Cascade, Dlib HOG Detector, CNN, . . .). Enfin, les options de formation sont un ensemble de paramètres qui définissent les caractéristiques du modèle formé. Ces paramètres peuvent être correctement ajustés afin d'obtenir le comportement souhaité du modèle généré.

2.3 Clustering

Le clustering ou la classification non supervisée en français, est une technique de classification en apprentissage automatique, en analyse et en fouille de donnée ainsi qu'en reconnaissance de formes. Il fait une partie intégrante de tout un processus d'analyse exploratoire de données permettant de produire ses outils de synthétisation, de prédiction, de visualisation et d'interprétation d'un ensemble d'individus (personnes, objets, processus, etc.). L'objectif est, à partir de données constituées d'un ensemble d'individus ou d'objets et d'une relation de proximité entre ceux-ci, de construire des groupes d'individus homogènes dans les sens où :

— Deux individus proches doivent appartenir à un même ensemble (groupe).
— Deux individus éloignés doivent appartenir à des groupes différents.

Afin de définir l'homogénéité d'un groupe d'observation, il est nécessaire de mesurer une ressemblance entre deux observations. D'où la notion de similarité et de dissimilarité

comme la montre la figure 1.5 : [14].

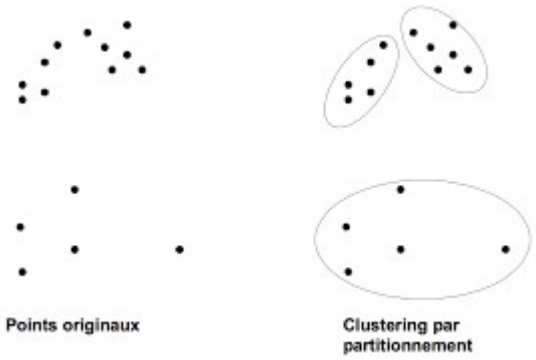
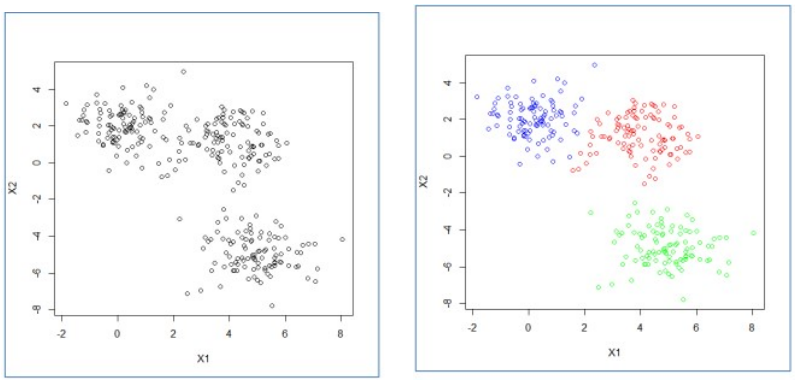


FIGURE 2.5 – Clustering

Ici, les points originaux ce sont les données constituées par exemple d'un ensemble d'individus, le clustering les partitionne dans des groupes où chaque groupe représente un ensemble d'individus qui partagent une même caractéristique. Le clustering comprend plusieurs algorithmes de classifications pour classifier chaque point de données dans un groupe spécifique tels que : k-Means, Mean-Shift, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Expectation-Maximization (EM) à l'aide de Modèles de Mélange Gaussiens (GMM) et Self Organising Map (SOM). Dans ce qui suit nous allons nous intéressés aux deux algorithmes suivent k-Means et l'algorithme Safe Organising Map (SOM).

2.3.1 K-means [16]

K-Means est probablement l'algorithme de classification le plus connu. C'est un algorithme facile à comprendre et à implémenter.



(a) avant le clustering (b) aprés le clustering

FIGURE 2.6 – K-means clustering

La figure ci-dessus montre un exemple de classification par l'algorithme k-Means, tels

que les points noirs à droites sont les points de données de départs et à la fin on affecte chaque point dans son groupe, ici nous avons trois groupes (groupes des points bleus, verts et rouges).

Pour commencer, nous avons d'abord sélectionnés un certain nombre de classes (groupes) à utiliser en initialisant aléatoirement leurs points centraux respectifs. Dans notre exemple nous avons trois classes. Les points centraux sont des vecteurs de la même longueur que chaque vecteur de points de données.

Chaque point de données (points noirs) est classé en calculant la distance entre ce point et chaque centre de groupe, puis en classant le point dans le groupe dont le centre est le plus proche. Sur la base de ces points classés, nous recalculons le centre du groupe en prenant la moyenne de tous les vecteurs du groupe.

Nous répétons cette étape pour un nombre défini d'itérations ou jusqu'à ce que les centres de groupe ne changent plus beaucoup d'une itération à une autre. Comme il est possible de choisir d'initialiser plusieurs fois le centre de groupe de manière aléatoire, puis de sélectionner le cycle qui donne les meilleurs résultats.

K-Means a l'avantage d'être assez rapide, car nous ne faisons que calculer les distances entre les points et les centres de groupe. Très peu de calculs, sa complexité linéaire est O (n).

D'autres parts, K-Means présente quelques inconvénients. Tout d'abord, nous devons sélectionner le nombre de groupes (classes). Ce n'est pas toujours évident et idéalement avec un algorithme de classification, nous aimerions qu'il soit mieux compris pas ceux-ci, car il s'agit là d'obtenir un aperçu des données. K-Means commence également par un choix aléatoire de centre de grappes et peut donc donner différents résultats des différentes exécutions de l'algorithme. Ainsi, les résultats peuvent ne pas être reproductible et manquer de cohérences.

2.3.2 Carte auto-organisatrice (SOM)

Self Organising Map ou la carte auto organisatrice (SOM), est un type de réseau de neurones artificiels (RNA) formé à l'aide d'un apprentissage non supervisé afin de produire une représentation discrète, généralement bidimensionnelle, de l'espace d'entrée des échantillons d'apprentissage. La carte est donc une méthode pour faire la réduction de dimensionnalité. Les cartes auto organisatrices différent des autres réseaux de neurones artificiels par le fait qu'elles appliquent l'apprentissage compétitif par opposition à l'apprentissage avec correction d'erreur et qu'elles utilisent une fonction de voisinage pour préserver les propriétés topologiques de l'espace d'entrée. La figure ci-dessous montre la construction de la map à partir d'un ensemble de vecteurs d'entrés :

SOM a été introduite par le professeur finlandais Teuvo Kohonen dans les années 1980 [12], elle est aussi appelée carte de Kohonen [12].

Chaque point de données dans l'ensemble des points de données se reconnait en compétition pour la représentation. Les étapes de mappage SOM commencent par l'initialisation des vecteurs de pondération. A partir de là, un vecteur d'échantillon est sélectionné de manière aléatoire et la carte des vecteurs de poids est explorée pour trouver quel poids représente le mieux cet échantillon. Chaque vecteur de poids a des poids voisins qui lui sont proches. Le poids choisi est récompensé par sa capacité de ressembler davantage à

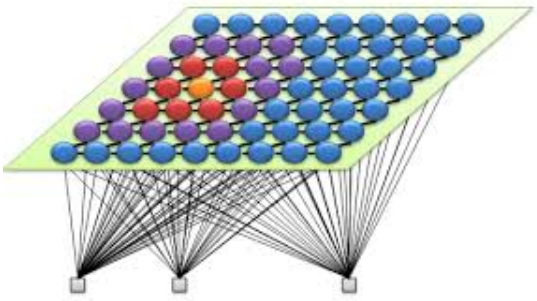


FIGURE 2.7 – Réduction de la dimensionnalité dans la SOM

cet échantillon de vecteur sélectionné au hasard. Les voisins de ce poids sont également récompensés par leur capacité à ressembler davantage au vecteur échantillon choisi. Cela permet à la carte de s'agrandir et de former différentes formes. Plus généralement, ils forment des formes carrées, rectangulaires, hexagonales et L dans un espace de fonction 2D.

L'algorithme SOM :

1. Initialiser chaque poids de chaque nœud.
2. Choisir un vecteur au hasard dans l'ensemble des données d'apprentissages.
3. Chaque nœud est examiné pour calculer les poids qui ressemblent le plus au vecteur d'entrée. Le nœud gagnant est généralement appelé unité de meilleure correspondance ou Best Matching Unit (BMU)
4. Le poids gagnant est récompensé par le fait de ressembler davantage au vecteur échantillon. Les voisins deviennent également davantage comme le vecteur échantillon. Plus le nœud est proche du BMU, plus des poids sont modifiés et plus le voisin est éloigné du BMU, moins il en apprend.
5. Répéter l'étape 2 pour N itération.

BMU est technique qui calcul la distance entre chaque poids et le vecteur échantillon en parcourant tous les vecteurs de poids. Le poids avec la distance la plus courte est le gagnant. Il existe nombreuse façon de déterminer la distance, cependant la méthode la plus couramment utilisée est la distance euclidienne.

Conclusion

Nous avons exploré, dans ce chapitre, les différents notions, existantes, liées à notre solution. Cela nous permet d'entamer les détails de notre conception dans le chapitre suivant.

Chapitre 3

Analyse et réalisation

« ...si vous ne pouvez pas l'expliquer simplement, vous ne le comprenez pas assez bien. ». A. Einstein.

Sommaire

Introduction 19
3.1 Appoche globale 19
3.2 Phase de préparation 20
3.2.1 Prétraitement 20
3.2.2 Détection du visage 20
3.2.3 Éxtraction des points de saillances 21
3.3 Phase d'extraction des VCC 21
3.3.1 Éxtraction et encodage des caractéristiques 22
3.3.2 Vecteurs caractéristiques des changements d'expressions 24
3.4 Phase de décision 25
3.4.1 K-Means 26
3.4.2 Carte auto-organisatrice dynamique 26
Conclusion 28

Introduction

Les deux premiers chapitres étaient une introduction au contexte du projet ainsi q'une présentation de toutes les notions importantes pour la compréhension de l'approche.

Dans ce chapitre, nous présentrons les différentes étapes de l'approche suivie en expliquant ce que nous avons utilisé dans chacune de ces étapes.

3.1 Appoche globale

Notre objectif, comme mentionné précédemment, est de concevoir un système qui permet à la lampe de réagir en fonction des changements des expressions du sujet qui se trouve en face d'elle.

La réalisation de ce système nécessite de passer par plusieurs étapes indispensables en commençant par la détection de visage et l'extraction des points de saillances. Ensuite nous allons passer à l'extraction des caractéristiques et l'encodage des solutions. Enfin, nous arrivons au regroupement (clustering) des différents changements et à la définition du comportement.

Pour simplifier la compréhension du fonctionnement de l'approche suivie, nous avons jugé utile de partitionner le travail en trois phases (figure 3.1) :

- La phase de préparation qui prend en entrée un flux d'image sous forme d'une vidéo (séquences d'images) pour produire en sortie un vecteur de 68 points de saillances.
- La deuxième phase, dite d'extraction des Vecteurs Caractéristiques des Changements VCC, comporte deux étapes : l'extraction des caractéristiques des *expressions statiques*, puis, le calcul des vecteurs caractéristiques des *changements d'expressions*.
- La dernière phase est la phase de décision qui consiste à regrouper les vecteurs caractéristiques en entrée afin de décider le comportement à exécuter (les commandes à envoyer à la lampe).

Nous verrons ces phases en détail dans les sections qui suit.

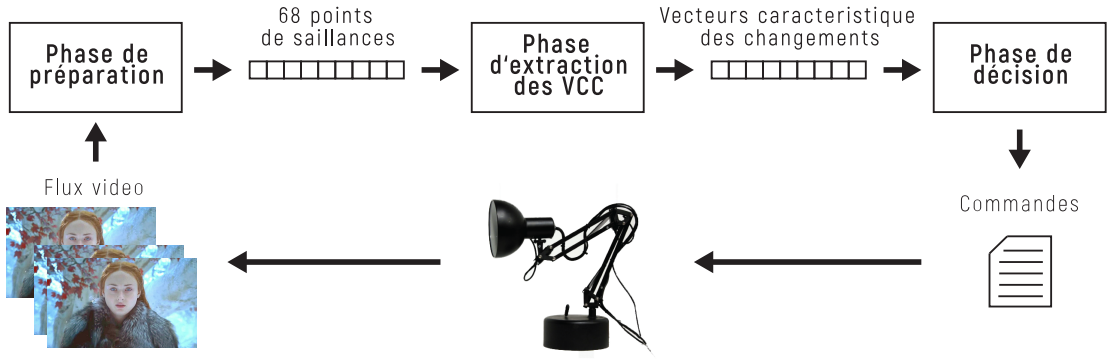


FIGURE 3.1 – Structure globale du systèume

3.2 Phase de préparation

Dans cette section nous entamons la phase de préparation, cette phase a pour but d'extraire un vecteur de points de saillances pour les faire passer à la phase suivante.

3.2.1 Prétraitement

Le prétraitement est l'étape avant-première (ou l'étape zéro), dans laquelle nous récupérons les images RGB de flux vidéo. Ensuite, nous les convertirons en niveaux de gris (grayscale) car nous n'avons pas besoin des couleurs dans nos prochains traitements (détection du visage, extraction des points de saillances...). Ces images seront ainsi redimensionnées pour diminuer le temps des traitements.

Afin d'obtenir de bons résultats dans les prochaines étapes, nous avons jugé utile d'appliquer une égalisation d'histogramme en niveaux de gris. Cette dernière permet de mieux répartir les intensités des images à faible contraste pour obtenir des images de meilleure qualité. Un résumé de l'étape de prétraitement est illustré par la figure suivante :

FIGURE 3.2 – Prétraitement

3.2.2 Détection du visage

Comme illustré par la figure 3.3 ci-dessous, l'étape de détection du visage s'agit de retrouver la zone du visage (la région d'intérêt ROI) dans une image sur laquelle nous effectuons des traitements par la suite.

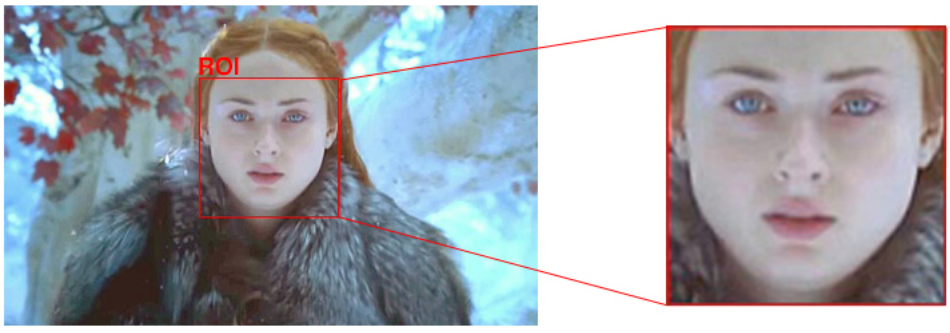


FIGURE 3.3 – Détection de visage

La méthode utilisée pour cette étape est l'histogramme de gradient orienté (HOG) décrit dans (section 2.1.2). Le modèle, selon [8], est très léger, il permet de de détecter les visages dans la plupart des cas avec un temps du réponse très court. De plus, son implémentation est fournie par la bibliothèque dlib [2]. Comme indiquée dans la section 2.1.2, le modèle ne détecte pas les visages latéraux ou quand le visage est tourné vers le bas ou vers le haut, ainsi, la ROI détecté exclut souvent une partie du front et même une partie du menton parfois.

3.2.3 Éxtraction des points de saillances

Une fois l'étape de détection de visage est terminée, l'étape d'extraction des points de saillances prend place. Cette étape utilise la région (ROI) extraite de l'étape précédente pour déterminer 68 points de saillances. Nous allons utiliser un algorithme d'estimations des points de saillances qui est basé sur une approche inventée par Vahid Kazemi et Josephine Sullivan en 2014 [10] utilisant des arbres de régressions.

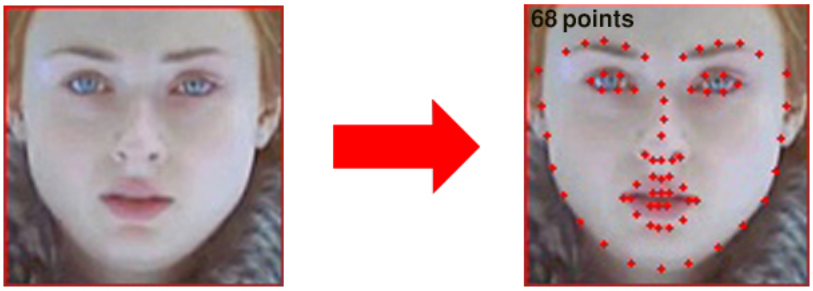


FIGURE 3.4 – Éxtraction des points de saillances

L'idée de base est de définir 68 points spécifiques (appelés repères) qui existent sur chaque visage : le haut du menton, le bord extérieur de chaque œil, le bord intérieur de chaque sourcil, etc. Ensuite, utiliser un algorithme d'apprentissage automatique pour avoir un modèle capable de trouver ces 68 points sur n'importe quel visage.

Ce modèle présente des performances intéressantes, il permet d'extraire les 68 points dans la majorité des circonstances. De plus, il est disponible dans la même bibliothèque dlib [2] ainsi que c'est le modèle le plus compatible avec le modèle de détection choisie. Néanmoins, il ne s'adapte pas avec les changements extrêmes d'expressions (Figure 3.5). Cela est dû à l'absence des visages avec des expressions extrêmes sur les images de base d'apprentissage [13] utilisé pour entrainer le modèle.

3.3 Phase d'extraction des VCC

Cette phase permet en premier lieu, de définir des caractéristiques d'expressions, choisis explicitement, à partir des vecteurs de points de saillances. Ensuite, nous utilisons ces

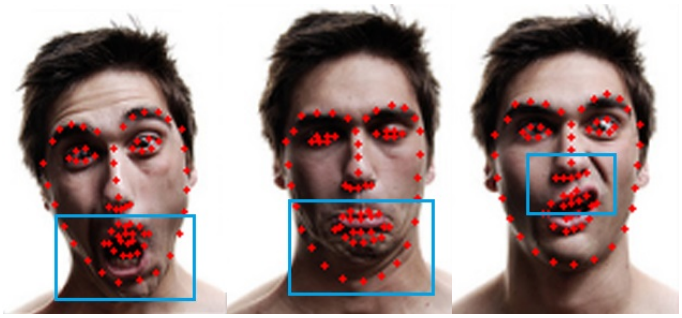


FIGURE 3.5 – Limites de la méthode d'extraction des points de saillances.

caractéristiques pour calculer les vecteurs caractéristiques des changements d'expressions qui seront utilisé pour le clustering dans la phase qui suit.

3.3.1 Éxtraction et encodage des caractéristiques

Effectuer un apprentissage sur tous les 68 points extraits est, d'une part, très couteux en terme du temps et d'espace ainsi, ils peuvent générer du bruit qui affecte la qualité des résultat obtenus. Par conséquent, nous allons extraire, à partir de ces points, des caractéristiques qui nous permettent de distinguer les différentes expressions du visage. Nous avons choisi neuf caractéristiques (à savoir : la distance des sourcils, l'ouverture des yeux, la position du visage, l'ouverture de la bouche et la rotation du visage). Ces neuf caractéristiques sont illustrés dans la figure ci-dessous.

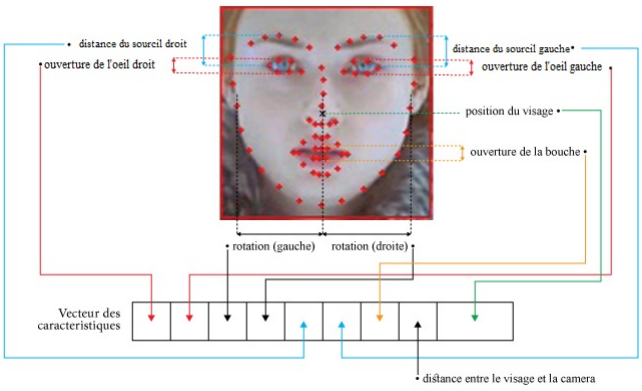


FIGURE 3.6 – Éxtraction des caractéristiques

Dans notre expérience, la lampe robotisée est mise devant différentes personnes voire les mêmes personnes peuvent se retrouver à des distances différentes de la lampe. Cela engendre une diversité de valeurs des caractéristiques définis ci-dessus pour chaque personne et aussi pour la même personne avec des différentes distances (voire la figure 3.7). De ce fait, il est nécessaire de normaliser ce vecteur de caractéristiques pour que la diversité des personnes (ou leurs distances de la caméra) n'affecte pas les résultats retournés.

FIGURE 3.7 – Exemple du problème de représentation des distances sans normalisation

Nous avons donc normalisé chacune des valeurs de nos caractéristiques entre 0 et 1, en prenant le rapport des caractéristiques sur des distances fixes dans le visage. Les détails des calculs sont présentés ci-dessous. Les points utilisés dans les formules sont présenté dans la figure 2.4. De plus, l'utilisation de la moyenne sur les points permet d'obtenir des valeurs plus stables que de prendre un seul point.

3.3.1.1 Écartement des sourcils

L'écartement des sourcils est calculé en fonction de la distance sourcil-œil et la distance sourcil-sourcil. Nous avons utilisé la formule suivante (sourcil droit) :

$$EcartementSourcil = \frac{\|P_{24} \cdot Moy(P_{42}...P_{47})\|}{\|P_{21} \cdot P_{22}\|} \tag{3.1}$$

avec : P_i : le point numero i, $Moy(X_i...X_n)$: la moyenne (le centre) des points X_i jusqu'à $X_n$et $\|x \cdot y\|$: est la distance euclidienne entre x et y

3.3.1.2 Ouverture des yeux

L'ouverture des yeux est le rapport entre la largeur d'œil et sa hauteur. Elle est calculée par la formule suivante (Oeil droit) :

$$OuvertureOeil = \frac{\|Moy(P_{43}, P_{44}) \cdot Moy(P_{46}, P_{47})\|}{\|P_{42} \cdot P_{45}\|} \tag{3.2}$$

où : P_i : le point numero i, $Moy(X_i...X_n)$: la moyenne (le centre) des points X_i jusqu'à $X_n$et $\|x \cdot y\|$: est la distance euclidienne entre x et y

3.3.1.3 Rotation du visage

La rotation est calculée par la formule suivante :

$$Rotation = \frac{\|Moy(P_0...P_3) \cdot Moy(P_{27}...P_{30})\|}{\|Moy(P_{27}...P_{30}) \cdot Moy(P_{13}...P_{16})\|} \tag{3.3}$$

où : P_i : le point numero i, $Moy(X_i...X_n)$: la moyenne (le centre) des points X_i jusqu'à $X_n$et $\|x \cdot y\|$: est la distance euclidienne entre x et y.

3.3.1.4 Ouverture de la bouche

L'ouverture de la bouche est calculée par la formule suivante :

$$OuvertureBouche = \frac{\|P_{62} \cdot P_{66}\| \times \|P_{51} \cdot P_{57}\| + \varepsilon}{\|P_{39} \cdot P_{42}\| + \varepsilon} \tag{3.4}$$

avec : P_i : le point numero i, ε est une constante, et $\|x \cdot y\|$: est la distance euclidienne entre x et y.

3.3.1.5 Distance visage/caméra

La distance visage/caméra est difficile à calculer car nous n'avons pas assez d'informations. Donc, nous avons simplifié les calculs en prenant la surface du ROI sur la surface de toute l'image :

$$Distance = \frac{h * w}{H * W} \tag{3.5}$$

où : H : la hauteur de l'image, W la largeur de l'image, h la hauteur de ROI et w sa largeur.

3.3.1.6 Position du visage

La position du visage représente le centre de ROI, elle est calculée en fonction des quatre points du coin de ROI.

$$Position = Moy(C_0, C_1, C_2, C_3) \tag{3.6}$$

où : C_i représente le point du coin i

3.3.2 Vecteurs caractéristiques des changements d'expressions

Jusque là, nous avons pu extraire les points de saillances et en définir des caractéristiques avec. Donc on peut passer ces vecteurs à la prochaine phase pour le clustering des expressions. Cependant, puisque les gens ont des proportions du visages diffèrentes, nous pouvons avoir deux comportements differents pour deux personnes avec les mêmes expressions (par exemple l'expression « neutre »). Par conséquent, nous nous intéressons au changement des expressions au lieu des expressions statiques.

Pour réaliser cela, il faut traiter plusieurs images (séquence d'images) pour qu'on puisse calculer le vecteur caractérisant le changement d'expressions sur cette séquence.

Une approche simple pour le faire, est de prendre les images du flux vidéo deux à deux. Puis, on calcule le changement entre ces deux images. Soient deux images i et j, extraites de flux vidéo dans deux instants différents tel que $temps(i) < temps(j)$. D'abord, les étapes de détection du visage, d'extraction des points de saillances et d'extraction des caractéristiques sont effectués sur ces deux images pour avoir les vecteurs caractéristiques

Vi (respectivement Vj) des images i (respectivement j). Ensuite, nous calculons le *vecteur caractéristique des changements* en appliquant une simple soustraction $Vj - Vi$ sauf pour la position, nous calculons la distance euclidienne. Nous obtenons finalement un vecteur caractérisant les changements entre l'image i et l'image j.

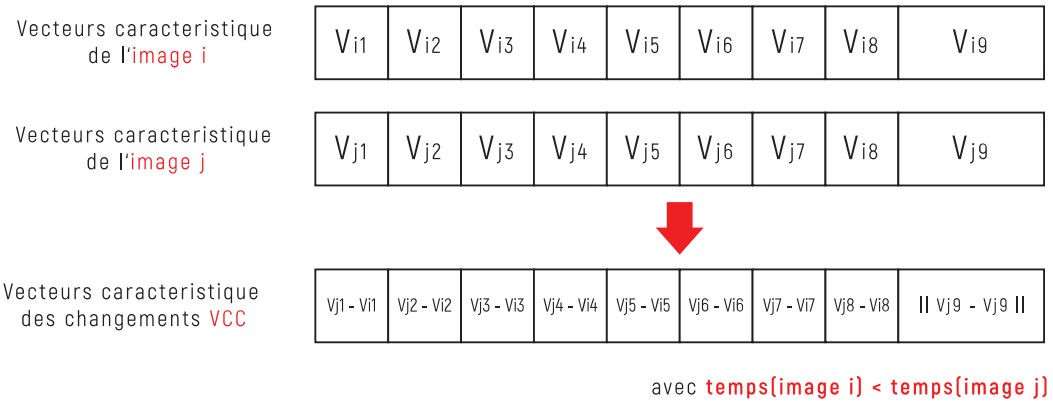
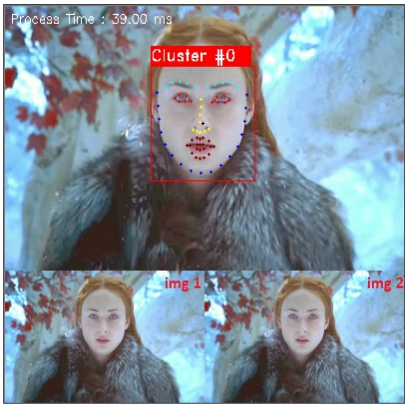


FIGURE 3.8 – Calcul des vecteurs caractèristiques des changements

Pour simplifier la compréhension, la figure 3.9 ci-après présentes deux exemples concrets des VCC.

Img1	0.28	0.26	0.0	0.26	0.53	0.43	0.0	0.13	[197, 100]
Img2	0.36	0.32	0.0	0.34	0.38	0.68	0.0	0.39	[198, 106]
VCC	0.07	0.06	0.0	0.07	-0.14	0.24	0.0	0.26	0.01

(a) Changement détecté



Img1	0.32	0.33	0.0	0.0	0.54	0.55	0.0	0.43	[198, 106]
Img2	0.32	0.33	0.0	0.0	0.54	0.55	0.0	0.43	[198, 106]
VCC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

(b) Aucun changement détecté

FIGURE 3.9 – Exemples VCC

3.4 Phase de décision

La dernière phase de l'approche, est celle de décision, dans laquelle nous prenons les vecteurs caractéristiques issues de la phase antérieure, et on applique les algorithmes

d'apprentissage KMeans et DSOM dessus pour avoir en sortie le comportement approprié.

Nous pouvons formuler cela sous forme d'un problème d'apprentissage non supervisé discret où les données sont les vecteurs caractéristiques des changements d'expresions et les clusters, ou encore classes ou groupes, représentant les comportements. En effet, nous devons entrainer un modèle f (une fonction de X dans Y) qui permet de retourner pour chaque entrée $x \in X$ le comportement approprié $y \in Y$:

$$f : X \to Y$$

Avec X ensemble des vecteurs caractéristiques des changements et Y l'ensemble des comportements.

De plus, l'algorithme doit être capable d'apprendre en temps réel en raffinant le modèle à chaque nouvelle donnée d'entrée. Ce qui rend la tache plus complexe. Pour ce faire, nous avons décidé de mettre en place deux modéles, une variante de k-moyennes (k-means) et une variante de la carte auto-organisatrice.

3.4.1 K-Means

L'algorithme de base de k-means (expliqué dans le chapitre précèdent section 2.3.1) prend en entrée une base de vecteurs de données et le nombre de clusters qui représente dans notre cas d'étude les comportements souhaités. Cependant, notre application est en temps réel, c'est-à-dire, nous aurons une nouvelle entrée dans chaque intervalle du temps, tout au long du flux vidéo. De plus, le modèle doit être capable de s'adapter avec les différentes entrées à n'importe quel instant. L'utilisation de l'algorithme de base pour l'application n'est pas le bon choix, car il nécessite un espace mémoire important dû à l'extension de la base en temps réel, et les temps d'exécution augmentent en fonction de la taille de la base ce qui résulte un temps de réponse très long.

Pour pallier à ce problème, nous avons pensé en premier temps à une approche qui consiste à construire une base, chaque période du temps, et la passer à l'algorithme. Cette approche permet d'économiser l'espace mémoire en conservant qu'une petite base chaque période du temps. Néanmoins, on aura un nouveau modèle pour chaque base diffèrent totalement du modèle qui le précède car les bases d'entrées sont différentes.

Les inconvénients de l'approche précédente nous ont amenés à penser à une deuxième approche plus performante. Cette dernière commence par la collection d'une base initiale de taille n prédéfinie, ensuite, elle passe cette base à l'algorithme k-moyennes qui permet de produire un modèle initial. Par la suite, nous gardons en mémoire que les poids des prototypes des classes (les points centrals représentant les clusters), puis, nous raffinons ce modèle pour chaque vecteur d'entrée.

3.4.2 Carte auto-organisatrice dynamique

Nous avons vu dans le chapitre deux, la carte organisatrice de Kohonen et son fonctionnent. Pour notre problème, nous utilisons une des variantes des cartes auto-organisatrices bi-dimensionnelle dite la carte auto-organisatrice dynamique (en anglais Dynamic Self Organizing Map - DSOM) proposé par Nicolas P. Rougier et Yann Boniface dans [19].

Au contraire de l'algorithme original de la carte organisatrice qui est dépendante du temps (taux d'apprentissage et voisinage), celui de DSOM est invariant dans le temps. Cela permet un apprentissage en ligne et continuer sur les distributions de données statiques et dynamiques. De plus, la densité obtenue par cette variante n'est pas directement proportionnelle à la densité de la distribution [19].

La figure ci-dessus montre la structure de la carte auto-organisatrice choisie. Pour concrétiser ce problème nous pouvons observer que les entrées représentent nos vecteurs caractéristiques des changements et la sortie est une matrice de neurones dont chacun représente un comportement bien défini. Le neurone est représenté par un vecteur des poids de taille des entrées. Ces poids seront modifiés par apprentissage tout au long de l'exécution.

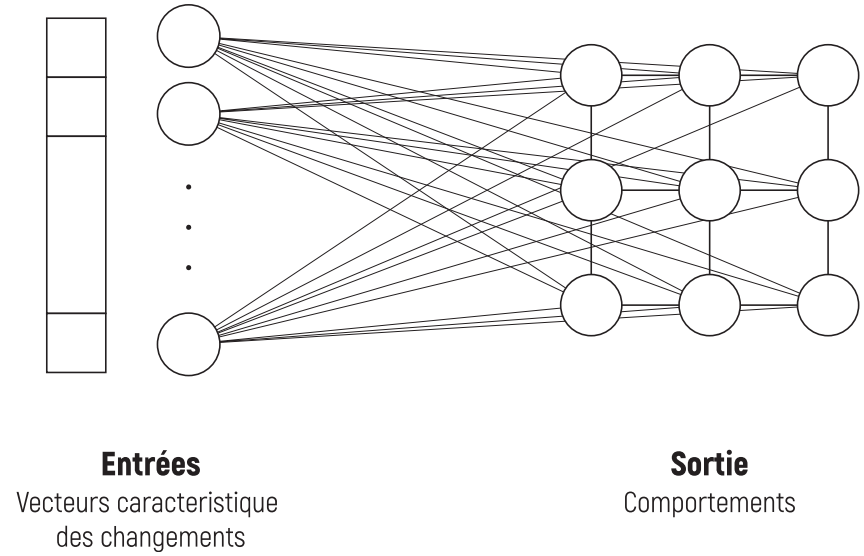


FIGURE 3.10 – Structure de la DSOM utilisée

Au départ, nous initialisons le poids des neurones aléatoirement. Ensuite pour chaque vecteur caractéristique des changements en entrée, nous récupérons d'abord les comportements appropriés à ce dernier en prenant le neurone qui a la plus petite distance (le meilleur match BMU). Puis, nous utilisons ce vecteur d'entrée et le BMU pour ajuster les poids des neurones. Cela correspond à l'entraînement de la DSOM. En effet, les poids de BMU et son voisinage, determiné par la fonction gaussienne, seront modifiés par la formule suivante [19] :

$$\Delta w_i = \varepsilon \parallel v - w_i \parallel h_\eta(i, s, v)(v - w_i) \tag{3.7}$$

Avec :
- Δw_i : le nouveau poids et w_i l'ancien poids
- v : le vecteur d'entrée
- ε : est la constante du taux d'apprentissage (learning rate)
- $h_\eta(i, s, v)$: la fonction du voisinage où
  - i l'indice du neurone

- s représente le meilleure correspondant (BMU)
- η l'élasticité (un seuil pour considéré qu'un neurone et suffisament proche pour representer les données et donc les poids ne seront pas modifiés)

Dans le cas où le neurone est suffisamment proche de la nouvelle entrée (distance $<$ seuil), nous ne modifierons pas les poids car ce neurone est considéré comme représentant optimal du groupe dont cette donnée appartient. Nous répétons ces étapes tous le temps de l'exécution.

Conclusion

Nous avons présenté, dans ce chapitre, notre solution à la problématique posée au début de rapport. Cependant, certaines méthodes, utilisées dans les différentes étapes de la solution, présente des capacités assez limitées. En effet, la détection du visage été inutile avec les orientations extrêmes du visage. D'une autre part, le model d'extraction des points de saillances nécessite des adaptations à l'application (entrainer le modèle sur des données plus adaptées). Ensuite, nous avons défini certaines caractéristiques pour les expressions, où nous avons mentionné notre intérêt aux changements des expressions au lieu des expressions statiques. Finalement, nous avons présenté la dernière phase de notre solution dans laquelle nous avons utilisé k-means et DSOM. Certaines méthodes de la solution peuvent être subi à des amélioration et servir à améliorer les résultats.

Chapitre 4

Implémentation et Tests

« ...pour comprendre mieux, il faut travailler sur son implémentation. ».
J. Guyau.

Sommaire

Introduction **30**
4.1 Implémentation **30**
4.1.1 Outils utilisés 30
4.1.2 Interfaces 30
4.2 Tests **33**
4.2.1 K-Means 33
4.2.2 DSOM 36
Conclusion **38**

Introduction

Dans la première partie de ce chapitre, nous présentons l'implémentation de notre solution, ainsi que les outils employés dans la mise en œuvre de cette dernière. La deuxième et la dernière partie sera consacrée aux différents tests, observations et conclusions.

4.1 Implémentation

Le but de notre implémentation est de mettre en œuvre une application fonctionnelle de notre solution, mais aussi, de fournir un maximum de liberté dans le contrôl de l'application afin d'analyser les différents résultats.

Pour ce faire, nous avons utilisé le langage Python, qui est un langage très puissant et qui offert une grande flexibilité et simplicité dans la programmation. Nous avons employé plusieurs outils et API pour faciliter certaines de nos tâches.

4.1.1 Outils utilisés

Les outils que nous avons utilisé sont :

— **sklearn** : est une bibliothèque libre destinée à l'apprentissage automatique. nous avons utilisé l'implémentation de l'ACP dans cette bibliothèque pour réduire les dimensions de nos vecteurs de données dans le but de les afficher dans un plan 2D.

— **matplotlib** : est une bibliothèque destinée à tracer et visualiser des données sous formes de graphiques. Elle nous a permet de tracer les différents courbes et diagrammes de notre application.

— **opencv** : est une bibliothèque libre, spécialisée dans le traitement d'images. Nous l'avons utilisé pour le prétraitement et la lecture de flux vidéo.

— **dlib** : est une bibliothèque d'apprentissage automatique. • Nous avons utilisé les modèles implémentés dans cette dernière dans l'étape de détection du visage et d'extraction des points de saillances.

— **git/github** : est un outil de gestion des versions. Il nous a facilité la gestion de notre programme voire notre dossier de travail, ainsi que la collaboration à distance.

4.1.2 Interfaces

Nous avons défini dans notre application plusieurs paramètres contrôlables à partir de la ligne de commande, par exemple : la nombre de clusters, les différents paramètres de la dsom, les figure à afficher, etc. Vous trouverez la liste intégrale des paramètres dans l'annexe.

Comme nous voulons analyser le comportement de notre application en fonction des différentes entrées et pour comparer les deux modèles utilisés dans notre solution, nous avons besoins de visualiser tous les résultats retournés d'une manière lisible et interprétable. Pour cela, nous avons implémenté plusieurs interfaces qui affichent les courbes et les diagrammes des différents attributs de l'application en temps réel. Ces interfaces sont les suivantes :

4.1.2.1 La fenêtre principale

La fenêtre principale (figure 4.1), permet d'afficher le flux vidéo, les points de saillances détectés, le numéro de comportement retourné ainsi que les deux images utilisées pour calculer le VCC actuel.

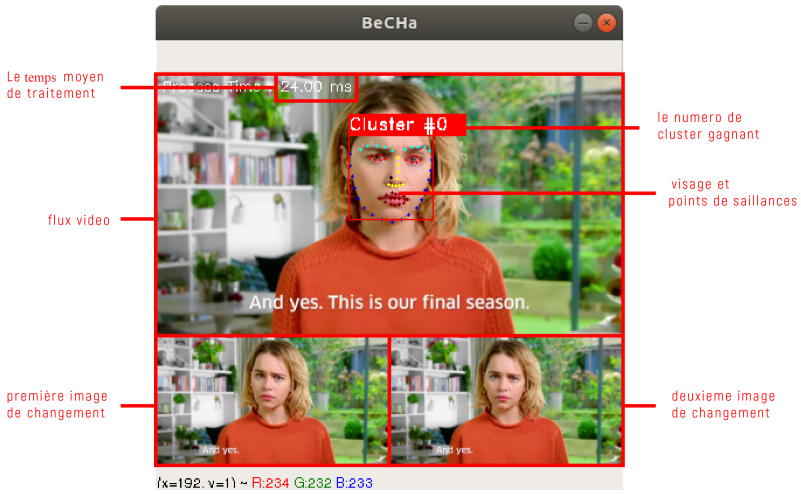


FIGURE 4.1 – Fenetre du flux vidéo

4.1.2.2 La fenêtre du plan 2D

L'interface, présentée par la figure 4.2, affiche les clusters et la nouvelle donnée dans un plan 2D à l'aide de l'ACP. Cela nous a été très utile pour visualiser la forme de nos modèles, et spécifiquement la forme de la DSOM.

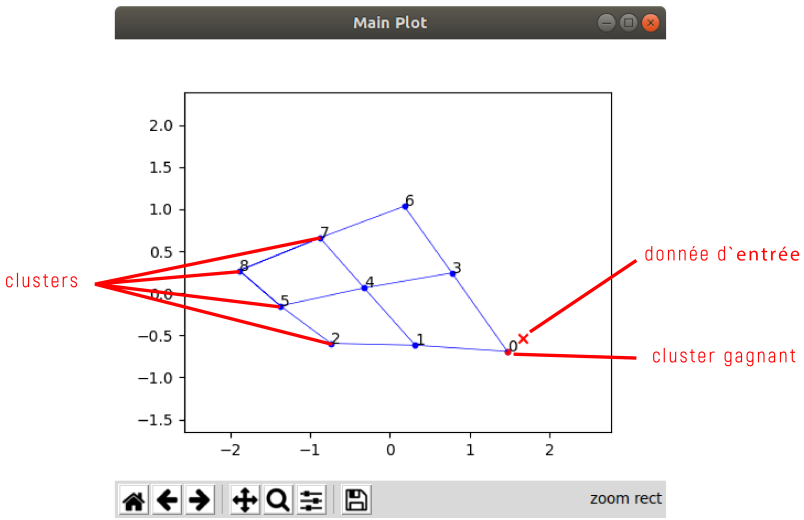


FIGURE 4.2 – Fenetre du plan 2D du modèle

4.1.2.3 La fenêtre des dimensions des clusters

La fenêtre des dimensions des clusters (figure 4.3) nous permet de visualiser les valeurs de chaque dimension pour chaque cluster. Par conséquent, nous pouvons observer facilement si notre modèle s'adapte aux données d'entrées.

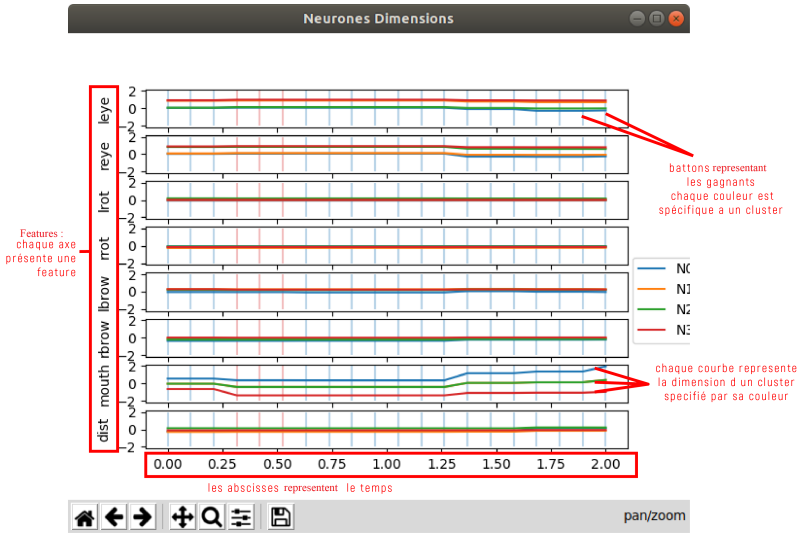


FIGURE 4.3 – Fenetre des dimensions des clusters

4.1.2.4 La fenêtre des données d'entrée-sortie

L'interface suivante (figure 4.4) est la plus importantes, nous pouvons la considérer comme une visualisation d'un résumé des valeurs de notre application. Nous pouvons suivre la variation des données d'entrée, la distance euclidienne entre la donnée d'entrée et le cluster le plus proche, ainsi que la distribution des sorties.

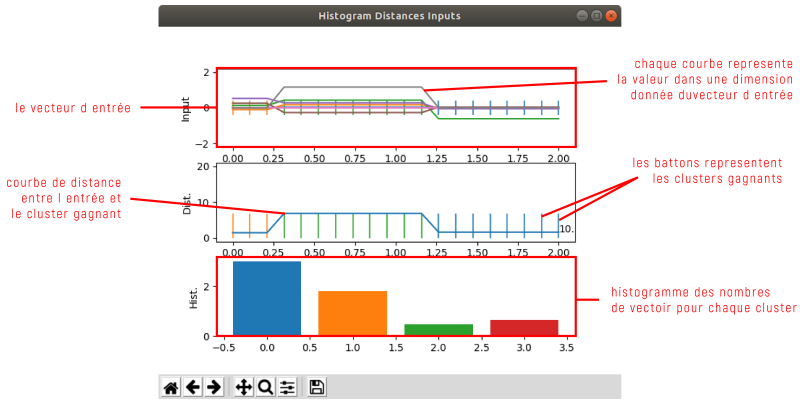


FIGURE 4.4 – Fenetre des entrées-sorties

4.2 Tests

Dans cette partie, nous essayons de tester tous les paramètres spécifiques à chaque méthode et essayer d'analyser et interpréter le comportement des modelés. Cependant, il n'existe pas une méthode (ou un algorithme) précise pour faire ces tests. Par conséquent, nous fixons les paramètres et nous varions un seul paramètre à la fois.

Avant de commencer les tests, nous avons besoin d'exécuter les tests sur les mêmes données car il est difficile d'observer le comportement sur des nouvelle données à chaque test (le cas d'utilisation directe de la webcam). Pour ce faire, nous avons enregistrer une vidéo dont nous avons essayé de varier toutes les valeurs des caractéristiques possibles. Nous l'avons utilisé pour les tests ci-après.

4.2.1 K-Means

Test 1 : cas général

La figure suivante représente les résultats des tests en utlisant les parametres, taille de la base initiale n=10 et le nombre de clusters k=9.

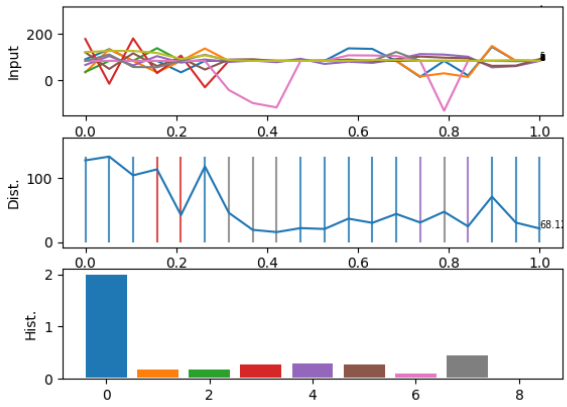
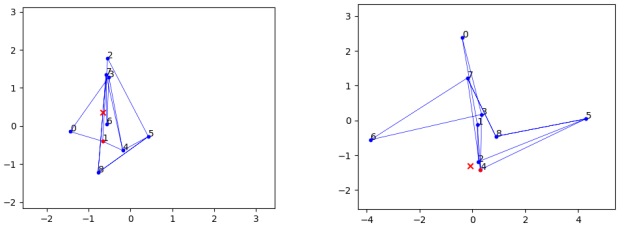


FIGURE 4.5 – Test 1 - cas général

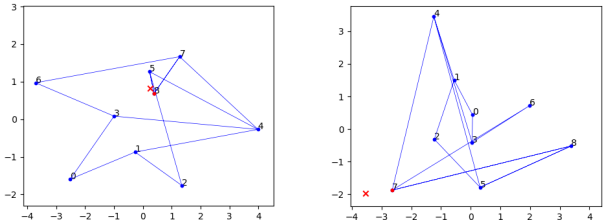
Nous remarquons qu'il y a un cluster qui gagne plus que les autres. Cela, nous a amené à établir l'hypothèse qui dit que ce clusters (avec la plus grande fréquence) représente l'état neutre (absence du changement) car cet état est souvent fréquent. Autrement, nous remarquons une décroissance de la courbe de la distance qui signifie que le modèle apprendre des données d'entrées.

Test 2 : taille de la base initiale

Pour tester le model avec les différentes tailles de la base initiale, nous avons fixé le nombre de clusters $k = 10$. Les résultats obtenus sont présenté par les figures 4.6 et **??** suivantes.



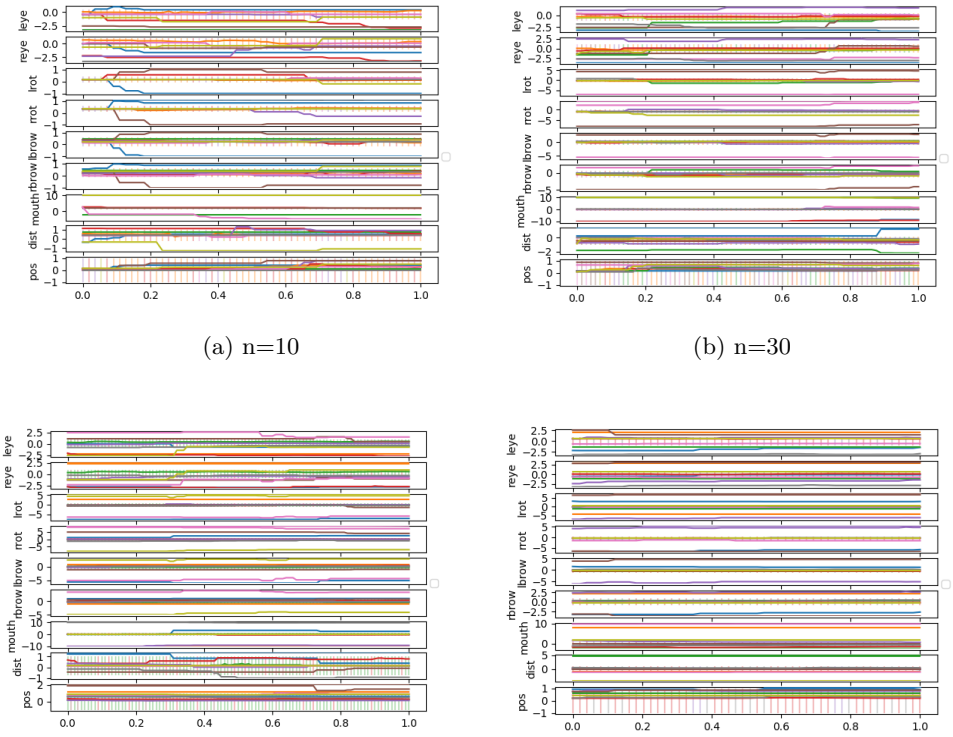
(a) n=10 (b) n=30



(c) n=50 (d) n=150

FIGURE 4.6 – Test 2 - taille de la base - plan 2D

D'après la figure 4.6, nous remarquons que incrémentation de la taille de la base engendre une meilleur distribution des clusters. Ce peut être expliqué par le fait qu'une grande base initiale permet de mieux représenter l'espace des données d'entrées. Donc, les clusters seront mieux distribué dans cet espace.



(a) n=10 (b) n=30

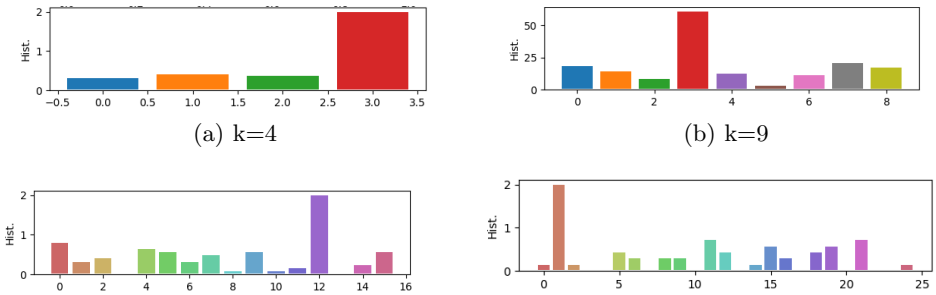
(c) n=50 (d) n=150

FIGURE 4.7 – Test 2 - taille de la base - dimensions des clusters

Nous remarquons que les poids des clusters sont sensibles aux entrées (plus adaptatives), en revanche, le vecteur d'entrée n'influe pas ces poids quand la taille de la base initiale est importante. De cela, nous pouvons déduire que la contribution d'une nouvelle donnée à l'adaptation du modèle dépend de la taille de la base initiale.

Test 2 : nombre de clusters

Nous avons fixé la taille de la base initiale à 60, et nous avons fait varier le nombre de clusters. La figure suivante représente les résultats.



(a) k=4 (b) k=9

(c) k=16 (d) k=25

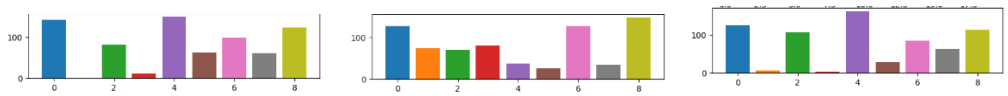
FIGURE 4.8 – Test 2 - taille de la base - dimensions des clusters

Nous remarquons que, si le nombre de clusters k est inférieur ou égal à 9, on a un avec une fréquence élever et le reste ayons une fréquence non nulle avec une variance faible entres eux (leurs fréquences sont proches), ainsi, que si le nombre de clusters k est supérieur à 9 on a un cluster avec une fréquence élevée et on a d'autres avec des fréquence nulle ou presque nulle. En plus, la variance est faible entre les autres clusters. Donc, nous pouvons dire que plusieurs clusters ne sont pas très utiles quand k est supérieur à 9.

4.2.2 DSOM

Test 1 : méthode d'initialisation

Tout d'abord, nous allons commencer par la méthode d'initialisation (regular, fixed or random) avec le paramètre Initial_Method (im), les résultat obtenu sont illustrés dans la figure suivante :



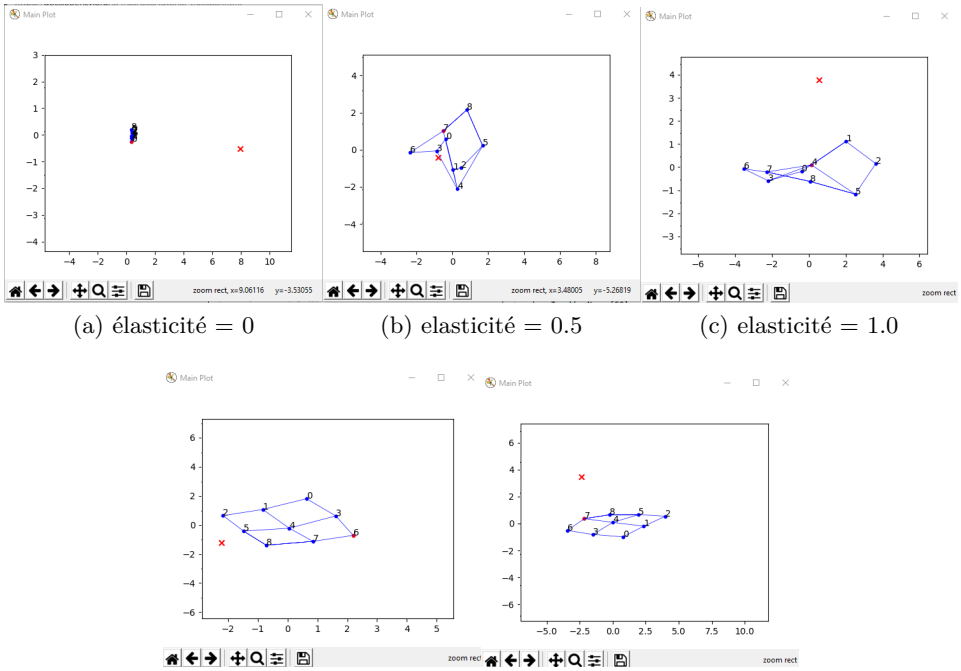
(a) im = fixed (b) im = random (c) im = regular

FIGURE 4.9 – Test avec le paramètre Init_Methode

(a) est le résultat avec im= fixed : ici, on remarque que le cluster 1 n'apparait pas chose que nous aimerions éviter d'avoir dans nos résultats. Tandis que dans les deux autres histogrammes, tous les clusters apparaissent mais avec une distribution inégale de telle sorte que les résultats avec im=random dans (b) les clusters sont apparus avec un nombre plus important que pour im=regular dans (c). Donc, on peut dire que la DSOM est mieux en mettant la méthode d'initialisation avec random.

Test 2 : élasticité

Ensuite, nous avons tester le paramètre de l'élasticité de la map, la figure suivante montre les résultats du test avec les valeurs : 0, 0.5, 1, 1.5 et 2 comme suit :



(a) élasticité = 0 (b) elasticité = 0.5 (c) elasticité = 1.0

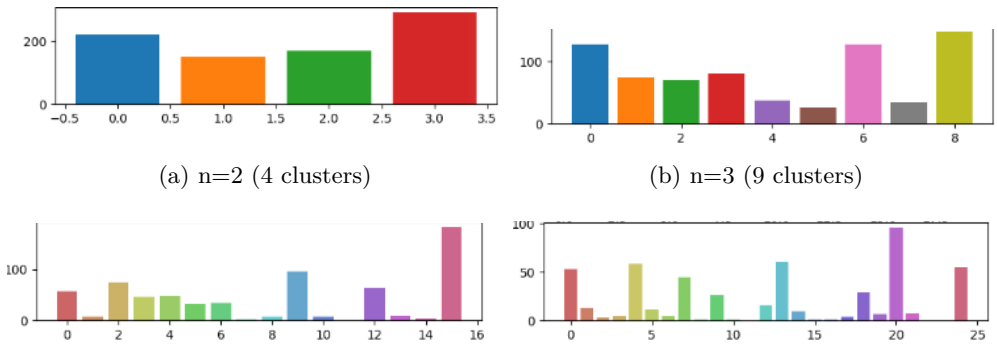
(d) elasticité = 1.5 (e) elasticité = 2

FIGURE 4.10 – Test avec le paramètre d'élacticité

La valeur de l'élasticité sur les images (a), (b), (c), (d) et (e) est 0, 0.5, 1, 1.5, et 2 respectivement. Les résultats se ressemblent beaucoup mis à part pour élasticité=0 où on peut voir que les neurones ne sont pas très réactifs avec le vecteur d'entrée, cependant le résultat avec l'élasticité=0.5 donne un résultat plus au moins meilleur en comparant aussi avec les histogrammes des clusters gagnants où nous avons remarqués qu'il y avait plus d'équilibre.

Test 3 : nombre de clusters

Dans le test suivant, nous avons essayé de fixer la taille de la map avec le nombre de neurones (n=n*n) qui donne le meilleur résultat, pour cela, nous avons essayé de tester avec n=2 donc 4 neurones (2*2), puis avec n=3, n=4 et n=5. Les résultats sont comme ceci dans la figure ci-dessous :



(a) n=2 (4 clusters)

(b) n=3 (9 clusters)

(c) n=4 (16 clusters)

(d) n=5 (25 clusters)

FIGURE 4.11 – test avec le paramèrte n

Avec ce test, on remarque qu'avec 4 neurones on a une meilleure distribution au niveau des clusters gagnants par rapport à la distribution avec 9 neurones, tandis qu'avec 16 ou 25 neurones, on des clusters qui ne sont jamais apparus et la distribution est encore plus faible dans les clusters gagnants. Chose qui est inattendue, car la méthode est censée donner un résultat équilibrer peu importe le nombre de neurones.

Pour la suite de nos tests, nous allons fixer le nombre de neurones à 4 (n=4).

Il ne nous reste qu'un seul paramètre, nous avons essayés de tester le paramètre concernant le taux d'apprentissage de la méthode DSOM (LEARNING_RATE), nous nous sommes rendus compte que ce n'était pas évident et que c'était très difficile de tester sur une vidéo avec des valeurs différentes, même avec un écart très grand entre ces valeurs. Alors, nous avons préférés le laisser à 0.1.

Conclusion

Nous avons abordé dans ce chapitre notre implémentation de la solution et les différents outils et packages utilisés. Par la suite, nous l'avons mis en examen. Cependant, par faute du temps, nous n'avons pas pu tester tous les paramètres possibles. De plus, due à l'absence d'une méthode exacte pour faire les tests et la difficulté de définir des critères dans lesquelles nous nous basons pour faire des interprétations sur le comportement du modèle, ont rendu difficile, voire impossible, d'en faire des interprétations et des conclusions sur la qualité des résultats obtenus.

Conclusion générale

Au cours de ce projet, notre contribution consiste en l'affectation, à la lampe rebotisée, la capacité de réagir (effectuer des comportement) en fonction des différents changements d'expressions des personnes à travers des expériences.

La solution que nous avons proposé comporte plusieurs étapes. Nous avons commencé par le prétraitement pour améliorer les résultats. Puis, la détection du vissage par le choix de la méthode basée sur l'histogramme de gradients orientés qui donne de meilleurs résultats comparé aux autres méthodes mais avec quelques limitations. Par la suite, nous avons utilisé un model basé sur les arbres de régressions pour extraire 68 points de saillances. Ce model est parmi les modèles les plus précis dans la littérature. Cependant, il présente un manque de précision pour certaines régions du visage très utiles pour notre application car cela est due à la base utilisée pour entrainer le model. Nous avons terminé notre méthode par l'utilisation d'une variante dynamique de la carte auto-organisatrice a deux dimensions. Cette dernière offre un grand avantage au niveau d'apprentissage en temps réel en termes d'espace et du temps.

Notre solution a été mise à l'examens à travers plusieurs tests, mais, l'absence d'une méthode exacte pour fixer les paramètres et la difficulté de définir des critères dans lesquelles nous nous basons dans nos tests, ont rendu difficile, voire impossible, d'en faire des conclusions sur les performances de notre solution. Néanmoins, Elle peut être sous réserve de plusieurs améliorations que nous avons envisagés de faire, mais malheureusement et par faute du temps, nous n'avons pas pu les mettre en place. Parmi ces perspectives nous citons :

- Mettre en place des méthodes d'évaluation de performances de model du clustering et implémenter d'autres méthodes performantes afin d'en comparer.
- Intégrer la solution sur la lampe rebotisé.
- Utiliser la régression pour encoder les changements d'expressions, et donc la possibilité de traiter plusieurs images de flux qu'on veut.
- Entrainer un nouveau model pour l'extraction des points de saillances sur des données plus adaptées à notre application.

Bibliographie

[1] Adaboost training algorithm for viola-jones object detection.

[2] dlib library.

[3] Virginie André and Yann Boniface. Quelques considérations interactionnelles autour d'une expérience robotique. In *Workshop ACAI 2018*, 2018.

[4] Stanislas BARBILLON, Marjorie JULIO, and Aurélien STAB. Lampe robot, 2017.

[5] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[6] Eva D'Alessandro and Augustin Giovinazzo. Initiation à la recherche suivi de visages et détection de saillances, 2018.

[7] Samy Dany, Tram Hugo, Nogatchewsky Matthieu, and Bauer Simon. La pinokio lampe, 2016.

[8] Vikas Gupta. Face detection – opencv, dlib and deep learning, Oct 2018.

[9] NOWAK Julien and RIMLINGER Stéphane. Projet de découverte de la recherche suivi de visage et détection de saillances, 2018.

[10] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1867–1874, 2014.

[11] Davis E King. Max-margin object detection. *arXiv preprint arXiv :1502.00046*, 2015.

[12] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9) :1464–1480, 1990.

[13] Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir Bourdev, and Thomas S Huang. Interactive facial feature localization. In *European conference on computer vision*, pages 679–692. Springer, 2012.

[14] E. Lebarbier and T. Mary-Huard. Classification non supervisée. Master's thesis, AgroParisTech, 2016.

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd : Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[16] Unnati R Raval and Chaita Jani. Implementing & improvisation of k-means clustering algorithm. *Int. J. Comput. Sci. Mob. Comput*, 5(5) :191–203, 2016.

[17] Manuel Rebuschi. Psyphine.

[18] Buisine Roman and Riou Oksana. Suivi de visage et detection d'expression, 2018.

[19] Nicolas Rougier and Yann Boniface. Dynamic self-organising map. *Neurocomputing*, 74(11) :1840–1847, 2011.

[20] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1 :511–518, 2001.

Annexe

A Liste des paramètres du programme

A.1 Paramètre générales

Usage	Description
-h	afficher l'aide
-c CAMERA	numero de la webcam utilisée
-f FILE	chemin dans lequel on sauvgarde le fichier de plotting
-e ELASTICITY	valeur d'elasticité de la DSOM
-im INITIAL_METHOD	methode d'initialisation de la DSOM (regular, fixed or random)
-lr LEARNING_RATE	le taux d'apprentissage de la DSOM
-dt DELTA	intervale du temps entres les images prises pour les traitements
-s SPEED	vitesse du plotting (1 = la vitesse maximale)
-r RANGE	taille de la plage de plotting
-pca PCA_SAMPLES	taille de la plage de l'ACP pour le plotting
-n ORDER_N	ordre de la map. ex : n = 3 implique que le nombre deneurones n*n = 3*3 = 9
-d DISPLAY	les figres de plotting a afficher c'est une chaine de trois bits XXX où chaque chiffre correspond à une figure (1 pour afficher la figureet 0 pour ne pas l'afficher)

A.2 Paramètres de DSOM

Usage	Description
-e ELASTICITY	valeur d'elasticité de la DSOM
-im INITIAL_METHOD	methode d'initialisation de la DSOM (regular, fixed or random)
-lr LEARNING_RATE	le taux d'apprentissage de la DSOM
-n ORDER_N	ordre de la map. ex : n = 3 implique que le nombre de neurones n*n = 3*3 = 9

A.4 Paramètres de K-Means

Usage	Description
-n NUMBER_CLUSTERS	nombre de clusters pour le Kmeans
-sz DB_SIZE	la taille de la base initial de Kmeans avant le raffinnement

Résumé

Le projet Psyphine vise à explorer et étudier les interactions entre l'homme et la machine ainsi, que l'attribution d'intentions et de conscience à cette dernière. Ce rapport présente notre contribution au projet, à savoir, la conception d'un système qui donne, à un prototype robotisé, la capacité d'interagir en fonctions des différents changements d'expression perçues. Le système comporte plusieurs phases dont la première s'agit de l'emploi de la méthode d'histogramme des gradients orientés combinée avec l'apprentissage automatique pour extraire des points de saillances qui seront par la suite utilisés dans le calcul des vecteurs caractéristiques des changements pour enfin effectuer un clustering utilisant deux algorithmes, K-Moyennes et la carte auto-organisatrice dynamique.

mots clés : Psyphine ; intéractions ; détection ; HOG ; dlib ; apprentissage-automatique ; carte auto-organisatrice dynamique

Abstract

The Psyphine project aims to explore and study the interactions between man and machine as well as the attribution of intentions and consciousness to the latter. This report presents our contribution to the project, namely, the design of a system that gives a robotic prototype the ability to interact according to the different perceived changes in expression. The system consists of several phases, the first of which is the use of histgram of oriented gradient method combined with machine learning model to extract landmark points that will then be used to calculate the characteristic vectors of the changes (VCC) and finally perform clustering using two algorithms, k-means and dynamic self-organizing map.

Keywords : Psyphine ; interactions ; face-detection ; HOG ; machine-learning ; dlib ; DSOM