Initiation à la Recherche

Suivi de visages et détection de saillances

Encadrants : Boumaza Amine
Alain Dutech
Boniface Yann

Etudiants : D'Alessandro Eva
Giovinazzo Augustin

Année Universitaire : 2017-2018

Table des matières

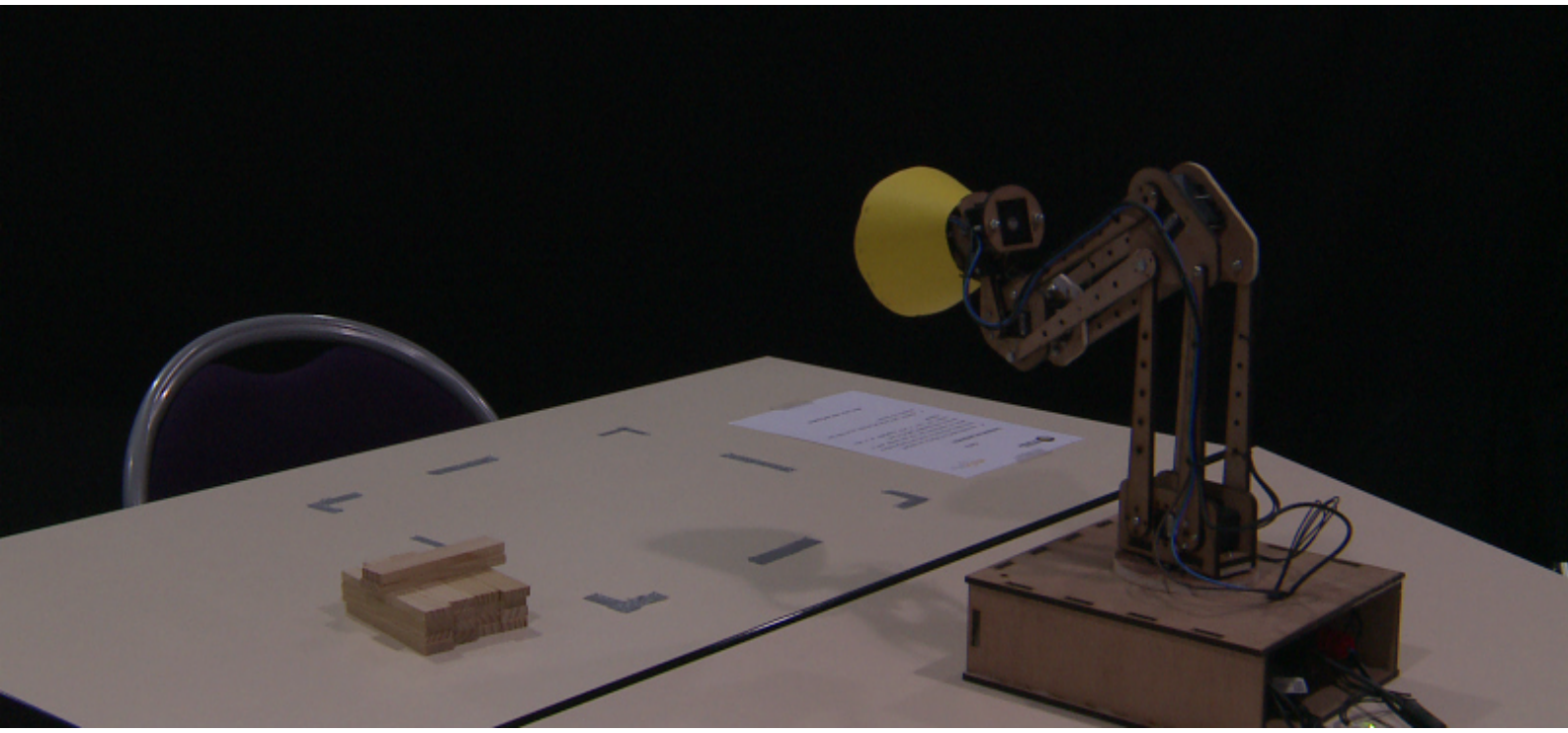
I. Introduction

Le sujet d’initiation à la recherche que nous avons choisi, à savoir « Suivi de visage et détection de saillances », est encadré par trois enseignants-chercheurs du Loria : Mr Amin Boumaza, Mr Yann Boniface et Mr Alain Dutech.

Le projet de recherche s’inscrit dans un projet plus large appelé « Psyphine ».

Dans le cadre de notre projet d’initiation à la recherche, nos encadrants nous ont présenté le prototype d’un robot censé interagir avec un utilisateur.

Nous nous sommes alors vu confié la tâche d’améliorer ce robot à l’apparence de lampe Pixar.



II. Le projet

Le Projet Psyphine

Le projet Psyphine met en relation de nombreux chercheurs et intervenants de différentes disciplines, tels que des roboticiens, cogniticiens, anthropologues, sociologues, philosophes et psychologues, dans le but d'étudier l'anthropomorphisme humain ; c'est-à-dire, la propension des humains à prêter des traits humains à des choses non-humaines (animaux, objets).
L'objectif global est de réaliser un 'test de Turing non-verbal' afin d'enrichir les connaissances sur la cognition et l'intentionnalité.

Le Robot

Le robot sur lequel nous travaillons est inspiré du robot « Pinokio » réalisé par des étudiants de l'université de Wellington.

Il s'agit en fait d'un bras robot articulé muni d'une caméra à son extrémité, contrôlé par ordinateur et déguisé en lampe Pixar.

NB : Étant donné son apparence, on admettra que ce robot est une lampe.

Il est capable d'effectuer des «chorégraphies» et un suivi de visage grâce à la technologie d'opencv.

A ce qu'on sache, le robot a subi deux améliorations notables : Une produite par trois étudiants de l'école des Mines-Nancy, qui ont implémenté un suivi de visage. Une autre qui a permis la communication à distance avec l'ordinateur de contrôle par l'intermédiaire d'un raspberry.

Ce robot sera le centre d'un test de Turing non verbal. Le but est donc de lui donner des fonctionnalités rendant son comportement le plus humain possible.

Notre objectif

En ce qui nous concerne, nous nous intéressons uniquement à développer les outils nécessaires pour permettre et faciliter l’interaction avec des interlocuteurs humains. Actuellement la lampe suit le visage d’une personne. Notre travail sera que lorsque la lampe va suivre le visage elle devra nous renvoyer chaque frame(image) de ce qu’elle filme. Quand nous récupérerons ces images nous devrons les traiter, le but sera de détecter si la personne en face de la lampe montre un changement dans les traits de son visage selon un seuil que nous aurons décidé. Si le changement dans le visage dépasse notre seuil alors il faudra adapter le mouvement de la lampe.

Pour ce faire il va tout d’abord nous falloir percevoir des changements dans le visage de la personne qui fait face à la lampe. Nous allons nous servir de la bibliothèque graphique libre « OpenCV » (Open Computer Vision) associée au langage Python comme nos prédécesseurs. Opencv est une bibliothèque graphique libre spécialisée dans la vision par ordinateur ; elle se prête parfaitement bien à nos travaux car c’est un puissant outil de traitement d'image et de vision par ordinateur. En particulier, elle dispose de tout le matériel nécessaire pour mettre en place la célèbre méthode de classification Viola-Jones basée sur « les cascades de Haar ».

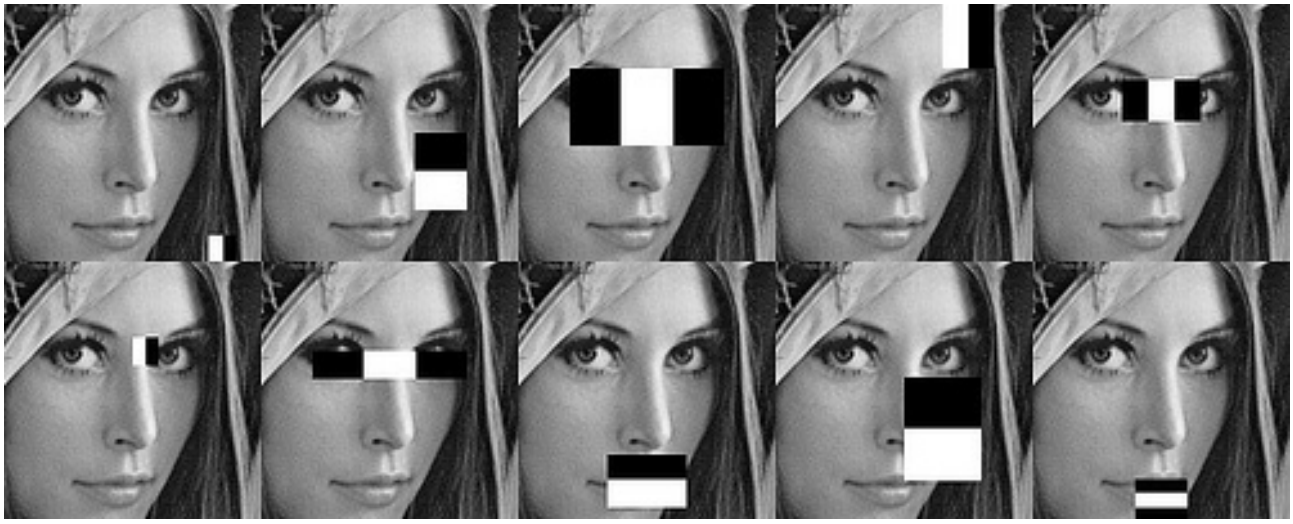
Les Cascades de Haar

« Les cascades de Haar » est une méthode proposée par *Paul Viola* et *Michael Jones* . C’est en fait un système permettant de classifier les objets (appelés features) sur une image. La méthode consiste à découper l'image en régions pour en extraire des « caractéristiques » et, suite à une phase d'apprentissage par classification supervisée, on finit par identifier les features correspondantes à un ensemble de caractéristiques.

Afin que cela marche et que la machine « apprenne » les inventeurs de ce système lui ont donné beaucoup d’images représentant un objet. Certaines images représentaient vraiment cet objet et d’autres non, ainsi la machine peut apprendre à faire la différence entre plusieurs objets et trouver un objet spécifique sur une image.

En ce qui nous concerne, on aura pas besoin de passer par cette phase d'apprentissage étant donné que opencv nous fourni des fichiers de classification (xml)où sont répertoriés les caractéristiques de Haar (obtenues par apprentissage) associées à une feature : on en a pour le visage, la bouche et les yeux .

Cette méthode, à défaut d'être hautement performante, a l'avantage d'être assez efficace et surtout très rapide ; suffisamment pour effectuer de la reconnaissance d'image sur un flux vidéo direct (webcam) sans trop de latence et sur une machine aux performances assez basses.



III. Le déroulement du projet

Choix des outils

Grâce aux Cascades de Haar nous pouvons détecter les saillances sur un visage. Pour cela il nous faut faire appel à Opencv. Sont inclus des fichiers de type « .xml » ; dedans sont contenus les caractéristiques d'une image, caractéristiques trouvées par la machine learning entraînée. Nous nous intéresserons dans un premier temps à la reconnaissance du contour du visage, de la bouche et des yeux. Tout simplement parce que ce sont les éléments qui bougent le plus sur un visage lorsque l'on veut changer d'expression. Nous pourrons introduire par la suite le traitement des lunettes, des profils et du nez.

Détecter les saillances

Avec Opencv nous pouvons donc nous exercer avec des images avant de passer directement à la caméra du robot. Détectons tout d'abord le visage et les yeux. On lit l'image désirée et on la convertie en niveau de gris, ce qui est plus adapté pour traiter les images avec OpenCV :

```
img = cv2.imread("C:\Users\Name\img.jpg")

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Il faut ensuite créer les classifieurs désirés grâce aux fichiers cités précédemment qui contiennent les caractéristiques de l'objet que nous désirons détecter :

```
face_cascade = cv2.CascadeClassifier(chemin +
"\haarcascade_frontalface_default.xml")

eye_pair = cv2.CascadeClassifier(chemin + "\haarcascade_eye.xml")

mouth_cascade = cv2.CascadeClassifier(chemin +
"\haarcascade_mcs_mouth.xml")
```

Ici nous prendrons pour le moment les détections du visage, de la bouche et des yeux. Il faut ensuite lancer la recherche sur l'image désirée en appelant une fonction de détection.

Lorsque l'on fait appel à la fonction *detectMultiScale* correspondant à la saillance que nous voulons détecter, celle-ci nous renvoie une liste de coordonnées. Plus précisément ces coordonnées correspondent à des points d'un rectangle (point en haut, il nous suffit par la suite de le tracer sur l'image que nous voulons traiter, prenons exemple avec la bouche :

```
mouth = mouth_cascade.detectMultiScale(roi_gray)   #détection sur l'image de la bouche

for (sx, sy, sw, sh) in mouth:

    #tracé des rectangles

    cv2.rectangle(roi_color, (sx, sy), (sx + sw, sy + sh), (0, 255, 255), 2)
```

Ainsi on obtient ces résultats sur différentes images :

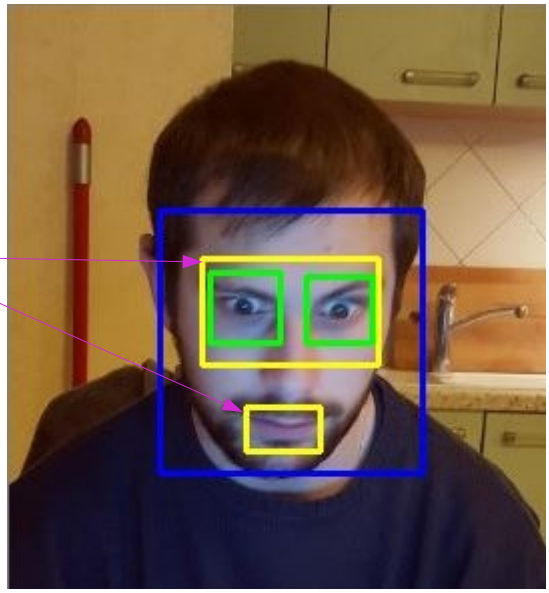


Illustration 1: Détection yeux et bouche personne 1

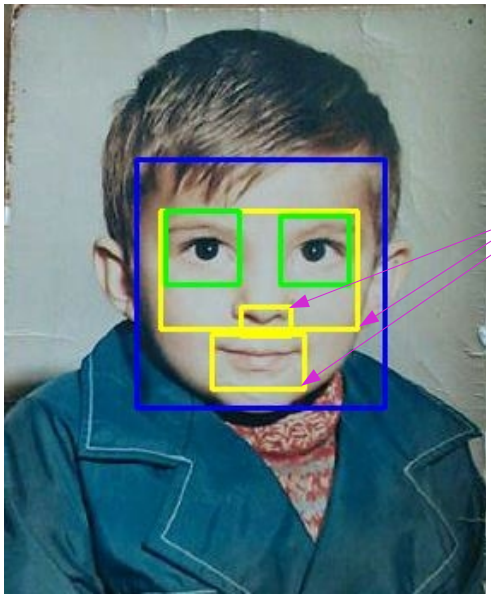


Illustration 2: Détection yeux et bouche personne 2

Malheureusement nous pouvons constater que tout ne se passe pas très bien. La détection du visage et des yeux paraît bonne mais pas celle de la bouche.

Améliorer les détections

Le classifieur détecte donc plusieurs bouches, il va falloir éliminer les fausses détections. Pour résoudre ce problème plusieurs solutions s'offrent à nous.

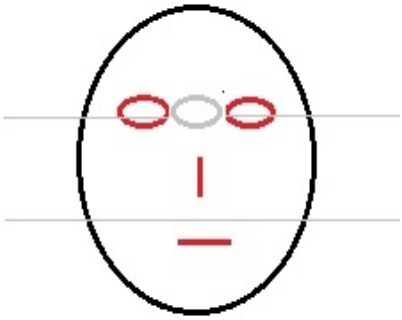


Illustration 3: Découpage du visage

Solution 1

Lors de l'échantillonnage du dessin il y a plusieurs règles à respecter afin que les saillances soient bien placées et bien proportionnées.

On sait notamment que la distance entre les deux yeux équivaut à peu près à la longueur d'un œil. Ainsi on peut trouver un couple d'yeux qui concorde.

Chaque élément du visage a sa place, le visage est découpé horizontalement en trois. Les yeux devront se trouver dans le premier tiers, la bouche elle dans le dernier. De plus nous pourront également tester l'espacement entre les yeux et la bouche pour vérifier que la paire d'yeux et la bouche sont bien placés.

Solution 2

On peut changer la méthode de classification : au lieu d'utiliser les cascades de Haar, on peut se servir des cascades de Hogg ou des cascades LBP. Le problème est qu'il est difficile d'évaluer quelle est la meilleure méthode puisque le résultat dépend énormément des images que l'on traite.

Toutefois, à la suite de quelques tests peu objectifs, on a l'impression que les cascades lbp sont plus intéressantes en terme de suivi de features et de faux positifs.

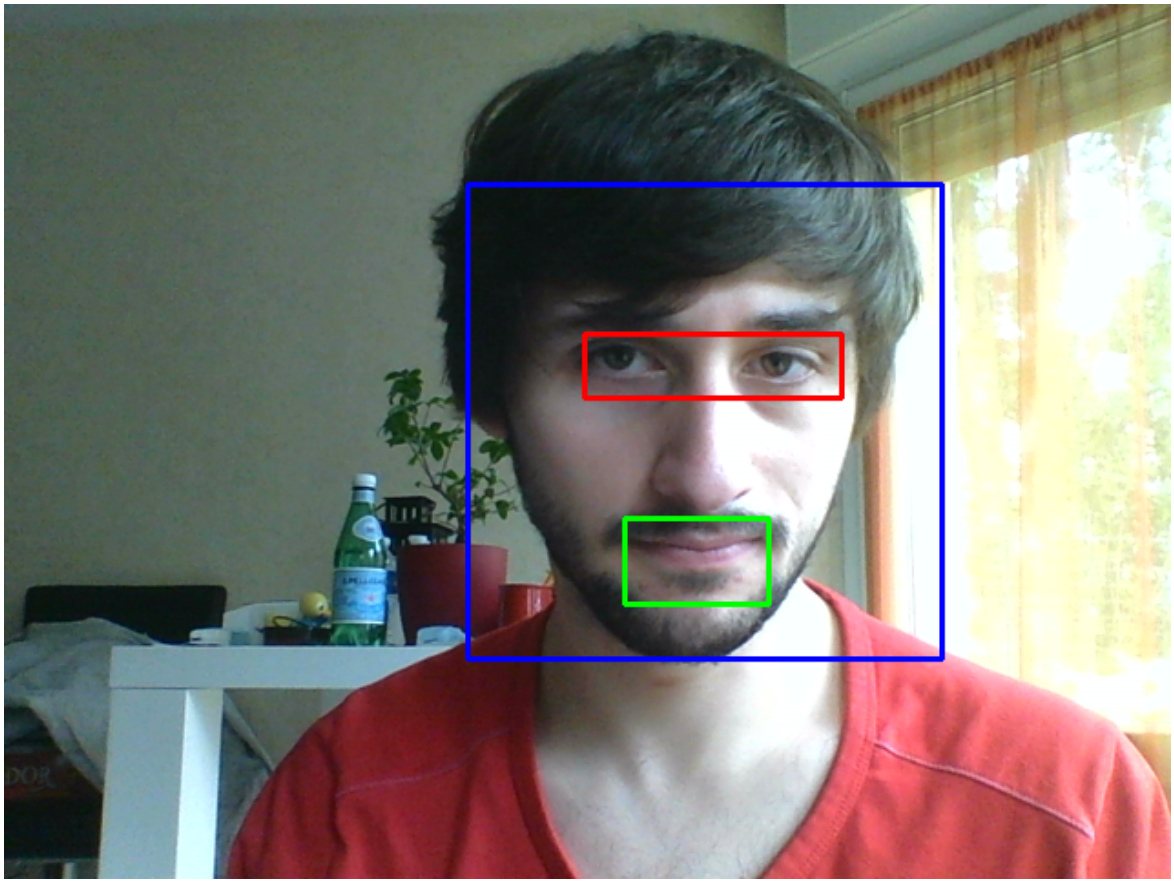
Résolution

Au final nous avons remplacé la méthode de détection des yeux ; dorénavant les yeux ne seront pas séparés mais détectés ensembles comme une paire, ce qui offre une meilleure stabilité de la détection et beaucoup moins de fausses détections.

Ensuite pour que chaque élément du visage soit à la bonne place nous avons choisi de traiter le visage en deux parties. Nous coupons la détection du visage en 2 et cherchons dans chaque partie les placements corrects de chaque membre du visage.

Ainsi, on restreint la recherche de bouches dans la partie inférieure du visage et la recherche d'yeux à la partie supérieure. On réduit alors drastiquement le nombre de fausses détections.

Pour être encore plus précis, lorsque l'on parcourt les bouches que nous avons isolés nous prenons la boîte qui se trouve à la plus proche distance de la bouche détectée sur l'image précédente. (Nous avons créé une fonction **distance(box1, box2)** qui établit une distance admissible entre 2 features.)



Évaluation des features

Principe

Maintenant que nous savions extraire les features d'un visage, on a dû mettre en place un mécanisme de quantification sur elles afin de pouvoir classifier les expressions faciales.

Invariance et Évaluation

Pour cela, on devait fixer une valeur aux features. Pour simplifier les choses, on a uniquement considéré la bouche et le visage ; les yeux subissant des changements difficilement captables. On a dû ensuite trouver une façon de quantifier cette bouche.

Il était aussi important d'évaluer la bouche par rapport au visage ; pour éviter que le rapprochement ou l'éloignement entre la caméra et la personne en face ne déclenche une détection (la bouche s'agrandissant et se réduisant avec la perspective).

L'évaluation devait donc être invariante par rapport au changement d'échelle du visage : Pour cela, on a donc pris l'hypoténuse de la boîte englobante de la bouche, puis on a divisé cette valeur par une valeur proportionnelle aux dimensions de la boîte englobante du visage.

Détection des changements d'expression

Principe

Ce qu'on veut faire, c'est intégrer l'évaluation de la bouche sur le temps pour obtenir ses variations et interpréter ces variations intelligemment pour détecter les changements pertinents.

Notamment, on souhaite détecter les variations monotones et suffisamment rapides.

Variations

On trouve la variation d'une image en faisant la différence entre l'évaluation de la bouche dans l'image courante et celle de l'image précédente. Cela revient à trouver une dérivée discrète/différence finie .

Régression Linéaire et Vitesse de Variation

Il restait un problème dont on devait se préoccuper : les parasites dans la détection de la bouche et du visage qui se manifestent par des sursauts des boîtes englobantes et, par conséquent, des variations brutales de l'évaluation.

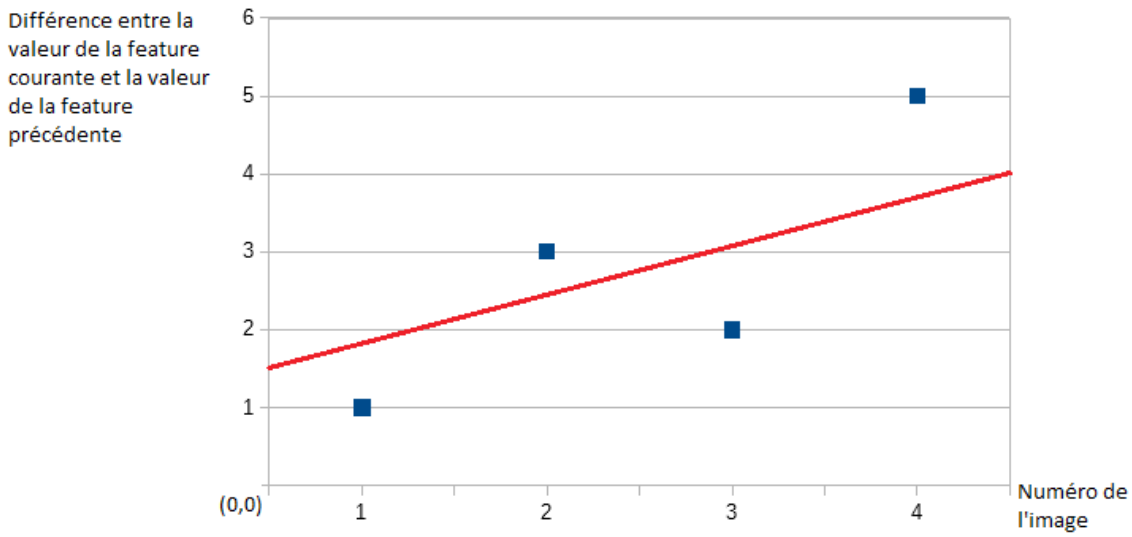
Il fallait trouver un moyen de palier à ce soucis ; l'idée a donc été de se fixer un intervalle de sécurité de taille n paramétrable, composé des n dernières évaluations ordonnées de la plus vieille à la plus récente.

On calcule alors le coefficient directeur de la droite de régression linéaire; ce qui nous donne une vitesse de variation globale de la bouche sur les n dernières images.

Exemple :

Régression linéaire sur les quatre dernières valeurs de la dérivée

Droite qui tend à être à distance minimum de chacun des points



Détection

La détection se fait lorsque la valeur absolue de la vitesse de variation globale des n dernières évaluations dépasse un seuil qu'on fixe.

Plus le seuil sera faible, plus la détection sera sensible aux changements de la bouche.

Aussi, plus n sera grand, moins la détection sera sensibles aux changements.

Dans la Théorie

Soit *V* la feature visage, H sa hauteur, W sa largeur et *(X,Y)* sa position (point en bas à gauche).

Soit *B* la feature bouche, *h* sa hauteur, *w* sa largeur et *(x,y)* sa position (point en bas à gauche).

Eval la fonction d'évaluation, qui à deux features associe une valeur réelle, telle que :

$$Eval(B,V)=\sqrt{\frac{w^2+h^2}{W^2+H^2}}$$

Maintenant, soient $\mathbf{B_i}$ la bouche et $\mathbf{V_i}$ le visage dans l'image **i**.

On trouve $\mathbf{d_i}$, la dérivée sur le temps de l'évaluation au temps i :

$$d_i=Eval(B_i,V_i)-Eval(B_{i-1},V_{i-1})$$

Puis on trouve $\mathbf{r_i}$ le coefficient de la droite de régression linéaire au temps i sur les n dernières valeurs de la dérivée :

$$r_i=\frac{\left(n\sum_{k=i-n}^{i}k\,d_k\right)-\left(\sum_{k=i-n}^{i}k\right)\left(\sum_{k=i-n}^{i}d_k\right)}{\left(n\sum_{k=i-n}^{i}k^2\right)-\left(\sum_{k=i-n}^{i}k\right)^2}$$

On règle ensuite un réel $\mathbf{S}>0$ comme seuil de détection. On a alors :

$$\forall i\in\mathbb{N},\ |r_i|\geq S\ \Rightarrow DETECTION$$

En Pratique : Résultats

Avec cette implémentation plutôt sophistiquée, notre programme fonctionne étonnamment bien malgré la piètre qualité vidéo de nos caméras et l'extraction de features pas toujours bonne.

En choisissant un bon seuil, il ne détecte pratiquement que les mouvements de bouches intéressants même lorsqu'ils sont légers (et non pas exagérés) et ce peu importe la distance, ie. même si un mouvement de bouche paraît plus infime de loin, il sera pris en compte en fonction de la distance du visage par rapport à la caméra. Le rapprochement et l'éloignement de la caméra ne produit aucune détection indésirées.

IV. Difficultés Rencontrées

Jusqu'ici, le plus difficile a été le commencement du projet où l’on cherchait une amélioration intéressante pour la lampe, ce qui nous a fait partir sur des objectifs un peu trop ambitieux.

On a notamment voulu mettre en place un réseau de neurones convolutifs pour pouvoir effectuer nous-même un apprentissage à partir de la « Cohn-Kanade Database » (CK+) qui est une base de données répertoriant une très grande quantité d'images de visages déclinés avec différentes expressions faciales.

On a tenté de se documenter via internet mais, après y avoir consacré un peu trop de temps, on a constaté que le travail demandé ne serait pas réalisable. En tout cas, pas en un semestre.

Néanmoins, cela nous a permis d'en apprendre un peu sur les réseaux neuronaux et de comprendre un peu mieux leur fonctionnement.

Par la suite nous avons rencontrés quelques soucis pour l’installation d’opencv sur python via Pycharm. De plus nous ne travaillions pas sur les mêmes plateformes (Linux et Windows). Ceci a causé quelques soucis de coordinations au niveau du code.

Nous ne connaissions pas vraiment le langage Python, nous l’avions simplement survolé les années précédentes. Nous pouvons donc oser dire que parfois nous sommes restés coincés sur des problèmes de langage.

Conclusion

Pour finir nous avons donc fait de la détection de changement sur un visage. Malheureusement nous n’aurons pas eu le temps de tester tout cela sur la lampe robot. Peut-être que nos successeurs pourront avoir ce plaisir.

Ensuite il resterait plusieurs améliorations à faire pour continuer ce projet, tout d’abord nous nous sommes vraiment concentrés sur la bouche et n’avons pas vraiment pris en compte les autres parties du visage. Une chose à faire serait d’ajouter les détections de changement dans les yeux ou même le nez.

Il restera également à décider ce que la lampe doit faire en fonction des situations. Si un mouvement est assez brusque, celle-ci pourrait avoir une réaction en particulier.

Au final nous avons découvert un nouveau langage qui pourra peut-être nous être utile par la suite. Nous avons constaté que les possibilités de détection de saillances étaient très grandes et très évoluées de nos jours. Et de ce fait nous n ‘avons touché qu’à une infime partie du sujet.