**PROJET SYNTHESE :**

**RECONNAISSANCE DE SIGNES DE LA MAINS**

Par

Vincent Thomassin-Rochon

Présenté à

Jean-Christophe Demers

Cégep du Vieux Montréal

Synthèse

Cours 450-B65-IN, groupe 00001

02 Octobre 2019

# Présentation générale

Le but du projet est de créer un programme qui va aider les gens à apprendre ou à perfectionner leurs connaissances du langage des signes via une reconnaissance des placements des doigts d’une main dans une photo prise par eux-mêmes. Pour ce faire, les usagers du programme pourront prendre une photo d’une de leurs mains imitant une lettre du langage des signes. Si le signes est bien fait, alors le programme affichera la lettre représentée. De plus, les usagers pourront consulter une liste d’image démontrant comment reproduire toute les lettres du langage des signes. Pour finir, ils pourront aussi choisir de tenter d’imiter une lettre, dépendamment du placement des doigts, le système affichera un succès ou un échec. Le programme sera codé en C++ avec l’aide de la librairie Qt et Open Cv.

# Présentation précise du projet

Pour entrer dans d’avantage de détails, le projet seras donc réaliser en langage c++ avec l’aide de la librairie Qt. Bien que la librairie Qt soit aussi disponible dans le langage python, le langage de programmation c++ est plus intéressant car il va me permettre de manipuler bits par bits les images qui seront traiter lors du déroulement du programme. De plus, il est intéressant car il va permettre de développer des connaissances dans ce langage. La librairie Qt seras utiliser pour produire l’interface graphique que les usagers utiliseront.

L’interface graphique contiendras l’image prise avant que le traitement de celle-ci débute. Il y aura aussi un bouton pour créer ou récréer la carte d’éclairage et un message disant l’état de la carte (si elle est créée ou non). Comme il faut recréer la carte d’éclairage à chaque moment que l’éclairage change, le bouton seras activer par défaut, et ne seras pas désactiver une fois la carte créer au cas où l’usager voudrais recréer la carte. Il y aura un bouton pour prendre une capture d’image du vidéo représenter à l’écran au même moment. Étant donné que le traitement de l’image ne sera pas instantané, une barre de chargement se retrouveras sur l’interface graphique afin de montrer l’état du progrès du traitement de l’image. La lettre reconnue par le programme ainsi qu’un état (succès ou échec de la reconnaissance de la main) seras afficher afin que l’utilisateur puissent savoir s’il a réussi ou non à faire ce qu’il désirait. Avec l’aide de deux boutons mutuellement exclusif, l’usager pourras décider s’il souhaite essayer de reproduire le signe d’une lettre ou s’il veut signer une lettre et laissé le programme deviner qu’elle lettre l’usager à signé. Dans le cas où l’usager voudrais signer une lettre, une image de référence de la lettre apparaitras pour guider l’usager. Étant donnée certaine contrainte de difficulté, seulement une dizaine de lettre seront disponible. Les lettres disponibles à l’usager seront l’A, le B, le D, le E, le F, le G, le L, le M et le Y. Ces lettres ont été choisie car lorsqu’elles sont signées dans le langage des signes français (LSF), les 5 doigts sont reconnaissables et distincts et sans obstruction. Ceci facilitera la reconnaissance des doigts lors du traitement des images. Dans le cas où l’usager voudrait spécifier une lettre qu’il voudrait imiter, un champ de texte pour spécifier la lettre qu’il veut seras afficher.

Il faudra filtrer une première fois l’image afin d’enlever le bruit. Pour ce faire, un filtre médian seras appliquer individuellement pour chaque pixel de l’image. Ce filtre donnera à chaque pixel la valeur RGB médiane de tous les pixels l’avoisinant et l’incluant. Par la suite, une convolution avec une distribution de type gaussienne seras appliquer afin de réduire d’avantage les bruits de l’image et facilité le traitement par la suite. Une convolution est un procédé qui multiplie chaque voisin avec son équivalent positionnel d’un filtre puis fais une sommation sur le tout afin de donner la nouvelle valeur RGB du pixel central[[1]](#footnote-1).

Il faut ensuite passer à l’uniformisation de l’éclairages. Pour ce faire, une carte d’éclairage seras créer afin de faciliter la segmentation et la classification éventuelle. La carte sera créée avec l’aide d’une cinquantaine d’image de fond identique ne contenant aucun signe de la main. Ces images passeront ensuite parmi plusieurs filtres ainsi que des fusions afin que de finir qu’avec une seule image d’arrière-plan qui correspondras à la carte d’éclairage. Donc les images passeront dans un filtre de convolution 9x9 avec une distribution normale, par la suite une moyenne des images seras effectué afin d’obtenir une image synthèse. Puis l’image passeras dans un filtre maximum d’une taille de 9x9. Un filtre maximum prend en considération la valeur du pixel et des voisins formant un carré de la taille du filtre alentours du pixel et garde la valeur la plus élevé de ceux-ci. Par la suite, un autre filtre de convolution avec une distribution uniforme mais cette fois-ci avec une taille de 151x151. Puis pour finir, une uniformisation et une normalisation. L’uniformisation est le procédé de diviser l’image source par la carte d’éclairage (pixel par pixel) tout en évitant la division par 0. En ce qui concerne la normalisation, le but est d’éviter les cas divergent et s’assurer que la valeur des pixels est dans une plage de valeur acceptable tel que [0,1] ou [0,255]. Pour normaliser, on prend en compte la valeur minimal et maximal de tous les pixels de l’image ainsi que la valeur de normalisation souhaité. Il s’agit de soustraire la valeur du pixel par la valeur minimale, puis diviser le résultat par la différence entre la valeur maximal et minimal. Il ne reste qu’à multiplier cette dernière valeur par la valeur de normalisation afin d’avoir la valeur normalisée souhaité. Dans mon cas, je choisi de normalisée pour une plage de valeur de [0,255], car c’est la même plage de valeur que la plage des couleurs RGB traditionnel.

Afin de poursuivre le traitement de l’image, une segmentation de l’image traité seras effectué. Cette segmentation a pour but identifier les pixels de l’image qui correspondent à notre recherche. Dans notre cas, les pixels que nous désirons être capable d’identifier sur l’image sont les pixels verts. Il s’agit des pixels verts, car pour faciliter la reconnaissance des signes, les usagers seront équipés d’une sphère verte en mouse sur le bout de chaque doigt et du pouce. La forme sphérique des objets au bout des doigts est très importante, car cette forme implique qu’elle ne sera jamais déformée par la perspective contrairement à la déformation qu’un losange ou un carré pourrait subir. La couleur verte des sphères porte moins d’importance. Elle a été sélectionnée car elle est différente de tous les tons de couleur de peau possible chez l’humain, et donc les pixels verts ont moins de chance d’appartenir à la main d’un humain. Étant donné que la segmentation va identifier tous les pixels verts, la segmentation va donc éliminer tout ce qui n’est pas un pixel vert en changeant la valeur des pixels non-vert pour 0 et la valeur des pixels vert pour 1. Le résultat sera donc une image binaire ou tous les pixels valant 1 seront les éléments recherchés, et les pixels valant 0 sont ceux qui ne sont pas important au traitement. Cependant, pour que la segmentation soit effectuée de manière adéquate, il faudra séparer l’image traité en trois images. Ces trois images correspondront aux bandes rouge, bleue et verte qui constitue la première image. Lorsqu’un pixel seras dans la plage acceptable pour le rouge, le bleu et le vert. Le pixel sera gardé, sinon le pixel se verra être attribuer la valeur de 0.

Il est possible que certain « anomalie » se crée lors du traitement ce qui résulterait en une image n’ayant pas des formes sphériques parfaitement définie et sphérique. Par exemple, une réflexion lumineuse vers la sphère pourrait causer les pixels plus illuminés d’être éliminé lors de la segmentation. Pour pallier à ce problème, un algorithme de remplissage de forme seras appliquer à l’image afin que chaque forme individuelle soit complétée autant que possible si elles ont besoins d’être compléter. La première étape de cet algorithme est d’identifier un pixel qui appartient à l’arrière-plan. Pour ce faire, il faudra trouver le premier pixel du contour ayant plus de voisins pareil le précédent que le nombre de pixel fessant la taille du plus gros trou de la sphère imparfaite. Par la suite, avec un simple algorithme de « flood filling » récursif à 4 voisin tout l’arrière-plan seras attribué la valeur de 1 afin qu’il ne reste que les trous sur l’image. Une fois l’arrière-plan rempli, toute l’image seras inversé afin que les trous représentent le 1 et que tout le reste de l’image aie une valeur de 0. Ceci nous permettra de fusionner notre image inversée avec notre image traitée de base (l’image avec les trous et les valeurs désiré valant 1) afin d’obtenir une image qui ne contient que des formes pleines et qui a fini d’être traité. Or, le programme ne s’arrête pas là.

La prochaine étape consiste à sortir toutes les informations appartenant à la sphère de l’image qui a fini d’être traité. On veut notamment savoir la taille de chaque sphère, son centre approximatif, et la position d’un pixel faisant partie de la forme. Pour ce faire, un algorithme d’extraction de blob seras appliquer à l’image. Cet algorithme d’extraction va appliquer le « flood filling » afin de détecter toute les pixels contigus d’une forme. Par la même occasion l’air occuper par chaque forme seras calculer afin de sauvegarder cette mesure, ainsi qu’une position en X et en Y et une position approximative du centre de la forme. Avec toute ces informations, il sera donc possible d’attribuer une position à chacune des formes qui va correspondre à quel doigt elle est supposée représenté.

# Présentation des patrons de conceptions envisagés

En ce qui concerne les patrons de conceptions, un singleton sera utilisé pour la connexion à la base de données embarqué afin de limiter le nombre de fois qu’une connexion seras fait à la base de données, ou le nombre de fois qu’elle seras instancier.

De plus, le patron de conception « stratégie » seras utiliser afin de permettre la permutation des algorithmes de convolution, de médiane, de segmentation, de remplissage de zone interne et de calcul des descripteurs de forme de l’image lors du processus du traitement de l’image.

# Présentation des aspects techniques

1. http://www.f-legrand.fr/scidoc/docimg/image/filtrage/convolution/convolution.html [↑](#footnote-ref-1)