GIF-4100 et GIF-7001 Vision numérique
Robert Bergevin
12/12/2024 - Automne 2024

Rapport final

Reconnaissance de l'alphabet de la langue des signes

Membres de l'équipe :

- Emma KILBERTUS - 537305286
- Nicolas TAUPIN - 53305474
- Alexandra Maria ONCIUL - 537305977
- Alexandre MIRALLES - 537316065
- Grégoire DE SAUVAGE VERCOUR - 536952373

I. Mise en contexte et description de la solution proposée

La détection du langage des signes par webcam représente une avancée technologique significative dans le domaine de l'accessibilité et de la communication. Notre projet vise à développer un début de solution en proposant un système capable de reconnaître les lettres de l'alphabet du langage des signes francophone en temps réel à l'aide d'une webcam. En combinant des techniques de vision par ordinateur et d'apprentissage automatique, ce système peut offrir une solution efficace pour faciliter la communication entre les personnes sourdes ou malentendantes et celles qui ne maîtrisent pas le langage des signes.

L'objectif principal de ce projet est de créer une solution alternative aux implémentations existantes qui utilisent la bibliothèque Python mediapipe. En effet, alors que ces solutions s'appuient sur des modèles d'intelligence artificielle pour détecter la main de l'utilisateur, notre projet se distingue en n'utilisant que des méthodes mathématiques de vision numérique, comme celles vues en cours, pour réaliser cette détection.

Notre solution s'articule autour de trois composantes principales que sont la création d'un dataset personnalisé, l'entraînement d'un modèle de reconnaissance, et l'application en temps réel.

Pour la création du dataset, nous avons développé un programme qui permet de capturer automatiquement des images de la main pour chaque lettre de l'alphabet. Le système définit une région d'intérêt (ROI) rectangulaire dans laquelle l'utilisateur doit placer sa main. Pour isoler efficacement la main du reste de l'image, nous utilisons une technique de soustraction d'arrière-plan dynamique. Le programme commence par une phase cruciale d'initialisation de 60 frames pendant laquelle il capture l'arrière-plan sans main. Durant cette phase, le système accumule et moyenne les images pour construire un modèle stable de l'arrière-plan. Cette accumulation utilise un poids de 0.5, permettant ainsi une adaptation progressive aux changements d'illumination et aux petites variations dans la scène.

La détection de la main se fait à travers plusieurs étapes successives de traitement d'image. Le système commence par extraire la région d'intérêt définie, puis convertit cette zone en niveaux de gris, réduisant ainsi la complexité du traitement tout en conservant l'information structurelle essentielle. Un flou gaussien est ensuite appliqué pour réduire le bruit haute fréquence qui pourrait perturber la détection. L'étape clé du processus est la soustraction d'arrière-plan, réalisée entre l'image courante et l'arrière-plan accumulé, produisant une image de différence où les zones de

mouvement apparaissent clairement. Cette image est ensuite binarisée, produisant une image binaire où la main est isolée en blanc sur fond noir. Le choix de ce seuil est crucial car il impacte directement la qualité de la segmentation : une valeur trop basse capturerait trop de bruit, tandis qu'une valeur trop élevée risquerait de perdre des détails importants de la main.

Le programme guide l'utilisateur pour capturer 250 images pour chaque lettre. Pour assurer une bonne diversité dans les données d'entraînement, l'utilisateur est encouragé à varier les angles et positions de la main. Chaque image capturée est automatiquement prétraitée selon le processus décrit ci-dessus et sauvegardée dans un dossier correspondant à la lettre concernée.

Pour la phase d'entraînement, nous avons implémenté un réseau de neurones convolutif (CNN) avec TensorFlow/Keras. Notre architecture se compose de trois blocs de convolution suivis de couches denses. Le premier bloc utilise une couche de convolution avec 64 filtres, suivie d'une couche de pooling et d'un dropout de 0.3. Le deuxième bloc augmente la complexité avec 128 filtres, toujours suivi de pooling et dropout. Le troisième bloc utilise 256 filtres avec la même structure. Ces blocs sont suivis par une succession de couches denses comprenant d'abord 512 neurones, puis 256, chacune avec un dropout de 0.5, avant la couche finale de 26 neurones correspondant aux lettres de l'alphabet.

Le modèle est entraîné sur 70% des données, les 30% restants servant à la validation. Nous utilisons l'optimiseur Adam avec un taux d'apprentissage initial de 0.001. Deux mécanismes d'optimisation sont mis en place : un réducteur de taux d'apprentissage qui diminue celui-ci de 20% après 2 époques sans amélioration, et un système d'arrêt précoce qui stoppe l'entraînement si aucune amélioration n'est constatée pendant 3 époques.

L'application finale fonctionne en temps réel en combinant ces différents éléments. Le système utilise la même chaîne de traitement d'image que celle développée pour la création du dataset, assurant ainsi une cohérence entre les données d'entraînement et les conditions réelles d'utilisation. Les images de la main, une fois isolées et prétraitées, sont redimensionnées en 64x64 pixels, converties en RGB et normalisées selon le standard VGG16 avant d'être soumises au modèle pour prédiction. Pour faciliter l'utilisation, le système affiche simultanément la lettre reconnue et une image de référence de l'alphabet des signes, permettant à l'utilisateur de valider et corriger ses gestes si nécessaire.

II. Justification des choix de design

Notre approche finale pour la détection de la main résulte d'une évolution progressive de nos expérimentations. Initialement, nous avions exploré une approche basée uniquement sur la détection des caractéristiques géométriques utilisant des seuils HSV pour segmenter les zones de peau, suivie d'un filtre de Canny pour la détection des contours. La distinction entre la main et le visage s'effectuait par détection de pics correspondant aux doigts et par analyse d'une forme "étoilée" caractéristique. Bien que cette approche était intéressante car purement géométrique, elle s'est révélée trop sensible aux conditions d'éclairage et à la position de la main.

Le choix d'une région d'intérêt (ROI) fixe constitue notre première décision majeure de design. Une ROI dynamique aurait offert plus de flexibilité mais aurait significativement complexifié la détection et augmenté le temps de traitement. La taille et la position de la ROI ont été choisies pour permettre une gamme de mouvements naturels de la main tout en minimisant les interférences avec d'autres parties du corps.

Nous avons alors considéré plusieurs alternatives classiques de vision numérique. Les modèles de mixture gaussienne nécessitaient un entraînement préalable, tandis que la détection de contours par Sobel ou Prewitt se révélait moins efficace que Canny. La technique de soustraction d'arrière-plan dynamique s'est finalement imposée comme un bon compromis, permettant une adaptation automatique aux changements d'illumination. Le choix de 60 frames pour l'initialisation et du poids d'accumulation de 0.5 permet une adaptation rapide aux changements graduels tout en évitant une sensibilité excessive aux variations brutales.

La chaîne de prétraitement a été optimisée par une conversion en niveaux de gris et l'application d'un flou gaussien avec un noyau 9x9, choisi empiriquement comme compromis entre réduction du bruit et préservation des contours. Le seuil de binarisation de 25 pour la détection du mouvement résulte également d'une analyse empirique : des valeurs plus faibles (10-15) généraient trop de faux positifs, tandis que des valeurs plus élevées (35-40) manquaient les mouvements subtils des doigts.

Pour la partie reconnaissance, nous avons privilégié une architecture CNN légère plutôt que des modèles complexes comme ResNet. Notre architecture progressive permet des performances compatibles avec une utilisation en temps réel, et le choix de créer notre propre dataset assure une cohérence avec notre méthode de détection.

III. Résultats

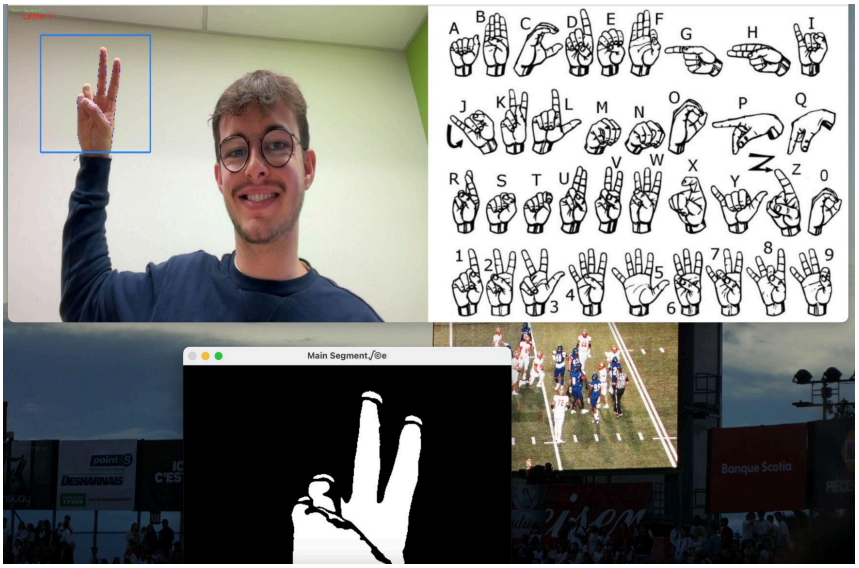
Concernant les résultats d'entraînement du modèle, nous avons une accuracy de l'ensemble de test de **1.00.** Notre précision est particulièrement élevée étant donné que nos données de test et d'entraînement sont capturées simultanément. Les lettres P,Q,U,V sont les seules n'ayant pas un score parfait. Nous pouvons expliquer cela par la similarité de entre P et Q, et celle de U et V.

	precision	recall	f1-score	support
A	1.00	1.00	1.00	58
B	1.00	1.00	1.00	65
C	1.00	1.00	1.00	55
D	1.00	1.00	1.00	57
E	1.00	1.00	1.00	56
F	1.00	1.00	1.00	56
G	1.00	1.00	1.00	60
H	1.00	1.00	1.00	83
I	1.00	1.00	1.00	74
J	1.00	1.00	1.00	67
K	1.00	1.00	1.00	51
L	1.00	1.00	1.00	55
M	1.00	1.00	1.00	64
N	1.00	1.00	1.00	65
O	1.00	1.00	1.00	59
P	0.98	1.00	0.99	56
Q	0.99	1.00	0.99	68
R	1.00	1.00	1.00	65
S	1.00	1.00	1.00	60
T	1.00	1.00	1.00	65
U	0.98	0.96	0.97	54
V	0.98	0.97	0.98	61
W	1.00	1.00	1.00	59
X	1.00	1.00	1.00	62
Y	1.00	1.00	1.00	56
Z	1.00	1.00	1.00	69
accuracy			1.00	1600
macro avg	1.00	1.00	1.00	1600
weighted avg	1.00	1.00	1.00	1600

Concernant l'arrière-plan, les résultats demeurent très dépendant de la capture de l'arrière-plan initial que ce soit lors de la capture du dataset ou à la reconnaissance, s'il n'est pas uni ou s'il y a des coins ou reflets apparents cela biaise la reconnaissance.

Lors de la capture du dataset certaines lettres sont plus compliquées comme le J ou le Z qui nécessitent un mouvement de main dans l'alphabet de langue des signes, ce qui biaise l'utilisation d'une personne malentendante de notre système.

Concernant l'utilisation en tant que tel du modèle entraîné sur nos données, la plupart des lettres fonctionnent correctement. Voici un exemple avec la lettre V.



D’autres présentent encore des ambiguïtés. Nous observons que les lettres M, N, S et T qui sont des lettres à poing fermé représentent un défi pour notre modèle. Cela peut s’expliquer par la difficulté de détecter des contours bien distincts des différents doigts surtout lorsque le doigt doit être placé devant le point comme pour la lettre S. Il en résulte que les images en contours de ces lettres sont très similaires et se rapprochent de la forme d’un poing classique, et donc de cette lettre S. Peut-être qu’avec beaucoup plus d’images et des images diversifiées le modèle sera finalement capable de faire la distinction.

Tout le processus d’utilisation du système est illustré dans une vidéo accessible via ce lien: **https://youtu.be/42tqg14p-u0**

Cette dernière comprend des exemples pour la capture des données, l’entraînement du modèle et l’application de celui-ci en temps réel.

IV. Améliorations futures

Notre solution pourrait bénéficier de plusieurs améliorations significatives. La première serait l'optimisation de notre système de détection par soustraction d'arrière-plan. En effet, bien que cette méthode soit efficace dans des conditions contrôlées, elle reste sensible aux changements brusques d'illumination. L'implémentation d'une mise à jour adaptative du taux d'accumulation, plutôt qu'un taux fixe de 0.5, permettrait une meilleure adaptation aux variations de l'environnement.

Une deuxième amélioration concernerait la région d'intérêt (ROI). Notre solution actuelle utilise une ROI fixe qui, bien que fonctionnelle, limite la liberté de mouvement

de l'utilisateur. L'implémentation d'une ROI dynamique, capable de suivre automatiquement la main tout en maintenant la stabilité de la détection, améliorerait significativement l'expérience utilisateur.

Une troisième amélioration serait d'étendre notre système pour gérer plusieurs échelles de détection. Actuellement optimisé pour une distance spécifique entre la main et la caméra, le système pourrait être amélioré pour s'adapter automatiquement à différentes distances et angles de vue, augmentant ainsi sa robustesse dans des conditions d'utilisation plus variées.

Enfin, la plus importante amélioration serait d'étendre notre système à la reconnaissance complète de la langue des signes, au-delà du simple alphabet. Cela impliquerait la détection simultanée des deux mains et du haut du corps, ainsi que l'élargissement significatif du dictionnaire pour inclure les mots et expressions courantes. Cette évolution nécessiterait une adaptation de notre méthode de soustraction d'arrière-plan pour gérer efficacement plusieurs zones d'intérêt mobiles, ainsi qu'une refonte de notre architecture de reconnaissance pour prendre en compte la dimension temporelle des signes.