Reconocimiento de Lengua de Signos

Javier Esmoris, Oriol Marión, Serena Sánchez

Resumen—
En este artículo, se presenta un proyecto de reconocimiento de la Lengua de Signos con el objetivo de facilitar la comunicación entre personas que hablan de forma oral y aquellas que se comunican exclusivamente mediante la Lengua de Signos. Este estudio consiste en desarrollar un algoritmo capaz de identificar letras estáticas de la Lengua de Signos y traducirlas a texto. Este proceso implica la recopilación de un conjunto representativo de imágenes para entrenar el modelo, la distinción de la mano del fondo en las imágenes y la identificación de la letra que se está gesticulando. Para llevar a cabo este desarrollo se han usado desde técnicas tradicionales de preprocesamiento de imágenes como la binarización de la imagen mediante un threshold, morfología binaria, geometría de una región y HoG que nos han proporcionado los descriptores de características necesarios para entrenar modelos clásicos como el KNN y el SVM, hasta redes neuronales convolucionales (CNN) y la librería en tendencia de Google Mediapipe. Mientras que los primeros resultados del proyecto muestran una precisión insatisfactoria siendo prácticamente un modelo que predice al azar, los últimos avances obtenidos por la red neuronal han sido altamente prometedores. A través de este proyecto, hemos combinado técnicas tradicionales de procesamiento de imágenes con enfoques más avanzados, para mejorar la precisión del reconocimiento de la Lengua de Signos incluso a tiempo real.

Palabras clave—Descriptor de características, morfología binaria, red neuronal, HoG.

1 Introducción

"Según la reciente Encuesta del INE, 27.300 personas emplean la Lengua de Signos (2'2%)". Esto quiere decir que hay un 97'8% de personas que no emplean esta lengua y muy probablemente tampoco sean capaces de entenderla. Por este motivo, nuestro proyecto se basa en facilitar la comunicación entre personas que tengan la capacidad de hablar de forma oral y las personas que solo puedan comunicarse a través de la Lengua de Signos. Somos muchos los que no seríamos capaces de mantener una conversación de forma eficiente ni comprensible con personas que no hablan una lengua que dominamos. Gracias a la tecnología, podemos usar una herramienta como el traductor para esta comunicación. Pero, ¿que ocurre cuando esta se basa en gestos? Este es el contexto en el que encontramos la Lengua de Signos. Con el Reconocedor de Lengua de Signos que se ha implementado para este proyecto conseguiremos poner fin a este problema. El objetivo principal es crear un algoritmo capaz de identificar diferentes letras estáticas de la Lengua de Signos y ponerlos en forma de texto. Para llevar esto a cabo debemos: establecer un conjunto de imágenes suficientemente representativo para entrenar un modelo, ser capaces de, primero, distinguir una mano de un fondo y, por último, identificar que letra es la que se está gesticulando en la imagen.

2 Estado del arte

2.1 Técnicas avanzadas de visión por computador

El reconocimiento de lenguaje de señas mediante visión por computador es un campo de investigación en constante evolución. Los métodos más recientes utilizan técnicas de Deep Learning para extraer características de las imágenes de las manos y de los gestos, con el objetivo de detectar y clasificar los signos realizados por una persona sorda.

Entre los métodos más utilizados se encuentran las redes neuronales convolucionales (CNN)[4], que han demostrado ser muy efectivas en la clasificación de imágenes. En particular, algunas arquitecturas como las redes convolucionales 3D (C3D) [2] y las redes convolucionales espaciales-temporales (STCNN) han mostrado buenos resultados en la detección de gestos de lenguaje de señas.

Además, se han propuesto diferentes enfoques para el preprocesamiento de las imágenes, como la normalización de la iluminación, la segmentación de la mano, el uso de técnicas de aumento de datos y la combinación de múltiples vistas de la mano.

2.2 Técnicas de visión simples y clasificadores tradicionales

El reconocimiento de lenguaje de señas utilizando técnicas de visión por computador simples se basa en el procesamiento de características geométricas y de textura de la mano y los gestos realizados. Las características geométricas incluyen la forma, el tamaño y la posición de la mano, mientras que las características de textura incluyen la intensidad de los píxeles y el contraste.

Entre las técnicas de preprocesamiento de imágenes utilizadas se encuentran la normalización de la iluminación, la segmentación de la mano, el filtrado y la eliminación de ruido. Además, se han propuesto diferentes enfoques para la extracción de características, como el cálculo de histogramas de gradientes orientados (HOG), características basadas en la forma (como el contorno o la convexidad), características basadas en la textura (como los descriptores de co-ocurrencia de la textura) y características basadas en la posición y movimiento de la mano.

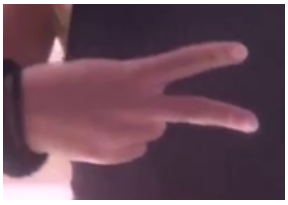
En cuanto a los clasificadores, los métodos más utilizados son el k-NN (k-Nearest Neighbours), SVM (Support Vector Machine) [1] [6], Naïve Bayes [5] y árboles de decisión. Estos métodos se utilizan para clasificar los gestos en diferentes categorías correspondientes a las palabras del lenguaje de señas.

3 Propuesta

Para crear el dataset que usaremos en nuestro algoritmo grabaremos nuestros propios vídeos. Para empezar, grabaremos 3 símbolos que sean diferenciables entre ellos. Para cada símbolo se grabarán algunos vídeos dónde se puedan conseguir frames a diferentes distancias en un fondo monocromático. De esta forma, tendremos muchos ejemplos de un símbolo en concreto para poder entrenar de forma correcta el modelo. Los 3 símbolos con los que trataremos las soluciones Old-School se muestran en las

siguientes imágenes y corresponden a las etiquetas: 'positivo', 'negativo' y 'tijeras'.

(a) Positivo (b) Negativo



(c) Tijeras

Fig. 1: Símbolos usadas en la solución Old-School

Una vez sepamos como de bien podemos reconocer estos 3 símbolos y que tipo de imágenes nos funcionan mejor, usaremos un dataset con las letras del alfabeto de signos americano. Cabe destacar que únicamente nos centraremos en aquellas letras del alfabeto de signos que no tienen movimiento, así que la letra J y Z quedarán fuera de nuestro algoritmo.

Cuando tengamos una base de datos que contenga suficientes casos como para poder hacer un reconocimiento preciso en los diferentes entornos en los que se espera que se utilice nuestro algoritmo, empezaremos a tratar las imágenes que nos lleguen de la cámara. Este tratamiento empezará con la identificación de una ventana que incluya la mano que nos interesa. Una vez tenemos ubicada la mano, será más fácil usar esa información más concreta para identificar todas aquellas características que nos va a dar información sobre el gesto que está realizando. Para el reconocedor de signos se nos piden dos métodos distintos: uno 'old school' y otro con técnicas más actuales. Para el primer caso, usaremos dos métodos distintos de creación del vector de características de la mano y sus descriptores y con estos aplicaremos algunos modelos de machine learning destinados a clasificación como: KNN y SVM. Para el segundo caso, se usarán modelos más complejos de deep learning como redes neuronales o librerías más completas como mediapipe.

Para medir el rendimiento de nuestro algoritmo utilizaremos diferentes métricas: Accuracy para medir el porcentaje de casos en los que el modelo acierta, para esta métrica nos aseguraremos que las clases estén balanceadas para que sea fiable i F1-score que combina: precisión, que determina la calidad de la predicción, i recall, que indica la cantidad de imágenes que el modelo ha sido capaz de identificar correctamente.

4 Experimentos, resultados i análisis

El proceso de desarrollo de este proyecto se ha basado en un objetivo de procesamiento incremental. Esto quiere decir que se han usado 4 procesamientos distintos a la base de datos empezando por uno sencillo basado en detección de regiones y finalizando con librerías de deep learning ya creadas con el objetivo de reconocer manos y gestos.

A continuación, se describirán las cuatro pruebas realizadas con el dataset, detallando los algoritmos utilizados, los resultados obtenidos y las expectativas asociadas a cada una de ellas.

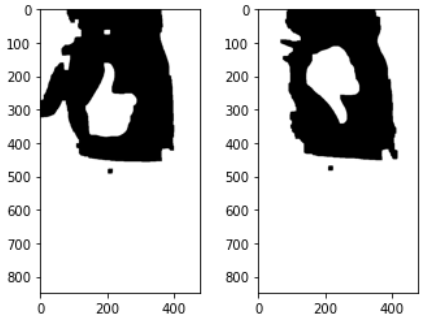
4.1 Método 1 Old-School

En la primera aproximación al algoritmo de reconocimiento de signos se ha usado el concepto de detección de regiones. En este caso, partimos de casi 100 imágenes como las que se muestran a continuación teniendo en cuenta variación de distancia a la cámara.

(a) Positivo (b) Negativo

Fig. 2: Imágenes prueba 1 Old-School

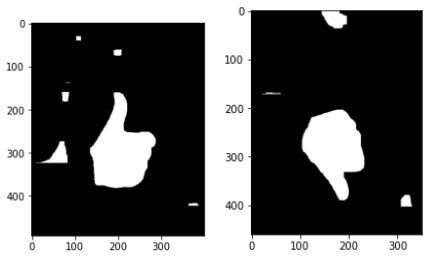
El objetivo con estas imágenes es conseguir la figura de la mano en blanco y el resto del fondo en negro para poder usar rasgos de esa región para construir el primer vector de características de cada imagen. Los pasos que se han seguido para conseguir la imagen binaria objetivo son: aplicación de un filtro de desenfoque para suavizar la imagen y eliminar pequeños ruidos que aparezcan, después, una binarización mediante un umbral general dónde a los píxeles más oscuros se les asigna el negro, para formar parte del fondo y a los más claros el blanco, nuestra región de interés. Este procesamiento es concreto para nuestras imágenes ya que se ha usado una camiseta negra detrás de la mano y un coletero negro que corta también la zona de interés consiguiendo como resultado una región blanca formada por la mano. Para eliminar imperfecciones en la representación de la mano se han aplicado operaciones morfológicas binarias tales como erosión y dilatación. La siguientes imágenes ilustran el resultado obtenido tras la aplicación de estas técnicas:



(a) Positivo (b) Negativo

Fig. 3: Imágenes procesadas Old-School

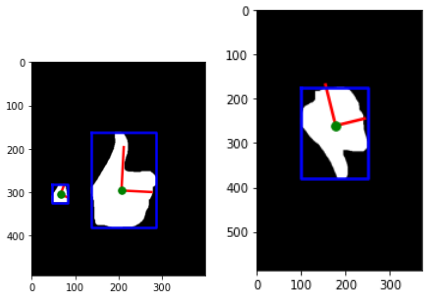
Como el objetivo es conseguir una sola región blanca, una vez tenemos las imágenes de esta forma hemos calculado dónde aparece el primer píxel negro y el último en las filas y en las columnas de la imagen. Así conseguimos una imagen con una zona blanca más concreta que nos facilita la detección de la mano. Por último, para obtener únicamente la mano en blanco, se ha utilizado un algoritmo de flood fill. La imagen adjunta a continuación muestra el resultado obtenido al aplicar el recorte y el algoritmo de flood fill.



(a) Positivo (b) Negativo

Fig. 4: Imágenes procesadas Old-School

Una vez ya tenemos una región blanca principal usamos la función region props de la librería skimage para detectar la región blanca más grande y guardar las medidas correspondientes al área de la zona blanca, el área de la caja en la que está cubierta y el ratio (altura dividida entre el ancho de la mano). Estos tres datos conformarán el vector de características. De forma visual las medidas que se han recogido se muestran en las siguientes imágenes:



(a) Positivo (b) Negativo

Fig. 5: Imágenes procesadas Old-School

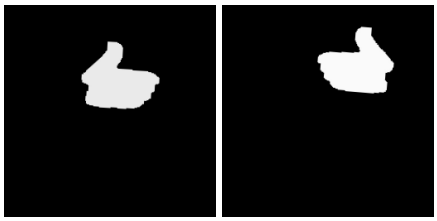
Cuando ya hemos conseguido estas medidas para todas las imágenes, usamos los clasificadores KNN y SVM para construir un modelo capaz de reconocer correctamente los 3 símbolos comentados anteriormente. Con esto obtenemos un accuracy de: 0.52 y un f1-score de: 0.52. Con este procesamiento y teniendo en cuenta los símbolos utilizados, podemos ver que mientras la clase tijeras se identifica correctamente, entre las otras dos clases restantes el clasificador tiende a confundirse ya que tanto las medidas de la caja como el ratio son muy similares. Como este procesamiento esta implementado paso a paso con funciones que conforman las bases del preprocesamiento, los resultados iniciales han sido muy mejorables. Cabe destacar también, que su precisión y capacidad para reconocer gestos sigue limitada debido a que solo se detectarían correctamente los signos cuyas imágenes de origen tuviesen un formato similar a las mostradas y siempre y cuando el tono de piel de la persona sea claro y la mano este situada delante de un fondo negro.

4.2 Método 2 Old-School

Con la prueba anterior los resultados de los clasificadores no han sido demasiado satisfactorios, pero sí hemos conseguido una ventana binaria con la forma de la mano suficientemente precisa como para extraer un vector de características diferente al anterior, esperando de este unos mejores resultados.

En este segundo método, se han probado dos técnicas diferentes consideradas más avanzadas dentro del campo del procesamiento de imágenes. En primer lugar, se probo el ORB (Oriented FAST and rotated BRIEF) que consiste en un detector y descriptor de características que tiene como objetivo ser una alternativa más rápida y eficiente a algoritmos como SIFT y SURF. Aun así, esta herramienta nos dio problemas debido a que para cada imagen devolvía un descriptor de características de distinto tamaño para cada imagen. Esto es un problema para los clasificadores que queríamos entrenar ya que necesitan que la información de cada muestra tenga la misma medida. Como solución a este problema, decidimos utilizar el HoG puesto que se ha visto que da mejores resultados que otros descriptores como el HAAR [3].

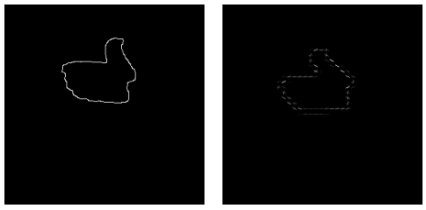
En primer lugar, se ha hecho una limpieza de aquellas imágenes cuyo preprocesamiento (método 1) era poco exacto y, por lo tanto, sería un problema usarlo para esta prueba. En segundo lugar, y con el objetivo de incrementar la diversidad del conjunto de datos de entrenamiento y mejorar la capacidad de generalización de los modelos, se ha aplicado la técnica de Data Augmentation. En nuestro caso, con esta técnica hemos generado nuevas imágenes mediante al rotación, translación y inversión horizontal. Algunos ejemplos serían los siguientes:



(a) Imagen original (b) Imagen modificada

Fig. 6: Ejemplo de data augmentation con inversión horizontal y translación

Una vez aumentado el conjunto de entrenamiento, la metodología que se ha seguido consiste en tres pasos principales: segmentación, extracción de descriptores puntuales y clasificación. En las imágenes monocromáticas se ha usado el detector de contornos de Canny para generar imágenes que contengan sólo los contornos más fuertes, como se muestra en la figura 7a. De estas imágenes de contornos se extrae su histograma de orientaciones del gradiente mostrado en la figura 7b. Con este método se pretende conseguir características distintivas de cada símbolo que mejoren el rendimiento de los clasificadores.



(a) Contorno Detectados (b) Descriptores de la imágen

Fig. 7: Proceso de extracción del vector de características

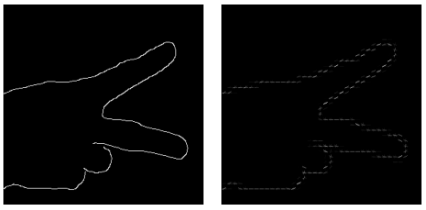
Con esta técnica y aplicando al vector un cambio de dimensión para poder entrenarlo con un clasificador, hemos obtenido unos resultados de 0.77 de accuracy y 0.77 de F1-score para el KNN y un 0.83 de ambas métricas para el SVM. Si nos basamos en las cifras de las métricas, esta prueba ha mostrado una mejora significativa en la precisión y la capacidad de reconocimiento de signos en comparación con el primer método.

En este punto del proyecto, las imágenes con las que contábamos antes del data augmentation eran bastante pocas y, haciendo limpieza de las que el procesamiento no era el óptimo, aún contábamos con menos imágenes para hacer un buen entreno de un modelo. Además, el hecho de tratar cada imagen y aplicarle todos los cambios necesarios requería de mucho tiempo de ejecución. Para que el hecho de tener que encontrar la mano dentro de la imagen no interfiriese en el objetivo del proyecto de reconocer los signos, se ha decidido volver a usar las técnicas de data augmentation y HoG pero esta vez con un dataset diferente. Este nuevo conjunto trata de imágenes a color tomadas a diferentes personas donde la mano esta en primer plano. De esta forma, el paso de preprocesamiento para encontrar la ventana es inexistente, mejorando el tiempo del algoritmo. Las imágenes generadas a partir del Data Augmentation han quedado de la siguiente forma:

(a) Imagen original (b) Imagen modificada

Fig. 8: Inversión, translación y modificación de brillo en la imagen original

Partiendo de imágenes similares, aplicamos una conversión a escala de grises, una ecualización para mejorar el contraste, detección de contornos de Canny y finalmente HoG para ver qué tipo de imágenes conseguimos y si los resultados se asemejan a los obtenidos anteriormente.



(a) Contorno Detectados (b) Descriptores de la imágen

Fig. 9: Proceso de extracción del vector de características

En este caso, el algoritmo KNN consigue un accuracy y un F1-score de 0.71. En cambio, el SVM alcanza un accuracy de 0.98 y un F1-score de 0.98. En el caso del segundo modelo de aprendizaje, hemos conseguido mejorar el tiempo y la precisión. Aun así, es cierto que la partición en train y test se ha hecho de forma aleatoria y, al tratarse de frames de un vídeo, una imagen del train puede ser prácticamente igual que la imagen del test, así que estaríamos entrenando el modelo con el conjunto de entrenamiento, siendo esta una práctica poco fiable de entrenamiento.

Para solucionar este problema, se ha decidido ejecutar el mismo algoritmo pero separando las manos de una persona del resto de manos de otras personas, usando la primera como conjunto de entrenamiento. Como se ha comentado, la precisión de ambos modelos de aprendizaje baja, KNN hasta un 0.33 y SVM hasta un 0.46.

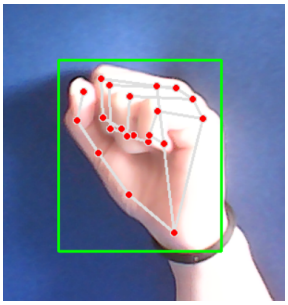
4.3 Redes neuronales

En la tercera prueba, quisimos intentar hacer un modelo más complejo mediante los conocimientos adquiridos en redes neuronales. Por lo tanto, el siguiente paso en nuestro algoritmo ha sido entrenar una red neuronal para que reconociera los símbolos de nuestro propio dataset aumentado. Hicimos un primer intento en pytorch pero cambiamos a tensorflow por la facilidad que ofrece. El input de esta red es una imagen a color, es decir, con tres canales representada por un numpy array de dimensiones (256,256,3). Las siguientes capas vienen en tres bloques de dos convoluciones y una reducción de dimensionalidad a la mitad. De esta forma en cada bloque reducimos en un cuarto la información, pero duplicamos el número de kernels. Para acabarla, añadimos una capa que aplane el resultado y otra capa densa con 3 neuronas, una para cada clase. Esta red convolucional se ha entrenado de forma bastante rápida en diez épocas, dando un accuracy del 96,77% sobre el test.

Este tipo de técnica de deep learning ya está preparada para entrenar con una cantidad de datos más grandes y sin necesidad de un preprocesamiento tan especifico. Así que, teniendo en cuenta los resultados de este tercer método, consideramos que era el momento de intentar reconocer signos en mayor medida, es decir, mediante las letras del alfabeto. Para llevar esto a cabo, hemos entrenado otra red convolucional con el dataset de lengua de signos de MNIST. La estructura de esta es igual que la anterior, cambiando la entrada a imágenes a 28x28 en escala de grises, además de hacer que las convoluciones devuelvan imágenes del mismo tamaño, ya que si no nos quedaríamos sin imagen antes de llegar al final. La salida es una capa densa con 24 neuronas, de nuevo una para cada clase. Hemos alcanzado un accuracy del 85,3% de nuevo en diez épocas. Este último modelo que ha dado los mejores resultados hasta el momento, lo hemos guardado y lo usaremos en la última prueba realizada.

4.4 Integración de Mediapipe

Para finalizar con nuestro proyecto, se ha querido integrar la librería de código abierto basada en aprendizaje profundo de Google: Mediapipe. De forma general, esta herramienta se usa en múltiples casos, por ejemplo, en la detección de objetos, seguimiento facial, reconocimiento de voz... En nuestro caso, la usaremos para el reconocimiento de la mano a tiempo real. Para esto, se usa la cámara del dispositivo dónde se este ejecutando el algoritmo. Gracias a esta librería se consigue un procesamiento como el que podemos ver en la siguiente imagen dónde se nos señala a la perfección nuestra mano y puntos característicos del símbolo que estamos representando.



(a) Signo

```
1/1 [==============================] - 0s 20ms/step
[0]
Sign Detected: [0]==>A
```

(b) Predicción

Fig. 10: Detección de la letra A

Esta información es la que pasamos al modelo de red neuronal explicado en el apartado anterior y como resultado recibimos la letra que el algoritmo ha detectado como más probable. Además, esta librería nos permite variar el fondo y la distancia a la cámara del gesto, ya que es capaz de detectar siempre dónde se encuentra la mano. Aun así, para nuestro algoritmo lo óptimo es que solo se enseñe una mano para poder entrenar de forma correcta la red neuronal.

Gracias a estas librería y el modelo de red neuronal implementado, se ha podido conseguir el objetivo final del proyecto con unos resultados muy buenos y con posible aplicación en el mundo real.

5 Conclusiones

El enfoque de procesamiento incremental utilizado en este proyecto nos ha permitido ir mejorando gradualmente los resultados a medida que se han aplicado técnicas más avanzadas de procesamiento de imágenes y aprendizaje automático.

A continuación, se resumirán los principales logros y conclusiones de cada prueba realizada. En las etapas iniciales, donde se buscaba un enfoque más tradicional, se ha conseguido obtener de forma clara la ventana dónde se encontraba la mano y utilizar esta como región de interés y herramienta para conseguir los descriptores de características de cada símbolo. Este método, aún habiéndolo reducido al reconocimiento de 3 símbolos, ha resultado tener muchas limitaciones y servir exclusivamente para un tipo de imágenes captadas de forma muy concreta. Siguiendo con los métodos más clásicos, se ha intentado mejorar los resultados obtenidos aumentando el conjunto de datos y escogiendo un vector de características basado en el HoG. Además, para este caso, se emplearon dos datasets diferentes, llegando a la conclusión que el preprocesamiento inicial sí servía para obtener unos mejores resultados al entrenar los modelos. En general, cabe destacar que el algoritmo de clasificación SVM siempre ha obtenido una precisión más alta que el KNN. Aun así, ninguno de los dos modelos obtuvo un resultado significativamente bueno.

En las etapas más avanzadas del proyecto, se dio un salto al aprendizaje profundo enfocándonos en el entrenamiento de una red neuronal. Los resultados obtenidos con el dataset de 3 símbolos creado por nosotros mismos fueron mucho mejores, por lo que, ya consideramos que el proyecto estaba suficientemente madurado como para considerar entrenar la red neuronal con un conjunto de imágenes que tuviese todas las letras del abecedario cuyos gestos fuesen estáticos. Los resultados obtenidos en este caso fueron notablemente buenos, ya que predecir correctamente 24 símbolos es difícil. Por último, y para dar por finalizado el proyecto, se integró la librería Mediapipe con el último modelo de red neuronal entrenado, lo que nos ha permitido capturar y procesar los datos sobre los gestos en tiempo real. Esta integración ha supuesto el culmen del proyecto puesto que hemos podido probar el modelo de manera parecida a la de un caso real.

En definitiva, cada uno de los avances explicados anteriormente nos ha permitido acércanos cada vez más a una herramienta útil dando lugar a un progreso continuo hacia el objetivo final: el reconocimiento de la lengua de signos.

Referències

[1] Xinran Bai, Chen Li, Lihua Tian, and Hui Song. Dynamic hand gesture recognition based on depth information. In *2018 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 91–100, 2018.

[2] Jie Huang, Wengang Zhou, Houqiang Li, and Weiping Li. Sign language recognition using 3d convolutional neural networks. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 283–294, 2015.

[3] Pablo Negri, Xavier Clady, and Lionel Prevost. Benchmarking haar and histograms of oriented gradients features applied to vehicle detection. In *ICINCO-RA (1)*, pages 359–364, 2007.

[4] Razieh Rastgoo, Kourosh Kiani, and Sergio Escalera. Hand sign language recognition using multi-view hand skeleton. *Expert Systems with Applications*, 150:113336, 2020.

[5] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *Proceedings of International Symposium on Computer Vision - ISCV*, pages 265–271, 1995.

[6] Hee-Deok Yang and Seong-Whan Lee. Robust sign language recognition by combining manual and non-manual features based on conditional random field and support vector machine. *Pattern Recognition Letters*, 34(16):2051–2056, 2013.