"Aplicación para dispositivos móviles que permite la traducciónde la lengua de señas mexicana al alfabeto en tiempo reamediante el procesamiento de imágenes usando el método de Haar-like Features"

Autor: GERMAN RODRIGUEZ BERMUDEZ

Tesis presentada para obtener el título de:
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

Nombre del asesor:
MCC. EDSSON MADRIGAL RICO

UNIVERSIDAD VASCO DE QUIROGA

FACULTAD DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

“Aplicación para dispositivos móviles que permite la traducción de la lengua de señas mexicana al alfabeto en tiempo real mediante el procesamiento de imágenes usando el método de Haar-like Features”

TESIS

PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

GERMAN RODRIGUEZ BERMUDEZ

ASESOR

MCC. EDSSON MADRIGAL RICO

CLAVE:
16PSU0049F

ACUERDO:
MAES091101

MORELIA, MICHOACÁN

Septiembre-2018

Dedicatoria

Como dijo Isaac Newton "Si he visto más lejos es porque estoy sentado a hombros de gigantes" … Agradezco a mi familia, amigos y maestros por ser mis guías, por ser parte de mi vida y acompañarme en este proyecto, sin ustedes no lo habría logrado,

Dedico este documento a cada una de las personas que he tenido la fortuna de conocer, ustedes son mi ejemplo y motivación. La búsqueda de respuestas, lo que soy y lo poco que he aprendido, es gracias a ustedes.

A ti, gracias por acompañarme en este camino y por hacer interesante este viaje…

Gracias totales.

ÍNDICE GENERAL

RESUMEN

Se propone una aplicación móvil que permite la traducción de la Lengua de Señas Mexicana (LSM) a texto y voz en tiempo real, con la finalidad de apoyar y asistir a personas que padecen sordera o que no pueden comunicarse verbalmente, pero conocen y usan dicha lengua de señas. Esta aplicación utiliza la cámara del dispositivo móvil para capturar las imágenes, mismas que después procesa para reconocimiento de la seña mediante clasificadores Haar-like features; da como resultado el texto de la o las señas que le corresponden y además se incluye un botón para reproducir dicho texto con voz artificial en el mismo dispositivo. Se desarrolló una aplicación móvil para dispositivos móviles que utilizan el Sistema Operativo (SO) Android. En México no se tiene registro de una aplicación como la mencionada; no obstante en otros países ya se tienen proyectos específicos para la lengua de señas correspondiente. La innovación tecnológica de este proyecto reside en el procesamiento en tiempo real de las capturas de las señas por medio de la cámara del dispositivo para hacer la traducción al texto correspondiente y hacer la “lectura” en voz alta de esta traducción.

ABSTRACT

A mobile application is proposed that allows the translation of the Mexican Sign Language (LSM) into text and voice in real time, in order to support and assist people who suffer from deafness or who cannot communicate verbally, but know and use this sign language. This application uses the camera of the mobile device to capture the images, which are then processed for sign recognition using Haar-like features classifiers; it displays as a result the text of the corresponding sign(s) and also includes a button to reproduce the text with artificial voice on the device itself. The mobile application was developed for mobile devices using the Android Operating System (OS). In Mexico there is no record of such an application; however, in other countries there are already specific projects for the corresponding sign language. The technological innovation of this project lies in the real-time processing of the captures of the signs by means of the camera of the device to make the translation to the corresponding text and to do the "reading" aloud of this translation.

PLANTEAMIENTO DEL PROBLEMA

Desarrollar una aplicación de comunicación para dispositivos móviles, diseñada para personas con sordera y que utilizan la lengua de señas mexicana con el objetivo de comunicarse con las personas que no conocen o usan dicha lengua.

ANTECEDENTES

En los últimos años el uso de los dispositivos móviles ha crecido a un ritmo vertiginoso, existen varias marcas en el mercado de estos aparatos con software tanto bajo licencia como de código abierto y a la vez estas herramientas tienen una mayor capacidad de procesamiento. Actualmente el uso de multimedia está en la gran mayoría de dispositivos móviles gracias al desarrollo de Sistemas Operativos y lenguajes de programación, mismos que permiten el desarrollo de aplicaciones móviles (app o apps) más complejas y robustas.

Uno de los usos actuales más importantes es el desarrollo de aplicaciones móviles para apoyo a necesidades sociales, por ejemplo, usuarios que han perdido la capacidad auditiva, la pérdida de visión entre otras; estas situaciones representan un área en desarrollo y una utilización responsable de las aplicaciones y dispositivos portatiles. A. Koon y De la Vega mencionan que la tecnología asistiva o adaptativa puede llegar a reducir el impacto de la discapacidad y satisfacer el derecho de la calidad de vida de las personas con capacidades especiales y asimismo llegar a influenciar la economía y bienestar en sus familias [1].

La lengua de señas es una de las principales herramientas que permiten a personas que perdieron su capacidad auditiva, comunicarse con los demás. Es la lengua que utilizan las personas sordas y como toda lengua, posee su propia sintaxis, gramática y léxico [2].

Han habido varios proyectos, tanto en sitios web como aplicaciones móviles para usuarios sordos. En Argentina El servicio Web “Lengua de Señas Argentina” (LSA) administra un diccionario generado a partir de un conjunto de videos que se encuentran ordenados en distintas categorías: Alfabeto, Animales, Colores, Comidas, Cosas, Frases, Números, Personas y Transportes. De este modo, el servicio está conectado a una base de datos para administrar diferentes funciones referidas al manejo de los archivos de videos [3].

De acuerdo con Acevedo, Flores, Lima, y Alducin [4] se logró el desarrollo una aplicación móvil cuya intención es facilitar la comunicación entre personas sordomudas y también entre sordomudos y personas sin esta discapacidad. Su funcionamiento es simple, la aplicación traduce palabras de la LSM (imágenes de señas) a texto y viceversa a través de un diccionario construido para tal fin. Estas imágenes son predefinidas lo que limita esta aplicación. Otro ejemplo es la aplicación “ProDeaf Traductor” o “Hand Talk”, que permiten introducir un texto (portugues) para posteriormente generar una animación en lenguaje de señas como traducción a Libras (Lengua de señas de Brasil).

Sobre el proceso de traducción de lengua de señas a partir de imágenes hay varios proyectos de software que han diseñado diversas opciones de traducción, reconocimiento y procesamiento. Por ejemplo Medhi, Doshi, Pawar, Kesarkar y Dalvi en [5] proponen un sistema que reconoce gestos dinámicos y estáticos, Para los gestos estáticos se utiliza el método “momentos de Zernike” y para los dinámicos (2 segundos de video) extraen un vector de características en curva que muestra una alta precisión en rutas de identificación única del movimiento, luego estas rutas o patrones son clasificados usando Máquinas de Soporte Vectorial (SVM).

También Chiguano, Moreno y Corrales propusierón un software donde, para el procesamiento digital de imágenes se aplicaron algunos filtros y operaciones morfológicas para resaltar las características de la imagen y eliminar información innecesaria como ruido. También se eliminaron objetos extraños en la imagen mediante un recortado del área de interés. Con la ayuda del Vision Assitant de Labview, se elaboraron las bases de datos, para llevar a cabo la comparación con la imagen recortada y de esta manera asignar la clase (letra) correspondiente a cada imagen. Con la clase asignada se forma el texto que se muestra en forma escrita en la pantalla o a su vez se puede enviar a un documento de Word [6].

Por otro lado Raheja, Singhal y Chaudhary [7] diseñaron un proceso que consiste en extraer información relevante de las imágenes de los videos y guardarlos en vectores

(PCA: Principal Component Analysis) y compararlo con los patrones guardados en una red neuronal, esta información representa el patrón de cada imagen en lenguaje de señas.

También Vintimilla Sarmiento en [8] propone una aplicación para traducción de Lenguaje de Señas de Ecuador donde, para el reconocimiento de la seña primero se realiza el tratamiento de la imagen en el móvil, el resultado se envía al servidor de inteligencia artificial ubicado en la nube, el servidor obtiene los 1600 bits de la imagen y realiza la búsqueda de la misma en una red neural mediante el algoritmo back-propagation, finalmente éste servidor envía la respuesta al dispositivo móvil desplegándose en la pantalla.

Otra de las herramientas utilizadas para la detección de objetos son los clasificadores en cascada o Clasificadores Potenciados (Boosted Cascade Classifiers), particularmente han sido usados para la detección de manos y detección de gestos en las manos. Kolsch y Turk propusieron un sistema de detección de manos que detecta seis gestos [9]. Por su parte Fang, Wang, Cheng y Lu desarrollaron una extensión de [9] en la cual los clasificadores potenciados se emplean para la detección de manos, mientras que los gestos o señas se reconocen utilizando características derivadas del espacio de escala [10].

La mayoría de los proyectos analizados utilizan una computadora para realizar el procesamiento de la imagen. Por ello el interes está en los proyectos basados en Clasificadores en Cascada, mismos que pueden ser integrados a cualquier dispositivo móvil a través de librerias de terceros para procesamiento de imagen compatibles. Para la propuesta es importante analizar las herramientas y librerias para que la app procese la búsqueda y detección de señas por medio de la cámara del movil para una mayor portabilidad y movilidad de la misma. Aunque se hace evidente también la necesidad de considerar la cantidad de recursos del móvil que tendría que utilizar dicha aplicación para llevar a cabo esta traducción, por lo que se debe revisar la opción del procesamiento “fuera” del mismo dispositivo, decir, través de un servicio web.

OBJETIVOS

Objetivo General

Desarrollar una aplicación móvil que realice la traducción de la lengua de señas mexicana (LSM) a texto por medio de la cámara del dispositivo y que permita posteriormente su reproducción con voz artificial para facilitar la comunicación de las personas que tienen problemas auditivos y dificultades del habla con aquellas personas que no conocen o usan dicha lengua.

Objetivos específicos

- Analizar los métodos y librerías reconocimiento de imagen mas eficientes.
- Implementar distintos métodos de reconocimiento de imágenes en dispositivos moviles con SO Android.
- Traducir de una imagen LSM al alfabeto (texto).

ALCANCES Y LIMITACIONES

Para el desarrollo de esta aplicación (denominada “SIGN-APP”) se considero el desarrollo para el Sistema Operativo (SO) Android debido a que es una plataforma de código abierto y porque constituye el SO mas utilizado actualmente [11].

La aplicación “SIGN-APP” realiza la traducción del alfabeto dactilológico, es decir, realiza la interpretación de las señas fijas. La traducción de ideogramas (que representan una palabra con una o varias configuraciones o movimientos de mano) no están consideras en este proyecto debido a la complejidad que representa el procesamiento de mucho de video en tiempo real en un dispositivo móvil.

Finalmente la otra limitante es el uso de un fondo blanco o con colores que sean claros, esto debido a que el procesamiento para la detección de la seña se basa en características (features) de bordes de cada seña, es decir, se realiza la detección de la seña con base en escala de grises.

JUSTIFICACIÓN

La prevalencia de la discapacidad en nuestro país es de 6%, es decir, en México existen 2.4 millones de personas sordas, de las cuales, 84 mil 967 son menores de 14 años, 124 mil 554 son jóvenes de entre 15 y 29 años y 597 mil 566 tienen entre 30 y 59 años según los datos de la ENADID 2014 [12]. Cabe señalar que cada vez se hace más notoria la necesidad de utilizar los dispositivos móviles como una herramienta de inserción social, y en este sentido el desarrollo de una aplicación móvil para que las personas que padecen sordera y que utilizan el lenguaje de señas como herramienta de comunicación tiene una utilidad e impacto relevantes.

Así como las lenguas orales no son universales, a la lengua de señas le ocurre lo mismo, cada país tiene su propia lengua de señas. Así mismo la Lengua de señas de un mismo país también tiene variaciones lingüísticas, debido a las diferencias culturales, de identidad y geografía [8]. En este tenor es importante indicar que ya existen desarrollos de aplicaciones para traducción de lengua de señas a texto, sin embargo tales aplicaciones son específicas para algún lenguaje de señas distinto al de México, por ejemplo la Lengua de Señas de Ecuador (LSEC), la Lengua de Señas Americana (ASL), entre otras. Por lo anterior es propicio el desarrollo de una aplicación móvil para facilitar la comunicación de las personas que usan la LSM con los demás con el propósito de proporcionar un medio de comunicación acorde al desarrollo tecnológico y accesible a la mayoría de las personas.

El beneficio de una aplicación móvil es que puede ser transportada por el individuo y se podría utilizar para comunicarse más fácilmente con cualquier persona que no conoce o sabe el lenguaje de señas. El impacto de una aplicación como esta seria amplio, ya que posibilita desarrollos importantes como parte del trabajo a futuro, por ejemplo podría utilizarse en personas de menor edad o bien podría utilizarse para incluir a las personas que utilizan la LSM como medio de comunicación principal en las aulas de las instituciones educativas del país.

Capítulo 1 INTRODUCCIÓN

Actualmente el uso de dispositivos y aplicaciones móviles está en continuo crecimiento. En estos días hay un creciente desarrollo de apps para la mayoría de actividades cotidianas de las personas. Este crecimiento implica la generación de aplicaciones que permitan a las personas mejorar su calidad de vida, por ejemplo, aplicaciones para monitorear el ritmo cardiaco, para el monitoreo de algún servicio público, o bien aplicaciones para guiar en un mapa algún viaje. Considerando las mejoras en cuanto a los recursos con los que estos aparatos cuenta tales como cámara, procesador, sensores, acceso a internet, entre otros el desarrollo de aplicaciones asistivas resulta una oportunidad factible y necesaria.

En México el desarrollo de aplicaciones móviles para sordomudos es un área aun por explorar. En el país hay 7.1 millones de habitantes que no pueden o tienen mucha dificultad para hacer alguna de las ocho actividades evaluadas: caminar, subir o bajar usando sus piernas; ver (aunque use lentes); mover o usar sus brazos o manos; aprender, recordar o concentrarse; escuchar (aunque use aparato auditivo); bañarse, vestirse o comer; hablar o comunicarse; y problemas emocionales o mentales. En este contexto se encuentras las personas sordas, quienes utilizan la Lengua de Señas Mexicana (LSM) en México, misma que, como toda lengua, posee su propia sintaxis, gramática y léxico.

En este documento se describe el desarrollo de una aplicación móvil que permitirá que personas que utilizan la Lengua de Señas Mexicana (LSM) puedan comunicarse con personas que no usan dicha lengua. El funcionamiento de la aplicación será el siguiente: la persona abrirá la aplicación, la cámara del dispositivo móvil reconocerá las señas que hará esta persona (en lenguaje dactilológico), luego la aplicacion procesara estas capturas de la camara y finalmente mostrará el texto correspondiente a las señas detectadas, finalmente este software contara con un botón para poder reproducir en

voz artificial el texto traducido. La aplicación será desarrollada para el Sistema Operativo abierto Android.

En el capitulo 2 se decriben todos los elementos y conceptos necesarios para la creación de la aplicacion propuesta con el objetivo de contextualizar el software desarrollado. Asi mismo se incluye una revisión de los trabajos previos relacionados con esta aplicación con la finalidad de conocer los avances que se han conseguido sobre este problema y la innovación de la aplicación.

En el capitulo 3 se describe de manera detallada el proceso de solución al problema. Primero se describe el método de detección de señas “Haar Cascade Classifier” donde se establecen los elementos y aspectos mas importantes para la detección de una seña por medio de visión artificial; asi mismo se describe el proceso para extraer características (features) de las imágenes que serán los patrones que utilizara la aplicación para la detección y traducción de la seña. Despues se describe el proceso de desarrollo de la app donde se muestran los elementos de la interfaz, las librerías necesarias para procesamiento de la imagen en el dispositivo y su implementación.

En el capitulo 4 se describen y analizan los resultados obtenidos con finalidad de establecer el grado y porcentaje de traducciones exitosas de la app para determinar la efectividad del software. También se describen los diferentes fondos sobre los cuales se probo la aplicación ya que en este proyecto se utilizan fondos claros.

En el capitulo 5 se señalan de manera detallada las conclusiones del trabajo realizado con el objetivo de exponer los alcances de los resultados obtenidos y las consideraciones para trabajos futuros. También se describen algunos factores importantes del funcionamiento de la aplicación con la fnalidad de describir posibles mejoras futuras.

Capítulo 2 Marco Teoríco

Introducción

En este capítulo, se presenta la base teórica o llamado también marco teórico para poder comprender los elementos involucrados en el proceso de traducción de la Lengua de Señas Mexicana a través de la obtención de imágenes por la camara del dispositivo móvil. Posteriormente se presentan brevemente los proyectos revisados relacionados con la traducción de señas de las manos a texto.Tambien se verán los aspectos y herramientas utilizados para procesar imágenes en tiempo real a través de la librería de Visión Computacional OpenCV. Finalmente se describirán las cuestiones necesarias para poder hacer la reproducción de texto con voz artificial (Text-To-Speech o Conversor-Texto-Voz) en dichos dispositivos.

2.1 Cómputo Móvil

Contexto Histórico

"*Los sistemas de cómputo móvil son sistemas de cómputo que se pueden mover fácilmente físicamente y cuyas capacidades informáticas se pueden usar mientras se mueven. Los ejemplos son computadoras portátiles, asistentes digitales personales (PDA) y teléfonos móviles*" [13].

En este sentido el cómputo móvil se basa en el uso de cualquier dispositivo que procese datos y que además tenga o permita una movilidad y portabilidad para que no este fijo y por lo tanto le sea más cómodo al usuario.

De acuerdo con Manilal Patel y Vikram Kaushik [14] la computación móvil es cualquier tipo de computación que utiliza Internet o intranet y los enlaces de comunicaciones respectivos, como WAN, LAN, WLAN, etc. Las computadoras móviles pueden formar una red personal inalámbrica o una Pico-net (ad-hoc).

La definición anterior permite establecer que para que exista esa movilidad y portabilidad del “computo” es necesario que el dispositivo tenga comunicación con otros elementos o componentes electrónicos y con el usuario, por lo tanto, se requiere de una infraestructura de red para una implementación adecuada de computo móvil.

Hay al menos tres clases diferentes de elementos de computación móvil [14]:

1. Computadoras portátiles, unidades livianas compactas que incluyen un teclado con juego de caracteres completos y principalmente diseñados para alojar software que se puede parametrizar, como computadoras portátiles, cuadernos, blocs de notas, etc.

2. Teléfonos móviles, incluido un juego de llaves restringidas diseñado principalmente, pero no restringido, para comunicaciones vocales, como teléfonos celulares, teléfonos inteligentes, teléfonos, etc.

3. Computadoras usables (Wearables) limitadas principalmente a funcionales clave y principalmente diseñadas como incorporación de agentes de software, como relojes, muñequeras, collares, implantes sin llave, etc. Se espera que la existencia de estas clases sea duradera, y complementario en el uso personal, ninguno reemplazando uno al otro en todas las características de conveniencia.

Por lo anterior la computación móvil se está volviendo día a día un paradigma tecnológico de uso más común, y que está cambiando la forma en que se realizan las actividades laborales, sociales, académicas, de investigación, de entretenimiento y muy particularmente las relacionadas con la salud y la inserción social. El cómputo móvil es un campo de investigación relativamente nuevo con poco más de tres décadas de historia. Durante su ciclo de vida, se ha pasado de ser principalmente técnico a ser también sobre usabilidad, utilidad y experiencia del usuario [15].

En los últimos años el uso de los dispositivos móviles ha crecido a un ritmo vertiginoso, existen varias marcas en el mercado de dispositivos móviles tanto bajo licencia como

de código abierto y a la vez estas herramientas cada vez tienen una mayor capacidad de procesamiento.

Las primeras computadoras móviles, los precursores de las computadoras portátiles actuales, se desarrollaron a fines de la década de 1970 y principios de la de 1980, inspiradas en la portabilidad del concepto de Dynabook de Alan Kay de 1968 [15].

La historia del cómputo móvil puede ser dividida en un número de fases o etapas, cada una de ellas caracterizada por un enfoque particular tecnológico, tendencias de diseño de interacción y por liderar ciertos cambios fundamentales en el uso y diseño de los dispositivos móviles. Esta evolución se puede dividir en siete fases consecutivas como se muestra en la figura 2.1 [15].

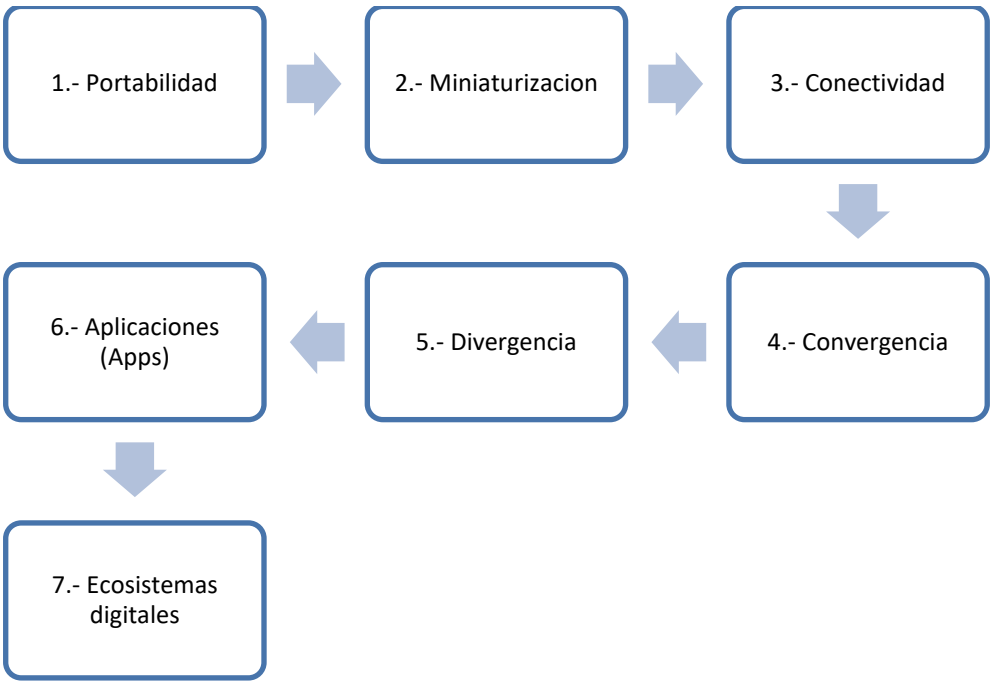


Figura 2. 1 Etapas de desarrollo del computo móvil.

La etapa de la **portabilidad** se enfocó en reducir el tamaño de hardware para permitir la creación de los dispositivos que físicamente podrían ser movidos fácilmente. La fase de la **miniaturización** fue sobre la creación de los nuevos y considerablemente más pequeños factores y elementos que permitieron el empleo de dispositivos personales móviles más pequeños y en movimiento.

El periodo de la **conectividad** fue el desarrollo de dispositivos y aplicaciones que permitieron a usuarios estar en línea y comunicarse vía redes de datos inalámbricas con movilidad y portabilidad.

La etapa de **convergencia** fue sobre la integración de dispositivos móviles digitales emergentes tales como PDA, Celulares móviles, reproductores de música, cámaras, juegos, entre otros.

La fase de **divergencia** tomo un rol contrario al anterior para enfocarse en el diseño de interacción promoviendo aplicaciones con funcionalidades especializadas en lugar de funcionalidades generales.

La penúltima etapa fue la de **apps** que tiene que ver con el desarrollo de software que será ejecutado y usado en dispositivos móviles, y haciendo que el acceso a éstas app sea interactivo, fácil y divertido.

Finalmente, la fase emergente de **ecosistemas digitales** se trata de los conjuntos más grandes de tecnologías penetrantes e interrelacionadas de las que los sistemas móviles interactivos se están convirtiendo cada vez más.

Es importante señalar que desde los inicios de la computación móvil y la interacción móvil humano-computadora, los contextos de uso de sistemas y dispositivos móviles interactivos a menudo han sido resaltados como particularmente importantes para que los desarrolladores de sistemas "sean conscientes de" y "tengan en cuenta" al diseñar y construir sistemas móviles interactivos, así como al evaluar y estudiar su uso. Los

contextos de uso móvil se han descrito como particularmente desafiantes en comparación con, por ejemplo, los contextos de uso de los sistemas de oficina estacionarios tradicionales debido a su naturaleza altamente dinámica, compleja y, de hecho, móvil. También se ha sugerido a menudo que cuando se utiliza un sistema de computadora móvil interactiva, otras actividades en el contexto son a menudo más importantes que la interacción real con el mismo sistema (por ejemplo: caminar por la calle, socializar en un bar o cafetería, o atender a un paciente en un hospital).

Actualmente los dispositivos móviles cuentan con diversos recursos tanto físicos como lógicos que pueden ser utilizados con fines específicos a través de lenguajes de programación y bibliotecas o librerías externas. Por lo tanto, en estos aparatos permiten el desarrollo e instalación de software (apps) para múltiples necesidades y con características avanzadas, es decir, se pueden explotar los elementos como la cámara y el procesamiento para el desarrollo de aplicaciones asistivas.

Las principales plataformas para dispositivos móviles actuales son: Android, IOS, Windows Phone y Black Berry.

Plataforma Android

Tanto en las computadoras conocidas (Lap-top, de escritorio, entre otras) como en dispositivos móviles el componente que hace funcionar el hardware es lo que se conoce como software. El software es el conjunto de líneas o instrucciones que contienen la lógica y funciones a realizar en el dispositivo, es decir, son el conjunto de programas escritos en algún lenguaje de programación para darle funcionalidad a la computadora o dispositivo móvil. En este sentido una aplicación móvil o “app” es un programa o software que tiene la característica de estar dirigido y desarrollado específicamente para ser utilizado en dispositivos móviles.

Debido al gran número de herramientas, dispositivos y lenguajes de programación para el desarrollo de apps se clasifican en tres tipos[16]:

1. **Aplicaciones nativas:** Son aquellas aplicaciones desarrolladas de manera específica para dispositivos con un Sistema Operativo en específico, por ejemplo, aplicaciones nativas para Android o IOS. La principal ventaja es que para su desarrollo se cuenta con kit de desarrollo de software que permite utilizar y explotar de manera más profundo los diferentes elementos disponibles del dispositivo.
2. **Aplicaciones Móviles Web (Web apps):** Son aplicaciones desarrolladas para ser ejecutadas en el navegador web del dispositivo, es decir, a través de tecnologías para el desarrollo de aplicaciones web se consigue el desarrollo de aplicaciones adaptables al tamaño de la pantalla del dispositivo. El beneficio principal es el ahorro de desarrollo de aplicaciones para cada sistema operativo.
3. **Aplicaciones hibridas:** Son aplicaciones que son construidas con herramientas para aplicaciones web pero que incluyen algunas “librerías”, “plug-in” o componentes adicionales para conseguir utilizar algunos recursos “nativos” del dispositivo. Son aplicaciones que tanto elementos de aplicaciones nativas y como de aplicaciones web.

En el desarrollo de aplicaciones móviles se considera más conveniente el desarrollo nativo cuando se requiere la utilización de funciones y elementos más avanzados tanto del sistema operativo como del dispositivo. No obstante, si se necesita una aplicación que no requiere acceso total a los recursos físicos es más conveniente y útil el desarrollo de una aplicación hibrida.

Por lo tanto, el desarrollo de aplicaciones móviles tiene que considerar entre otros aspectos:

1. El sistema operativo o firmware a utilizar.
2. El lenguaje de programación a utilizar.
3. Tipo de aplicación móvil a desarrollar.
4. Recursos físicos con los que cuenta el dispositivo
5. Librerías o funcionalidades adicionales (no disponibles) de manera nativa en el dispositivo.

La telefonía móvil está cambiando la sociedad actual de una forma tan significativa como lo ha hecho Internet. Esta revolución no ha hecho más que empezar; los nuevos terminales o dispositivos ofrecen unas capacidades similares a una computadora personal, lo que permite que puedan ser utilizados para leer el correo o navegar por Internet. Pero, a diferencia de una computadora, un teléfono móvil siempre está en el bolsillo de las personas. Esto permite un nuevo abanico de aplicaciones mucho más cercanas al usuario. De hecho, muchos autores coinciden en afirmar que la nueva computadora personal del siglo XXI será un terminal móvil [17].

El lanzamiento de Android como nueva plataforma para el desarrollo de aplicaciones móviles ha causado una gran expectación y está teniendo una importante aceptación tanto por parte de los usuarios como por parte de la industria. En la actualidad se ha convertido en la alternativa dominante frente a otras plataformas como iPhone o Windows Phone.

Existen muchas plataformas para móviles (Apple iOS, Windows Phone, BlackBerry, Palm, Java Micro Edition, Linux Mobile (LiMo),Firefox OS,etc.); sin embargo, Android presenta una serie de características que lo hacen diferente. Es el primero que combina en una misma solución las siguientes cualidades [17]:

1. **Plataforma realmente abierta:** Es una plataforma de desarrollo libre basada en Linux y de código abierto. Una de sus grandes ventajas es que se puede usar y customizar el sistema sin pagar derechos.
2. **Adaptable a cualquier tipo de hardware:** Android no ha sido diseñado exclusivamente para su uso en teléfonos y tabletas. Hoy en día podemos encontrar relojes, cámaras, electrodomésticos y gran variedad de sistemas empotrados que se basan en este sistema operativo. Este hecho tiene sus evidentes ventajas, pero también va a suponer un esfuerzo adicional al programador. La aplicación ha de funcionar correctamente en dispositivos con gran variedad de tipos de entrada, pantalla, memoria, etc. Esta característica

contrasta con la estrategia de Apple. En iOS tenemos que desarrollar una aplicación para iPhone y otra diferente para iPad.

3. **Portabilidad asegurada:** Las aplicaciones finales son desarrolladas en Java lo que nos asegura que podrán ser ejecutadas en cualquier tipo de CPU, tanto presente como futuro. Esto se consigue gracias al concepto de máquina virtual.
4. **Arquitectura basada en componentes inspirados en Internet:** Por ejemplo, el diseño de la interfaz de usuario se hace en XML, lo que permite que una misma aplicación se ejecute en un reloj de pantalla reducida o en un televisor.
5. **Filosofía de dispositivo siempre conectado a Internet:** Muchas aplicaciones solo funcionan si disponemos de una conexión permanente a Internet. Por ejemplo, comunicaciones interpersonales o navegación con mapas.
6. **Gran cantidad de servicios incorporados:** Por ejemplo, localización basada tanto en GPS como en redes, bases de datos con SQL, reconocimiento y síntesis de voz, navegador, multimedia, etc.
7. **Aceptable nivel de seguridad:** Los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja que hereda de Linux. Además, cada aplicación dispone de una serie de permisos que limitan su rango de actuación (servicios de localización, acceso a Internet, etc.). Desde La versión 6.0 el usuario puede conceder o retirar permisos a las aplicaciones en cualquier momento.
8. **Optimizado para baja potencia y poca memoria:** En el diseño de Android se ha tenido en cuenta el hardware específico de los dispositivos móviles. Por ejemplo, Android utiliza la Máquina Virtual ART(o Dalvik en versiones antiguas). Se trata de una implementación de Google de la máquina virtual de Java optimizada para dispositivos móviles.
9. **Alta calidad de gráficos y sonido:** gráficos vectoriales suavizados, animaciones, gráficos en 3D basados en OpenGL. Incorpora codecs estándares más comunes de audio y vídeo, incluyendo H.264 (AVC), MP3, AAC, etc.

Android nos ofrece una forma sencilla y novedosa de implementar potentes aplicaciones para diferentes tipos de dispositivos. A lo largo de este texto trataremos de mostrar de la forma más sencilla posible cómo conseguirlo.

Uno de los aspectos más útiles de la plataforma Android es la posibilidad de integrar software de terceros listo para usarse (librerías), es decir, reutilizar software desarrollado previamente y utilizarlo en aplicaciones móviles de manera que es mas ágil el desarrollo de aplicaciones, a la vez que es más flexible y eficiente. Por lo tanto es una plataforma flexible y versátil que permite integrar librerías adicionales para el procesamiento de imágenes necesarias para la detección de señas.

2.2 Lengua de Señas Mexicana (LSM)

La lengua de señas en la discapacidad auditiva

La comunicación es fundamental para el desarrollo social del ser humano. De hecho, la vida en comunidad no puede concebirse sin la facultad de acceder a la información que se genera en los diferentes entornos. Entre las diversas formas de comunicación, la expresión oral es la más común y acompaña a la persona, como herramienta de participación, durante toda su existencia.

Cuando, por cualquier motivo, el habla se ve impedida, la posibilidad de alcanzar una verdadera realización social se reduce de manera importante. La dificultad de las personas sordas para comunicarse disminuye su capacidad de interacción social; en consecuencia, su desarrollo educativo, profesional y humano quedan restringidos seriamente, lo que limita las oportunidades de inclusión que todo ser humano merece, y esto representa un acto discriminatorio. Como medio de socialización y mecanismo compensatorio, las personas sordas han desarrollado su propio lenguaje, la lengua de señas. Aun cuando ésta permite a las personas sordas comunicarse entre sí, no les facilita la relación con el resto de la comunidad, en especial, con los oyentes que desconocen ese lenguaje.

Una sociedad justa y equitativa debe otorgar a todos sus integrantes las mismas oportunidades; entre ellas, el acceso igualitario a la comunicación y a la información. Para lograr la participación plena de las personas sordas en el entorno social, es

necesario facilitar su interacción con la comunidad y aportar todos los medios para proporcionarles una comunicación eficaz y fluida. La herramienta básica en la consolidación de este proceso es la lengua de señas [18]..

La Lengua de Señas Mexicana (LSM), es la lengua que utilizan las personas sordas en México. Como toda lengua, posee su propia sintaxis, gramática y léxico

La LSM se compone de signos visuales con estructura lingüística propia, con la cual se identifican y expresan las personas sordas en México. Para la gran mayoría de quienes han nacido sordos o han quedado sordos desde la infancia o la juventud, ésta es la lengua en que articulan sus pensamientos y sus emociones, la que les permite satisfacer sus necesidades comunicativas así como desarrollar sus capacidades cognitivas al máximo mientras interactúan con el mundo que les rodea.

La lengua de señas mexicana está compuesta de la dactilología y los ideogramas. Se conoce como dactilología a lo que bien podría ser el deletreo en la lengua oral, y está representada en este diccionario sobre todo con el abecedario. Cada palabra se puede representar con la articulación de mano correspondiente de cada letra que la conforma. De acuerdo con esto, mamá puede representarse con cada una de sus letras, como se muestra en la figura 2.2 [18].

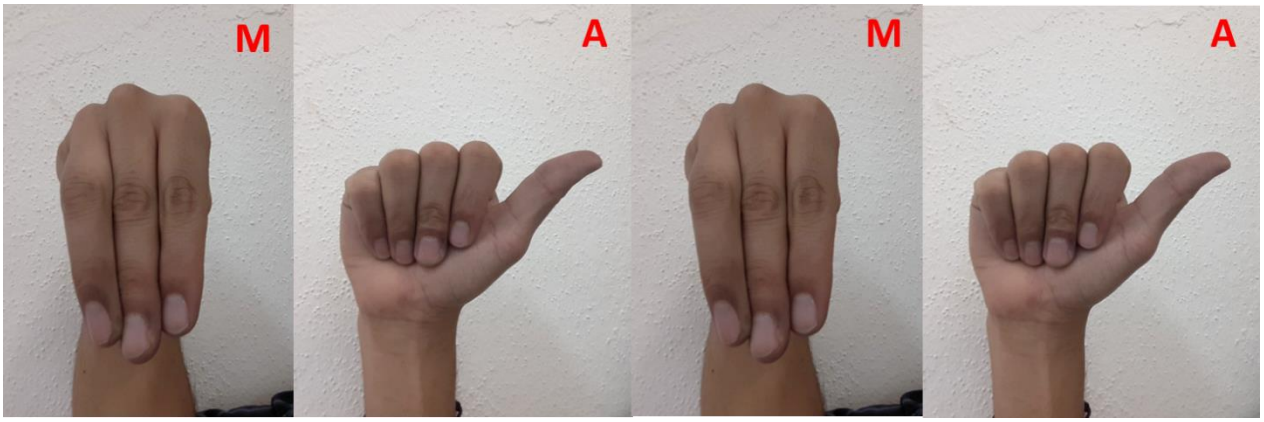


Figura 2. 2 Ejemplo de una palabra descrita con dactilologia en LSM

Los ideogramas representan una palabra con una o varias configuraciones de mano. De acuerdo con esta realización, mamá se articula con la letra “m” de ideograma sobre los labios, con la que se golpean varias veces los labios (figura 2.3) [18].

Figura 2. 3 Ejemplo de ideograma de la palabra MAMA en LSM

Fuente: http://www.wikisigns.org/es/lsm/mama

La lengua de señas es una de las principales herramientas que permiten a personas que padecen sordera comunicarse con los demás. Es importante aclarar que, a pesar de la creencia de que la lengua de señas es universal, es específica de cada país; incluso en el mismo país hay regionalismos, como en cualquier idioma. La persona que aprende lengua de señas debe considerarse bilingüe, aunque haya aprendido las señas de la lengua oral de su región.

Hay distintas metodologías para la comunicación con las personas sordas. La más básica y sencilla es seguir el orden sintáctico del español hablado al realizar las señas de las palabras para formar las expresiones que se deseen comunicar. Otra metodología para la comunicación es el lenguaje natural de las personas sordas, que sigue su propia gramática.

En la figura 2.4 se describe el alfabeto dactilológico de la Lengua de Señas Mexicana, particularmente las letras.

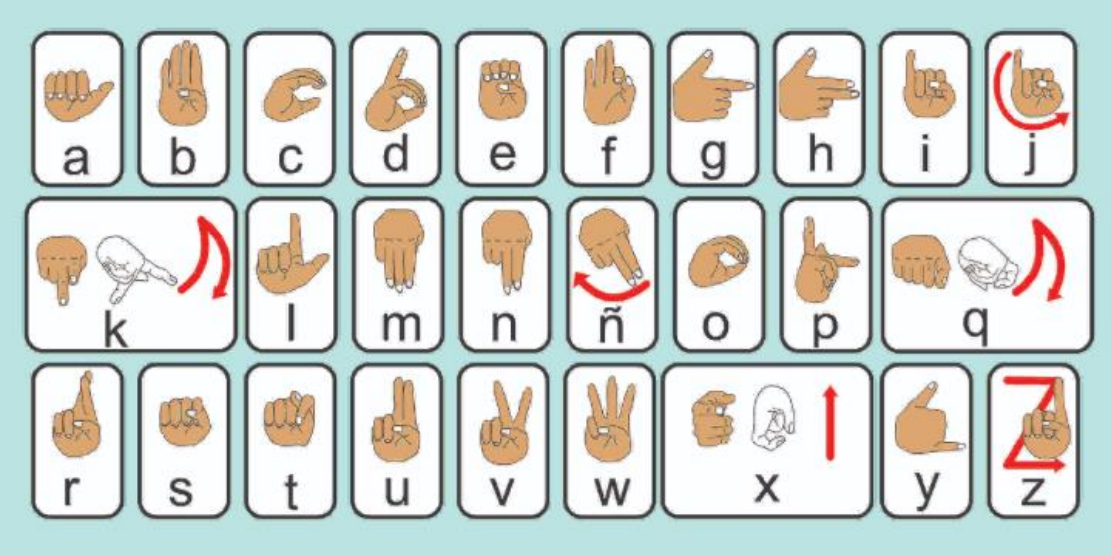


Figura 2. 4 Alfabeto de la LSM

Fuente: http://www.escuelaparasordos.com/lsm.php

En este proyecto se propone una aplicación móvil para realizar la traducción de dactilología ya que, por tratarse señas estaticas requiere menos procesamiento en el dispositivo.

Aplicaciones móviles para Lengua de Señas

Una de las tendencias actuales mas importantes es el desarrollo de aplicaciones móviles para apoyo a necesidades sociales, por ejemplo, usuarios que padecen sordera, pérdida de visión entre otras; el uso y desarrollo de tecnología para este tipo de necesidades es un área en desarrollo y representa una oportunidad para una utilización responsable e inclusiva de dichas apps y dispositivos móviles.

Al respecto ha habido algunos avances, tanto en sitios web como aplicaciones móviles para usuarios sordos, por ejemplo, en Argentina “El servicio Web “Lengua de Señas Argentina” (LSA) administra un diccionario generado a partir de un conjunto de videos

que se encuentran ordenados en distintas categorías: alfabeto, animales, colores, comidas, cosas, frases, números, personas y transportes. De este modo, el servicio está conectado a una base de datos para administrar diferentes funciones referidas al manejo de los archivos de videos" [3].

Otro ejemplo de lo anterior es el desarrollo de una aplicación móvil cuya intención es "facilitar la comunicación entre personas sordomudas y también entre sordomudos y personas sin esta discapacidad". Su funcionamiento es simple, "la aplicación traduce palabras de la LSM (imágenes de señas) a texto y viceversa a través de un diccionario construido para tal fin" [4].

En México se han desarrollado algunas aplicaciones [19] sobre la lengua de señas, sin embargo, el enfoque o propósito de dichas aplicaciones es variado ya que algunas de ellas están diseñadas para aprender dicha lengua, otras solo traducen texto a la lengua de señas en una imagen [20] y algunas de las mas avanzadas traducen de un texto en lenguaje español a señas a través de una animación 3D (que muestra un personaje animado haciendo las señas correspondientes al texto).

El grado de complejidad para el desarrollo de una aplicación móvil que procese las imágenes de LSM basada en ideogramas es elevado, ésto debido a los cálculos necesarios para la detección de las palabras o frases, mismos que dependen de posición de la mano en el espacio, su forma, su altura, asi como la interacción con otras partes del cuerpo en contextos específicos. En este sentido, serían necesarios mas recursos del dispositivo tales como: procesador (tiempo), memoria RAM, procesamiento de video, espacio en la pantalla e interacción con la persona.

Por lo anterior y basado en las condiciones actuales de desarrollo tecnológico de los dispositivos moviles se considera factible el desarrollo de una aplicación móvil para traducción de la LSM basada en dactilología, es decir, traducir a partir del abecedario, mismo que esta basado en imágenes estaticas obtenidas por medio de la cámara del mismo.

2.3 Procesamiento de imágenes

Visión computacional en dispositivos móviles

Visión es la ventana al mundo de muchos organismos. Su función principal es reconocer y localizar objetos en el ambiente mediante el procesamiento de las imágenes. La visión computacional es el estudio de estos procesos, para entenderlos y construir máquinas con capacidades similares [21].

Algunas definiciones de visión son:

- “Visión es saber que hay y donde mediante la vista” (Aristoteles).
- “Visión es recuperar de la información de los sentidos (vista) propiedades válidas del mundo exterior.
- “Visión es un proceso que produce a partir de las imágenes del mundo exterior una descripción que es útil para observador y que no tiene información irrelevante”.

La definición que más se relaciona con la visión computacional actual es la última en el sentido de que es un proceso computacional, depende del observador y es necesario quitar información irrelevante [21].

Un área muy relacionada a la de visión computacional es la de procesamiento de imágenes, aunque ambos campos tienen mucho en común, el objetivo final es distinto. El objetivo del procesamiento de imágenes es mejorar la calidad de las imágenes para su posterior utilización o interpretación, por ejemplo [21]:

- Remover defectos.
- Remover problemas por movimiento o desenfoque.
- Mejorar ciertas propiedades como color, contraste, estructura, etc.
- agregar “colores falsos" a imágenes monocromáticas.

En la figura 2.5 se ilustra el enfoque del procesamiento de imágenes, en el cual se obtiene una imagen “mejor” para su posterior interpretación por alguna persona.

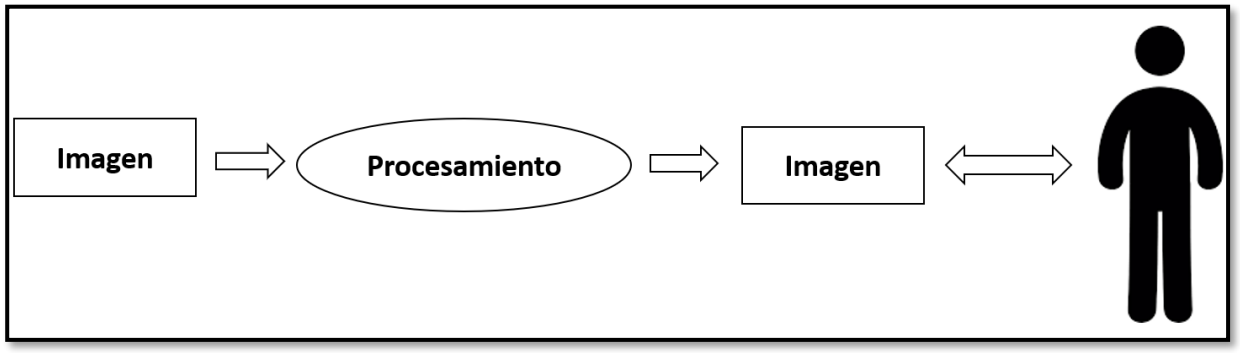


Figura 2. 5 Esquema del procesamiento de imagenes. la entrada y salida son imágenes.

El propósito de la visión computacional es extraer características de una imagen para su descripción e interpretación por la computadora o algún dispositivo móvil. Por ejemplo:

- Determinar la localización y el tipo de objetos en la imagen.
- Construir una representación tridimensional de un objeto.
- Analizar un objeto para determinar su calidad.
- Descomponer una imagen u objeto en diferentes partes.

Por lo anterior podemos considerar que el procesamiento de imágenes es uno de los elementos más importantes del área de visión computacional.

En visión computacional se busca obtener descripciones útiles de la imagen para cada tarea a realizar. La tarea demandará modificar ciertos atributos. La imagen de entrada es procesada para extraer los atributos, obteniendo como salida una descripción de la imagen analizada. En la figura 2.6 se muestra el esquema de la visión computacional [21].

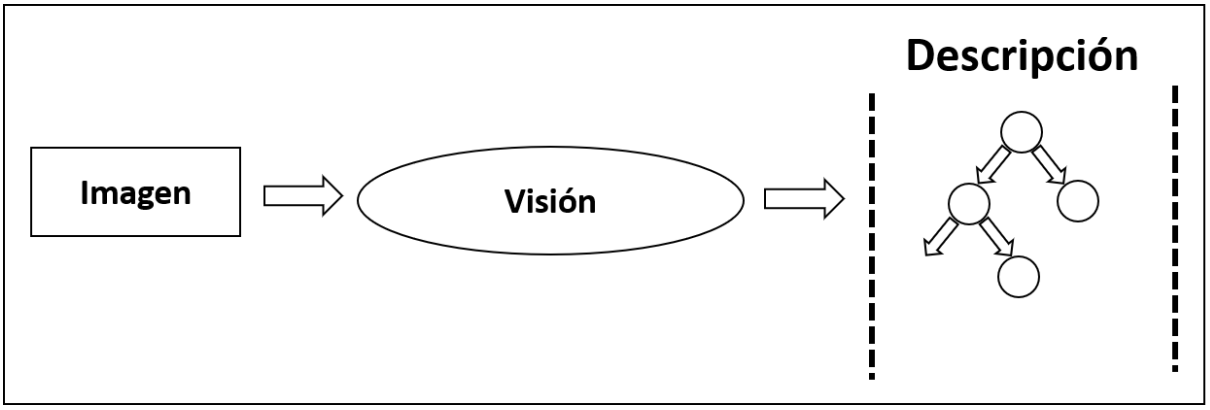


Figura 2. 6 Esquema general de la visión computacional. la salida es una descripcion de la imagen de entrada.

Con base en los elementos revisados se puede considerar la visión computacional móvil (visión movil) como el proceso para extraer características de una imagen para su descripción e interpretación a través de algún dispositivo móvil. En este proceso se buscan utilizar los componentes físicos del “Smartphone” para el procesamiento de la imagen y su posterior interpretación y utilización.

La arquitectura de un sistema de visión artificial se basaba, tradicionalmente, en una computadora como eje central y un conjunto de periféricos. Desde hace ya algunos años, los teléfonos móviles integran capacidad de adquisición de imagen, visualización y comunicación, pero seguían limitados por la capacidad de procesamiento. Sin embargo, esta situación está cambiando. Por un lado el hardware de los nuevos smartphones y tablets está creciendo enormemente en potencia, reduciendo cada vez más la distancia en capacidad de proceso a la de un ordenador tradicional. Por otro lado, los sistemas operativos orientados a la creación de aplicaciones (iOS, Android, etc.) aumentan de forma decisiva las posibilidades en cuanto a desarrollo. Además, el auge de las tiendas de aplicaciones online como AppStore o Google Play, contribuyen a atraer la atención de desarrolladores y consumidores.

Actualmente es factible desarrollar aplicaciones de visión artificial en estos dispositivos, aprovechando su portabilidad e integración de sensores. Una aplicación de visión

artificial en terminales móviles implica los mismos problemas que una de escritorio, como iluminación, oclusiones, variaciones de aspecto, etc.; además también está sujeta a otros factores: potencia computacional, baja calidad de las cámaras y memoria restringida. No obstante las capacidades y recursos de dichos artefactos actuales posibilitan el desarrollo de aplicaciones que reciben imágenes en tiempo real por la cámara y permiten su posterior procesamiento con la finalidad de interpretar el entorno.

En este sentido una aplicación importante de la visión móvil es la traducción de la lengua de señas (dactilología) a texto y voz, es decir, obtener o capturar las imágenes por medio de la cámara del dispositivo para hacer el procesamiento de ellas: detectar la seña hecha por la persona para después efectuar la traducción al texto (letra) correspondiente.

Proyectos sobre reconocimiento de señas

Las distintas herramientas actuales de desarrollo de software, particularmente para aplicaciones moviles han dado paso al desarrollo de apps para muchas áreas, entre ellas las de tipo asistivo, mismas que ofrecen alterativas de apoyo a las personas con alguna necesidad o discapacdad. Ademas los distintos componentes y recursos de los dispositivos actuales posibilitan aplicaciones cada vez mas sofisticadas.

Al respecto hay varios proyectos de software que han diseñado diversas opciones de traducción, reconocimiento y procesamiento. En [22] proponen el diseño de un sistema básico pero extensible capaz de reconocer gestos estáticos y dinámicos del lenguaje de señas americano (LSA), específicamente las letras a - z (donde j y z son dinámicas con movimientos manuales o digitales mientras que el resto es estático). El sistema es capaz de funcionar en un fondo dinámico y mínimamente desordenado con resultados satisfactorio ya que se basa en la segmentación del color de la piel para separar los gestos, es decir, centrarse en las manos y la cara.

En el sistema anterior se enfocan en las manos para reconocer las letras descritas. Los gestos que son clasificados en dinámicos y estáticos. Para los gestos estáticos se utiliza el método momentos de Zernike y para los dinámicos (2 segundos de video) extraen un vector de características en curva que muestra una alta precisión en rutas de identificación única del movimiento, luego estas rutas o patrones son clasificados usando máquinas de soporte vectorial (SVM). Es importante señalar que el sistema está diseñado y probado para el lenguaje de Señas Americano (American Sign Language).

También en [6] desarrollaron un software en el que "*para el procesamiento digital de imágenes se aplicaron algunos filtros y operaciones morfológicas para resaltar las características de la imagen y eliminar información innecesaria como ruido*". También realizan la eliminación de objetos extraños en la imagen mediante un recortado del área de interés. Con la ayuda del Vision Assitant de Labview, se elaboraron las bases de datos, para llevar a cabo la comparación con la imagen recortada y de esta manera asignar la clase (letra) correspondiente a cada imagen. Con la clase asignada se forma el texto que se muestra en forma escrita en la pantalla o a su vez se puede enviar a un documento de Microsoft Word.

La cualidad más importante del software anterior es la obtención de imagen y su procesamiento: una vez detectado el objeto de interés, se pasa esta imagen a escala de grises porque la aplicación no requiere de un análisis del color, pero sí de su forma, con esto se tiene una imagen de menor tamaño para su posterior proceso de segmentación. Después se realiza un contraste a la imagen con la finalidad de eliminar el ruido existente y resaltar la forma del objeto dentro de la imagen. Posteriormente se hace el proceso de convertir la imagen en escala de grises a una imagen binaria, donde los pixeles tengan dos valores, ya sea 1L o 0L. Finalmente se realiza el proceso de clasificación de dicha imagen con su carácter asociado para poderla usar al comparar otras entradas.

Por otro lado, la principal desventaja del proyecto es que no esta desarrollado para dispositivos móviles, es decir, si bien se hace un procesamiento de la imagen y se hace la traducción correspondiente a texto, dicha imagen es obtenida por medio de la cámara de una computadora. En los dispositivos móviles hay que considerar los permisos y el sistema operativo involucrado o bien el firmware especifico. Por otro lado la aplicación solo considera la lengua de señas de Ecuador.

En [7] diseñaron un proceso que consiste en extraer información relevante de las imágenes de los videos y guardarlos en vectores PCA (Principal Component analysis) y compararlo con los patrones guardados en una red neuronal, esta información representa el patrón de cada imagen en lenguaje de señas. Estos patrones son usados para comparar las imágenes a procesar. Una vez obtenida la imagen se detecta el borde con el método sobel. Luego se obtiene el patrón de esta imagen a través de los senos y cosenos de la figura (bordes). Estos patrones constituyen las entradas de la red neuronal. El paso final consiste en clasificar la imagen recibida: este es el resultado de la comparación de los patrones de la imagen con los de la red neuronal.

PCA es una herramienta estándar en el análisis de datos moderno, en diversos campos desde la neurociencia hasta los gráficos por computadora, porque es un método simple y no paramétrico para extraer información relevante de conjuntos de datos confusos. Es una forma de identificar patrones en los datos y expresarlos de tal manera que resalten sus similitudes y diferencias. Dado que los patrones en los datos pueden ser difíciles de encontrar en datos de alta dimensión, donde el lujo de la representación gráfica no está disponible, la PCA es una poderosa herramienta para analizar datos. La otra ventaja principal de PCA es que una vez que ha encontrado estos patrones en los datos, y comprime los datos, es decir, reduciendo el número de dimensiones, sin mucha pérdida de información.

Por su parte en [8] se describe la aplicación para traducción de Lenguaje de Señas de Ecuador (LSE), donde para el reconocimiento de la seña primero se realiza el tratamiento de la imagen en el móvil, el resultado se envía al servidor de inteligencia

artificial ubicado en la nube, el servidor obtiene los 1600 bits (de la imagen) y realiza el la búsqueda recibida en una red neuronal mediante el algoritmo back-propagation. Es necesario señalar que previo a esta búsqueda la representación de la imagen es guardada en dicha nube (internet).

En [31] se propone un sistema de reconocimiento de gestos de la mano en tiempo real, que también se basa en el sistema estándar de Viola & Jones [24]. Se agregan nuevas características rectangulares para el caso de detección de manos. El reconocimiento de los gestos se obtiene mediante el uso de varios detectores de gestos individuales trabajando en paralelo. El sistema final fue validado en un entorno controlado (pared blanca como fondo);

Los proyectos revisados utilizan distintos métodos y recursos para realizar la traducción de la seña a texto. En este sentido se ha revisado cada uno de ellos con la finalidad de establecer y determinar un método eficiente para el proceso de traducción de la seña en un dispositivo móvil con el Sistema Operativo Android en un fondo claro.

Existen varias bibliotecas o librerías para procesamiento de imagen en la plataforma Android:

1. **OpenCV**: Es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

2. **JavaCV**: Es una interfaz para usar OpenCV bajo Java. De hecho comparte la misma documentación que OpenCV. Ha quedado relegada a un segundo plano con la salida de la versión de OpenCV 2.4.4, que soporta Java.

3. **Jon's Java Imaging Library (JJIL):** Es una biblioteca para el procesamiento de imágenes en Java. Incluye una arquitectura de procesamiento de imágenes y más de 60 rutinas para diversas tareas de procesamiento de imágenes. Está especialmente dirigida a aplicaciones móviles. Es eficiente tanto espacio como en tiempo sobre teléfonos móviles. Incluye interfaces para que las imágenes se puedan convertir hacia y desde formatos nativos para J2ME, Android, y J2SE. La biblioteca está disponible como software libre bajo licencia GPL.

La biblioteca elegida para realizar este proyecto ha sido OpenCV, por ser código abierto, contar con funciones que cubren las necesidades del alcance del problema, una documentación muy amplia y un desarrollo con la plataforma Android muy completo.

Librería OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de software de visión abierta y software de aprendizaje automático. OpenCV fue construido para proporcionar una infraestructura común para aplicaciones de visión por computadora y para acelerar el uso de la percepción de la máquina en los productos comerciales. Al ser un producto con licencia de BSD, OpenCV facilita a las empresas utilizar y modificar el código [20]. Tiene interfaces C ++, Python, Java y MATLAB y es compatible con Windows, Linux, Android y Mac OS. OpenCV se inclina principalmente hacia las aplicaciones de visión en tiempo real [23].

La biblioteca OpenCV puede ser usado para proyectos escolares o comerciales, las aplicaciones de esta librería incluyen, la robótica, análisis y procesamiento de imágenes o vídeos, seguimiento y detección de objetos, detección y reconocimiento de rostros, reconocimiento de placas de vehículos, análisis de formas, reconstrucción 3D, realidad aumentada, entre otros.

OpenCV tiene una estructura modular, lo que significa que el paquete incluye varias bibliotecas compartidas o estáticas. Los siguientes módulos están disponibles [23]:

1. **core** : un módulo compacto que define las estructuras de datos básicas, incluida la matriz multidimensional densa `Mat` y las funciones básicas utilizadas por todos los demás módulos.
2. **imgproc** : módulo de procesamiento de imágenes que incluye filtrado de imágenes lineales y no lineales, transformaciones geométricas de imágenes (cambio de tamaño, afinación y deformación en perspectiva, remapeo genérico basado en tablas), conversión del espacio de color, histogramas, etc.
3. **video** : un módulo de análisis de video que incluye estimación de movimiento, sustracción de fondo y algoritmos de seguimiento de objetos.
4. **calib3d** : algoritmos básicos de geometría de vista múltiple, calibración de cámara única y estéreo, estimación de pose de objeto, algoritmos de correspondencia estéreo y elementos de reconstrucción en 3D.
5. **features2d** : detectores de características sobresalientes, descriptores y emparejadores de descriptores.
6. **objdetect** : detección de objetos e instancias de las clases predefinidas (por ejemplo, caras, ojos, tazas, personas, autos, etc.).
7. **highgui** : una interfaz fácil de usar para la captura de video, códecs de imágenes y video, así como capacidades simples de interfaz de usuario.
8. **gpu** - Algoritmos acelerados por GPU de diferentes módulos de OpenCV.
9. Ademas incluye algunos otros módulos auxiliares, como los contenedores (Wrappers) de prueba FLANN y Google, enlaces de Python y otros.

OpenCV fue diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real. Se propone el uso de la biblioteca OpenCV para el proceso de tratamiento de las imágenes obtenidas por medio de la cámara del dispositivo (señas), ya que cuenta con diversas funciones de reconocimiento y análisis de imágenes. Así mismo es una librería totalmente compatible y disponible para Android.

Reconocimiento de señas con OpenCV

OpenCV permite el uso de las denominadas “Haar-Like features” mismas que representan las caracteristicas de las imágenes digitales utilizadas para reconocimiento de objetos.

El modulo de OpenCV “**objdetect**” incluye un detector llamado “Haar Feature-based Cascade Classifier” (Clasificador en Cascada) mismo que hace uso del algoritmo Viola-Jones [24] para la detección de características Haar-like de una imagen, es decir, un clasificador en cascada básicamente le dice a OpenCV qué buscar en las imágenes que se le den como entrada.

Un clasificador “Haar Feature-based Cascade Classifier” (denominado en OpenCV “*cascade of boosted classifiers working with haar-like features”*) se entrena con unos cientos de vistas de muestra de un objeto particular (es decir, una cara, una seña o un automóvil), llamados ejemplos positivos, que se escalan al mismo tamaño (por ejemplo, 20x20) y ejemplos negativos: imágenes arbitrarias que no contienen al objeto buscado (muestra) del mismo tamaño [25].

Después de entrenar un clasificador, se puede aplicar a una región de interés (del mismo tamaño que se usó durante el entrenamiento) en una imagen de entrada donde se buscara dicho objeto. El clasificador genera un "1" si es probable que la región muestre el objeto (es decir, el objeto a detectar: cara, mano, entre otros), y "0" en caso contrario. Para buscar el objeto en la imagen completa, se puede mover la ventana de búsqueda por la imagen y verificar cada ubicación usando el clasificador. El clasificador está diseñado para que pueda "cambiarse el tamaño" fácilmente para poder encontrar los objetos de interés en diferentes tamaños, lo que es más eficiente que cambiar el tamaño de la imagen. Entonces, para encontrar un objeto de un tamaño desconocido en la imagen, el procedimiento de escaneo debe hacerse varias veces a diferentes escalas.

La palabra "cascada" en el nombre del clasificador significa que el clasificador resultante consiste en varios clasificadores (etapas) más simples que se aplican posteriormente a una región de interés hasta que en algún momento el candidato es rechazado o se pasan todas las etapas. La palabra "potenciado" significa que los clasificadores en cada etapa de la cascada son complejos y están construidos a partir de clasificadores básico.

Las características de Haar (Haar-features) describen la relación entre la oscuridad y áreas brillantes dentro de un kernel. Un kernel de imagen es una pequeña matriz utilizada para aplicar efectos y procesamiento. Un ejemplo típico es que la región del ojo en el rostro humano es más oscura que la región de la mejilla, y una característica Haar-like puede detectar eficazmente esa carácteristica. Un sistema basado en características tipo Haar puede operar mucho más rápido que un sistema basado en píxeles.

Las características Haar-like son la entrada a los clasificadores básicos [25], y se calculan como se describe en la figura 2.7.

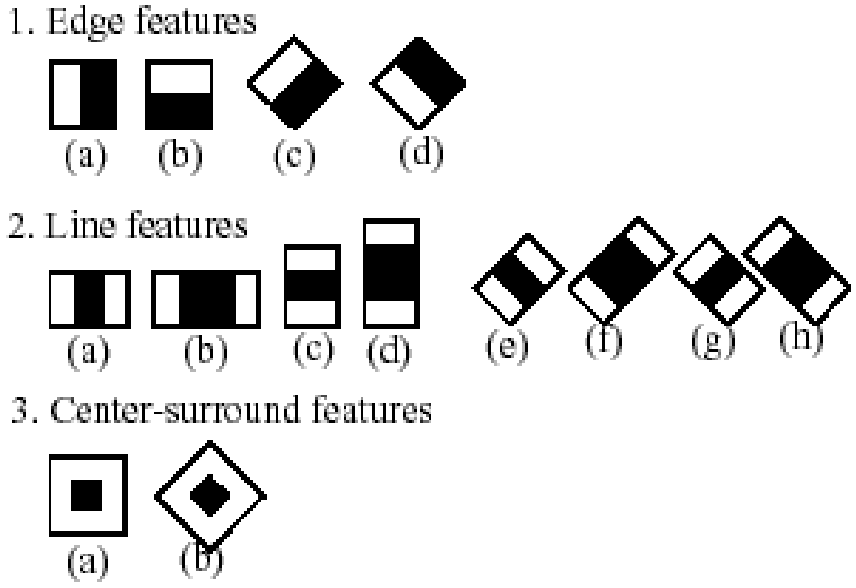


Figura 2. 7 Características Haar-like usadas en clasificadores en Cascada de OpenCV

Cada característica de Haar consiste en dos o tres rectángulos "negros" y "blancos" conectados. El valor de una característica similar a Haar es la diferencia entre las sumas de los valores de píxeles dentro de los rectángulos blanco y negro. En la figura 2.8 se describe visualmente el funcionamiento general de las características Haar-Like.

Figura 2. 8 Visualización simple del funcionamiento de Haar features

De manera general se puede decir que OpenCV busca esas características (a través del Haar Feature-based Cascade Classifier) en la imagen, una vez encontradas se deben asociar con un significado, en este caso una seña.

Finalmente es importante señalar que opencv desde 2010 libero una versión compatible con la plataforma Android, misma que permite utilizar toda la potencia de la biblioteca en el desarrollo de aplicaciones móviles. El IDE oficial para Android (Android Studio) es compatible con distintas versiones de la librería openCV, por lo tanto, es posible utilizar esta biblioteca para el reconocimiento de señas en un dispostivo con Sistema Operativo Android.

2.4 Conversor Texto a Voz (CTV/TTS)

Síntesis del habla

El habla es la forma más utilizada y natural para que las personas se comuniquen. Desde el comienzo de la investigación de la interfaz hombre-máquina, el habla ha sido uno de los medios deseados para interactuar con las computadoras. Por lo tanto, el reconocimiento de voz y la conversión de prueba a voz se han estudiado para hacer más probable la comunicación con máquinas [26].

En un sentido muy amplio, la síntesis del habla se puede definir como todo proceso de creación de habla de manera artificial (por medios mecánicos, electrónicos, etc.).

La interacción por medio del habla con los dispositivos que nos rodean requiere no solo que estos dispositivos nos entiendan (reconocimiento del habla), sino que nos puedan transmitir información. El medio de transmisión de esta información puede ser variado; por ejemplo, visual (puntos luminosos, texto, gráficas, imágenes o vídeos), acústico (sonidos o habla), háptico (vibraciones, temperatura, etc.) o una combinación de dos medios o más. La síntesis del habla reúne todas las técnicas necesarias para transformar en voz cualquier información [27].

La conversión de texto a voz (CTV) es la creación por medios automáticos de una voz artificial que genera el sonido producido por una persona al leer un texto cualquiera en voz alta o una voz artificial, es decir, es decir, son sistemas que permiten la creación de voz sintética o artificial para poder “leer” un texto.

Las tecnologías de conversión de texto a voz se utilizan para sintetizar el texto hablado generado por ordenador, que se asemeja a la voz humana, con un texto redactado que se utiliza como entrada, por lo tanto, con CTV se produce un habla humana de manera artificial [28].

Los sistemas de síntesis actuales se dividen en dos grandes grupos [27]:

- Sistemas de respuesta vocal o de texto restringido.
    - Finalidad: generación acústica de frases con una estructura predeterminada, en la que solo varía parte del vocabulario.
    - Ejemplo: avisos de los trayectos de los trenes (“Próxima estación: Palacio Real”) o información telefónica (“El número solicitado es 902 141 141”).
    - Técnica: funcionamiento basado en la concatenación de trozos de frase o palabras que se han grabado previamente de manera aislada.

- Ventajas: la calidad de la voz generada es elevada, a pesar de que a veces se notan los puntos de unión de las diferentes grabaciones.
- Inconvenientes: no sirven para aplicaciones en las que el vocabulario es muy variable (carteleras de cine, por ejemplo), puesto que incluir palabras nuevas en el vocabulario implica realizar nuevas grabaciones.

• Convertidores de texto a voz

- Finalidad. Lectura de cualquier texto, con independencia del origen que tenga.
- Ejemplo. Lectores de pantalla para personas con deficiencias visuales, entre otros.
- Técnica. Funcionamiento basado en dos pasos: un análisis del texto dado y la generación del audio correspondiente.
- Ventajas. Flexibilidad, puesto que tienen vocabulario ilimitado, y altamente configurables (por ejemplo, sexo y edad del locutor, y velocidad a la hora de hablar).
- Inconvenientes. En comparación con los sistemas de respuesta vocal, la complejidad técnica es muy superior.

Conversión Texto a Voz

Un sistema de conversión texto a voz (CTV) o bien TTS (Text to Speech System) es un software o sistema que convierte una entrada de texto en una salida en forma de señal de audio (voz) cuyo contenido se corresponde con el mensaje del texto de entrada, es decir, permite a un software o dispositivo “hablar” en voz alta de tal manera que permite una interacción mas humana; para ello emplea voces entrenadas a partir de bases de datos que son dependientes del idioma para sintetizar la señal de voz de salida en función de las etiquetas de entrada [27].

Una gran parte de las aplicaciones [27] de los convertidores de texto a voz TTS está relacionada con sistemas de diálogo, en los que trabajan junto con un ASR (Automatic Speech Recognition). Dentro de este tipo de aplicaciones destacamos la atención

telefónica (servicios de información, de notificación de averías, de concertar cita, etc.) y las aplicaciones de mando y control (sistemas domóticos, interacción vocal con coches, PC, etc.).

Los TTS también se utilizan mucho en aplicaciones de ayuda a personas con necesidades especiales. Los lectores de pantalla, junto con teclados adaptados, han permitido el acceso al mundo de la informática e Internet a las personas con dificultades visuales. Para las personas con dificultades en la producción del habla que les afectan a la inteligibilidad, existen dispositivos con teclados predictivos o de construcción rápida de frases, que les permiten sintetizar todo aquello que quieren expresar.

Otro campo de aplicación de los TTS es el aprendizaje de idiomas. En este caso, los estudiantes de idiomas extranjeros pueden sintetizar cualquier frase o palabra para aprender una pronunciación y entonación correctas. Los sistemas de traducción automática voz-voz, a pesar de encontrarse todavía en fase precomercial, constituyen un nuevo campo de aplicación.

Los convertidores de texto a voz tienen como entrada un texto y como salida el audio correspondiente.

Aunque existen diferentes técnicas para realizar esta conversión, todos los TTS comparten la misma arquitectura que se muestra en la figura 2.9:

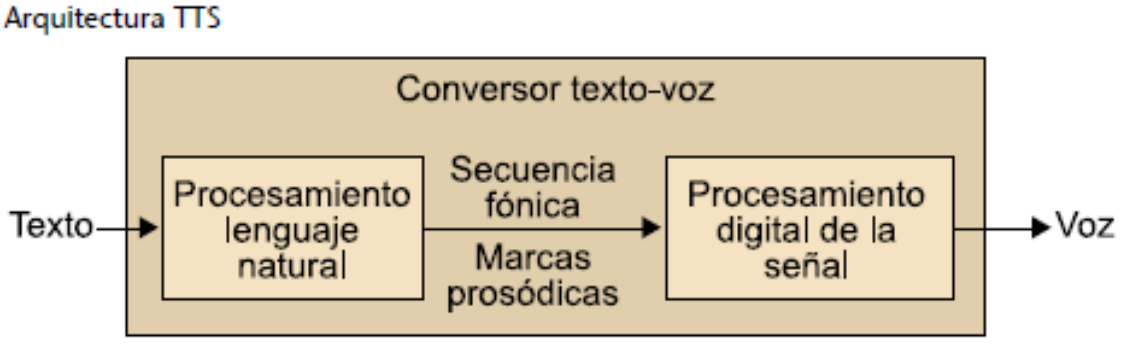


Figura 2. 9 arquitectura de los convertidores de texto a voz

Como se aprecia en la figura 2.9, la arquitectura de un TTS está formada por dos módulos: uno de procesamiento de lenguaje natural y otro de procesamiento digital de la señal.

1. El módulo de procesamiento de lenguaje natural es el encargado de transformar el texto de entrada (texto plano, HTML, etc.) en una transcripción fonética con marcas prosódicas.
2. El módulo de procesamiento digital de la señal genera el audio descrito en la entrada; por ello también se denomina módulo de síntesis acústica.

Síntesis de texto a voz en una aplicación para dispositivos móviles Android

Google libero desde 2013 su motor de síntesis de voz para Android, el cual venía de serie en algunos dispositivos, aunque no en todos, ya que algunos fabricantes meten sus propios motores de síntesis de voz (TTS) [29].

El motor de síntesis de voz de Google, preinstalado en la gran mayoría de los dispositivos, permite a las aplicaciones de Android "hablar" con una voz que suena razonablemente humana. Tambien permite que las aplicaciones lean en voz alta los textos o opciones de accesibilidad, se encuentra disponible para dispositivos con Android 4.0.3 o superior, y cuenta con los idiomas alemán, coreano, español, francés, inglés (Estados Unidos), inglés (Reino Unido) e italiano.

A través del SDK (Software development Kit) de Android es posible integrar el sitetizador de voz en cualquier aplicación, dotándola de la función “hablar”. Dicha funcionalidad permitirá reproducir con voz artificial el texto obtenido de la traducción de las señas.

Android proporciona la clase “TextToSpeech” para este propósito. Para usar esta clase, necesita instanciar un objeto de esta clase y también especificar el “initListener” [30].

La lógica interna utilizada por el motor de síntesis de voz es bastante compleja. Sin embargo, la plataforma Android permite integrar en cualquier aplicación este sistema de una forma sencilla sin necesidad de que el desarrollador conozca su metodología. Para la implementación de una aplicación basada en la síntesis de texto a voz, únicamente es necesario que ésta se encargue de enviar el texto que se quiere leer en voz alta al motor de síntesis para que lo reproduzca. Esta funcionalidad se consigue a partir del paquete **android.speech.tts** de la API de Android, en particular, la clase **android.speech.tts.TextToSpeech**.

Capítulo 3 REVISIÓN TÉCNICA Y METODOLOGÍA

En este capitulo se describe de manera detallada el proceso de creación de la aplicación propuesta. Primero se describe el método de detección de señas con “Haar Cascade Classifier” donde se establecen los elementos y aspectos mas importantes para la detección de una seña por medio de visión artificial con OpenCV; posteriormente se describe el proceso para extraer características (features) de las imágenes que serán los patrones que utilizara la aplicación para la detección y traducción de la seña. Despues se describe el proceso de desarrollo de la app donde se muestran los elementos de la interfaz, la integración de las librerías necesarias para procesamiento de la imagen en el dispositivo y su implementación.

3.1 Metodo para la detección de señas usando Haar-Like features

El funcionamiento de los Clasificadores Haar Cascades

Los clasificadores Haar son clasificadores en cascada que se basan en las características de Haar (Haar-Like features). Un clasificador en cascada es simplemente una concatenación de un conjunto de clasificadores simples o debiles (weak classifiers) que se pueden usar para crear un clasificador complejo. Los clasificadores simples son clasificadores cuyos desempeños o alcances son limitados. No tienen la capacidad de clasificar todo un objeto correctamente. Los clasificadores complejos o fuertes (strong classifiers), por otro lado, son muy buenos para clasificar los datos de un objeto completo correctamente.

Otra parte importante de las cascadas de Haar son las características de Haar (Haar-features). Estas características [32] son sumas simples de rectángulos y diferencias de esas áreas a través de la imagen. Para ilustrarlo mejor se muestra un ejemplo, considerado en la figura 3.1.

Si de la figura 3.1 se desean calcular las características Haar de la región ABCD, solo hay que calcular la diferencia entre los pixeles blancos y los pixeles coloreados en esa región. Como se muestra en las cuatro figuras, se usan diferentes patrones para contruir las características Haar. Existen muchos otros patrones que pueden ser usados de manera similar. Se crean en multiples escalas para hacer el sistema invariante de escala, es decir, se escala la imagen para calcular las mismas características nuevamente. De esta manera se hace mas robusto el sistema de detección de algún objeto.

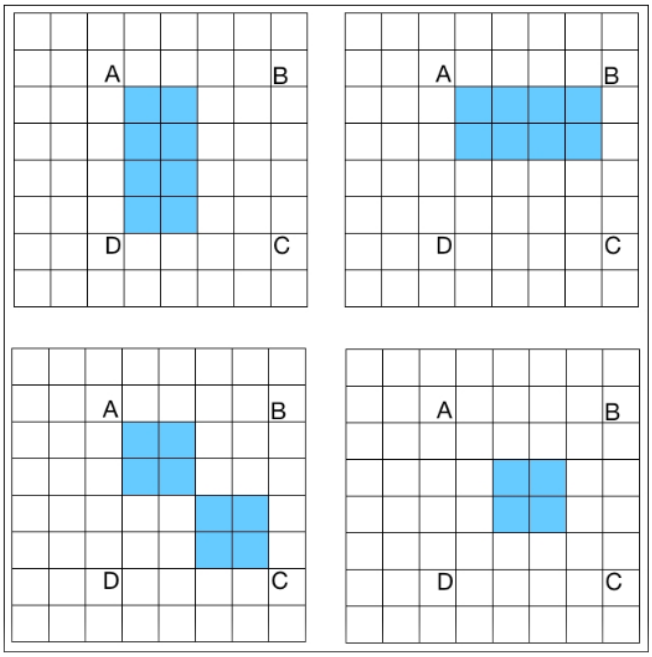


Figura 3. 1 Ejemplo de Haar-features para detectar objetos

Como resultado, este sistema de concatenación es un método muy bueno para detectar objetos en una imagen. En 2001, Paul Viola y Michael Jones [24] publicaron un artículo simiente en el que describieron un método rápido y eficaz para la detección de objetos.

Básicamente en describieron un algoritmo que usa un clasificador en cascada formado de clasificadores simples. Este sistema se usa para construir un clasificador fuerte que puede funcionar eficientemente. Usando esta técnica, pudieron evitar el problema de

tener que construir un clasificador único que pueda funcionar con alta precisión. Estos clasificadores de un solo paso tienden a ser complejos y computacionalmente intensivos. La razón por la cual su técnica funciona es porque los clasificadores simples pueden ser aprendices débiles, lo que significa que no necesitan ser complejos.

Por ejemplo si se considera construir un sistema que automáticamente aprenda cómo se ve una mesa. En base a este conocimiento de los clasificadores, debería ser capaz de identificar si hay una mesa en cualquier imagen dada. Para construir este sistema, el primer paso es recopilar imágenes que se pueden usar para entrenar el sistema. Hay muchas técnicas disponibles en el mundo del aprendizaje automático que se pueden usar para entrenar un sistema como este. Hay que tener en cuenta que se deben recopilar muchas mesas e imágenes que no sean mesas si queremos que el sistema funcione bien. En las cuestiones de aprendizaje automático, las imágenes de la mesa se denominan muestras positivas y las imágenes que no son de la mesa se denominan muestras negativas. Este sistema sistema tomará estos datos y luego aprenderá a diferenciar entre estas dos clases.

Para construir un sistema en tiempo real, se debe mantener el clasificador simple y funcional. La única cuestión es que los clasificadores simples no son muy precisos. Si se trata de hacerlos más precisos, terminarán siendo computacionalmente intensivos y, por lo tanto, lentos. Este tipo de equilibrio entre precisión y velocidad es muy común en el aprendizaje automático. Entonces, para solucionar este problema se concatena un grupo de clasificadores débiles para crear un clasificador fuerte y unificado. No se requiere que los clasificadores débiles sean muy precisos. Para garantizar la calidad del clasificador global, Viola y Jones [24] han descrito una técnica conocida como el paso de cascada, es decir, para determinar si en una imagen se encuentra una seña o no, el algoritmo divide la imagen (imagen integral) en subregiones de tamaños diferentes y utiliza los clasificadores. En cada clasificador se determina si la subregión es una seña o no, por lo que no serán procesadas subregiones de la imagen que no se sepa con certeza que contienen una seña y sólo se invertirá tiempo en aquellas subregiones que posiblemente si contengan una seña.

Ahora que se describió el proceso general, la aplicación móvil deberá detectar señas en tiempo real. El primer paso es extraer características de todas las imágenes. En este caso, los algoritmos necesitan estas características para aprender y comprender cómo son las señas. Se utilizan las características de Haar para construir los vectores de características. Una vez que se extraen estas características, se pasarán a través de una cascada de clasificadores. Es decir, simplemente se verifican todas las diferentes subregiones rectangulares y se van descartando las que no tienen señas. De esta manera, llegamos a la respuesta final rápidamente para ver si un rectángulo dado contiene una seña o no.

La imagen Integral.

Para extraer estas características de Haar se necesita calcular la suma de los valores de los píxeles incluidos en muchas regiones rectangulares de la imagen. Para que sea invariante a escala, se requiere calcular estas áreas en escalas múltiples (es decir, para varios tamaños de rectángulo). Si se implementa ingenuamente, este sería un proceso computacionalmente muy tardado. Se tendría que iterar sobre todos los píxeles de cada rectángulo, incluida la lectura de los mismos píxeles varias veces si están contenidos en diferentes rectángulos superpuestos. Si desea construir un sistema que pueda ejecutarse en tiempo real, no se puede perder tanto tiempo en el cálculo. Se necesita encontrar una manera de evitar esta enorme redundancia durante el cálculo del área porque se iteraría sobre los mismos píxeles varias veces. Para evitar esto, se puede usar algo llamado imágenes integrales. Estas imágenes se pueden inicializar en un tiempo lineal (iterando solo dos veces sobre la imagen) y luego pueden ser provistas con la suma de píxeles dentro de cualquier rectángulo de cualquier tamaño leyendo solo cuatro valores. Se ilustra el ejemplo con la figura 3.2.

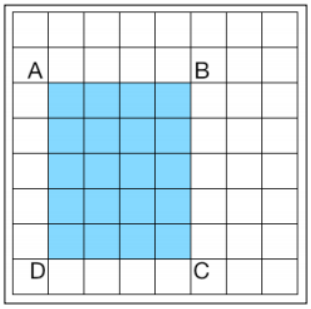


Figura 3. 2 Ejemplo de área a calcular para extraer características

Si se quiere calcular el área de cualquier rectángulo en la imagen, no se tiene que iterar a través de todos los píxeles en esa región. Por ejemplo en la figura 3.3 considere un rectángulo formado por el punto superior izquierdo en la imagen y cualquier punto P como la esquina opuesta. Entonces AP denota el área de este rectángulo. Por ejemplo, en la figura anterior, AB denota el área del rectángulo de 5X2 formado tomando el punto superior izquierdo y B como esquinas opuestas.

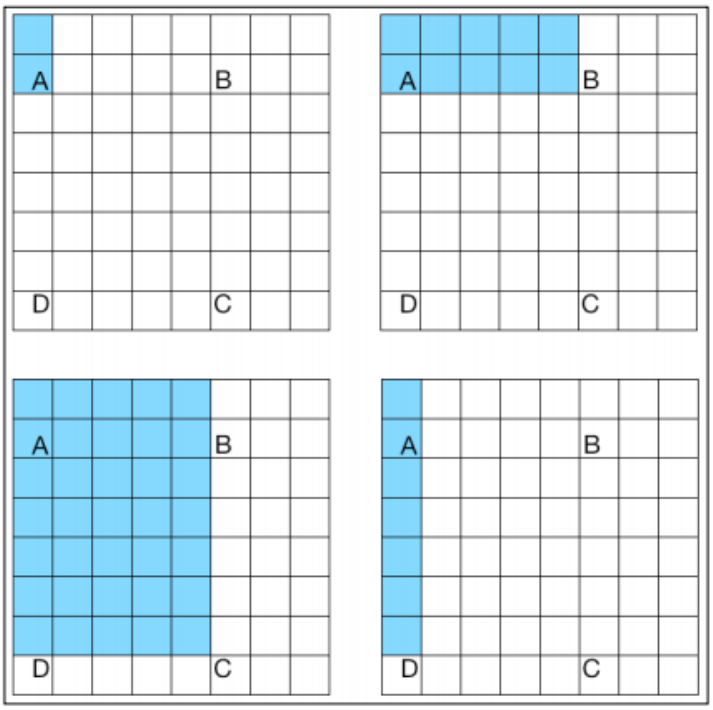


Figura 3. 3 Ejemplo de áreas para obtener la imagen integral

Se considerará el diagrama superior izquierdo en la figura 3.3. Los píxeles coloreados indican el área entre el píxel superior izquierdo y el punto A. Esto se denota por A_A. Los

diagramas restantes se denotan por sus respectivos nombres: A_B, A_C y A_D. Ahora, si se quiere calcular el área del rectángulo ABCD, como se muestra en la figura anterior, se usa la siguiente fórmula:

Área del rectángulo ABCD = A_C - (A_B + A_D - A_A)

Como se describió, extraer las características de Haar de la imagen incluye calcular estas sumas, y se tendría que hacer para muchos rectángulos en escalas múltiples en la imagen. Muchos de estos cálculos son repetitivos porque se estaría iterando sobre los mismos píxeles una y otra vez. Es tan lento que construir un sistema en tiempo real no sería factible. Por lo tanto, se necesita esta fórmula. Como se puede ver, no se tiene que iterar sobre los mismos píxeles varias veces. Si se quiere calcular el área de cualquier rectángulo, todos los valores en el lado derecho de la ecuación anterior están disponibles en la imagen integral. Simplemente se obtiene los valores correctos, se sustituyen en la ecuación anterior y se extraen las características.

OpenCV proporciona un buen marco de trabajo de detección de objetos. Para lograrlo se tiene que cargar el archivo en cascada y usarlo para detectar los objetos en una imagen o inclusive en un video.

Los clasificadores Haar representan los patrones que OpenCV buscara en las imágenes de entrada de la cámara (en tiempo real) con la finalidad de determinar que seña esta haciendo la persona. Dicha librería incluye los algoritmos para crear los clasificadores a partir de las imágenes positivas y negativas.

Proceso para la detección de señas de la LSM

En la aplicación propuesta se utilizan los clasificadores Haar (Haar-Like features) porque permiten la integración en un dispositivo móvil Android a través de la librería OpenCV y permiten la detección en tiempo real de objetos en imágenes captadas por la cámara del móvil.

Para realizar la detección de señas es necesario tener “integrada” la librería openCV con el fin hacer uso de los clasificadores Haar-Like. La app utiliza ésta biblioteca para procesar las imágenes captadas por la cámara de manera continua (frames), es decir, primero se obtienen las imágenes de la cámara (frames), después en cada una de ellas se “buscan” las señas con la ayuda de dichos clasificadores creados previamente para cada una de las señas, posteriormente una vez hecha la detección se mostrara el texto (letra) correspondiente tanto en la vista de la cámara como en un componente de la aplicación establecido para visualizar el texto en español.

El proceso para la traducción de la seña se muestra en la figura 3.4:

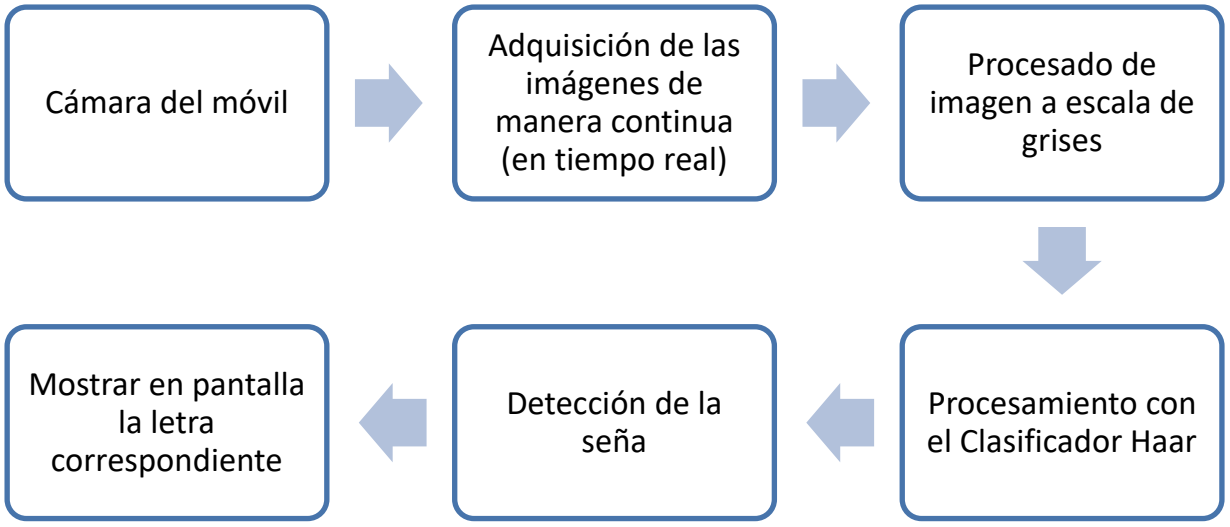


Figura 3. 4 Proceso para detectar una seña.

Como se puede observar el proceso inicia con la obtención de imágenes a través de la cámara integrada en el dispositivo. Después estas imágenes son recibidas por OpenCV donde se realiza el procesamiento: en primer lugar la imagen es procesada a escala de grises; en segundo lugar en la imagen en escala de grises es utilizada para buscar las diferentes señas a través de los clasificadores Haar generados; en tercer lugar una vez detectada alguna seña a la persona le es mostrada la letra correspondiente tanto en texto como en la visualización de la cámara que incluye la app.

Entonces un proceso fundamental para el desarrollo del sistema es generar los clasificadores Haar que utiliza la aplicación para poder detectar las señas por medio de la captura de iamgenes de la cámara en tiempo real. En el siguiente apartado se describe el proceso de creación de estos patrones.

3.2 Generación de Clasificadores Haar para LSM

El poder del clasificador Haar es que rechazará rápidamente las regiones que es muy poco probable que contengan el objeto. Lo hace haciendo uso de la cascada de clasificadores. En esta cascada, las primeras etapas rechazarán rápidamente la mayoría de las regiones falsas y la detección de objetos puede pasar a otras regiones. Sin embargo, las etapas posteriores requieren progresivamente más esfuerzo computacional para rechazar la región. Al hacer esto, el clasificador Haar solo pasará un tiempo considerable en regiones que probablemente contengan el objeto que se busca.

Las aplicaciones necesarias para implementar un clasificador Haar están incluidas en OpenCV mismas que pueden usarse para entrenar a un clasificador y para detectar objetos en una imagen. La detección de objetos usando clasificadores en cascada basados en características de Haar es un enfoque basado en el aprendizaje automático en el que una función en cascada se entrena a partir de muchas imágenes positivas (el objeto que se quiere detectar) y negativas (imágenes sin el objeto). Luego se usa para detectar objetos en otras imágenes.

Pasos para entrenar y usar un clasificador Haar con OpenCV [33]:

- Recolectar imágenes de entrenamiento positivas y negativas
- Marcar las imágenes positivas
- Las imágenes marcadas deben empacarse en un archivo vectorial (.vec)
- Entrenar al clasificador usando la aplicación HaarTraining de OpenCV
- Probar el clasificador con CascadeClassifier.

Aquí se busca la detección de señas. Inicialmente, el algoritmo necesita muchas imágenes positivas (imágenes de la seña) e imágenes negativas (imágenes sin la seña) para entrenar al clasificador. Para lograr esto es necesario extraer ciertar características de él utilizando las funciones de Haar que se mostraron anteriormente. Cada característica es un valor único obtenido a través del método de la **imagen integral**.

Sobre la cantidad de muestras [34] dependen de una variedad de factores, incluida la calidad de las imágenes, el objeto que desea reconocer, el método para generar las muestras, la potencia de la CPU que tiene y probablemente algo de magia.

No está claro exactamente cuántas muestras de cada tipo de imagen se necesitan. Si se toman demasiadas imágenes de muestra el tiempo de entrenamiento puede ser muy extenso [33]. En este sentido se tomaron 102 imágenes positivas y se colocaron 200 negativas. La fotos fueron tomadas con un fondo claro para facilitar el entrenamiento y las pruebas, en este caso las letras a, m, h, o, s y b ya que son señas estaticas. Para cada una de ellas es necesario crear su clasificador con la finalidad de que la aplicación las detecte en tiempo real. En los siguientes pasos se muestra el proceso de generación del clasificador de la letra A, este proceso será el mismo para las diferentes señas.

Imágenes de muestra

Se necesita recolectar imágenes positivas que contengan solo objetos de interés, en este caso la letra A, de la cual se tomarón 102 fotos con fondo claro. El tamaño original de las imágenes fue de 1151x1151 pixeles, sin embargo el tiempo de generación del clasificador con estos seria muy extenso, por lo tanto las imágenes son reducidas de tamaño para la generación del clasificador mas rápida. Es importante señalar que este cambio no afecta el proceso de detección pues el algoritmo funciona con diferentes escalas de las imágenes donde se buscará la seña (captadas por el móvil). En la figura 3.5 se muestra este cambio.

Figura 3. 5 Tamaños de las imágenes para la generación del clasificador Haar

Para las muestra negativas de utilizarón 200 imágenes, mismas que por definición no contienen el objeto a detectar (seña). En la figura 3.6 se muestran algunas imágenes.

Figura 3. 6 Algunas imagenes negativas (sin la seña)

Crear el clasificador

Para el proceso de generación del clasificador se utilizó la herramienta “Cascade Trainer GUI” desarrollada por Amin Ahmadi [35]. En la figura 3.7 se muestra la interfaz.

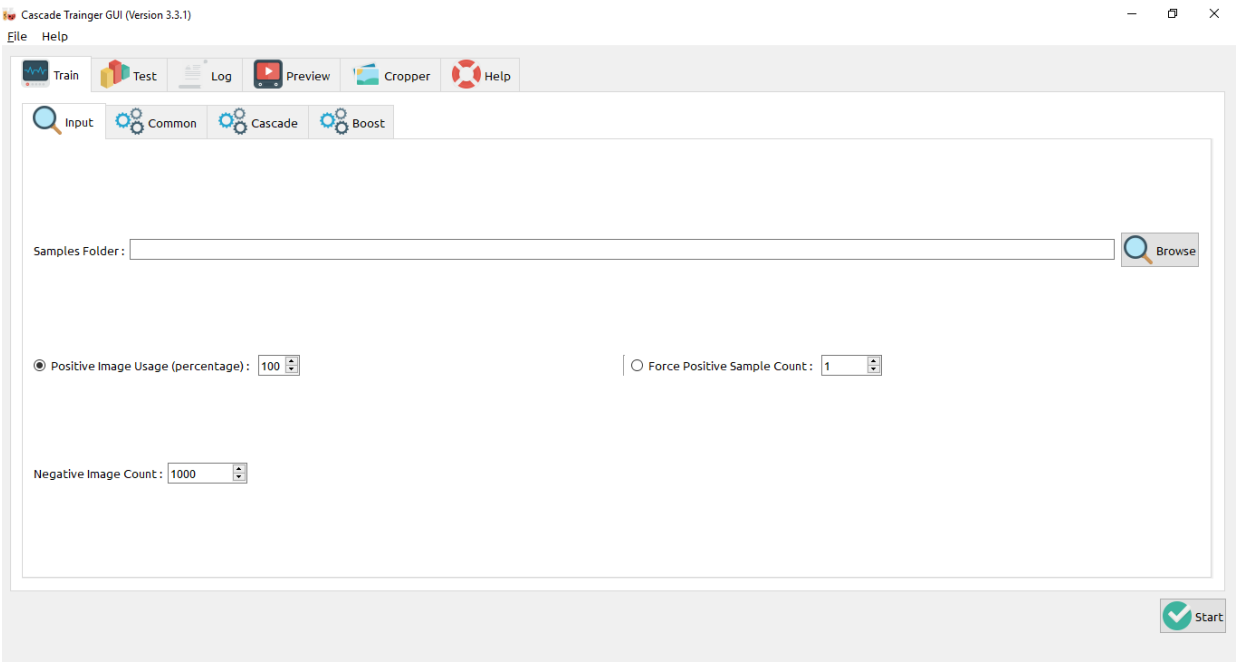


Figura 3. 7 Herramienta gráfica para generar el clasificador Haar

Para comenzar el entrenamiento, se debe crear una carpeta para el clasificador. Se crean dos carpetas dentro de ésta. Uno debe ser nombrada "p" (para imágenes positivas) y el otro debe ser "n" (para imágenes negativas). En la figura 3.8 se muestra este diseño.

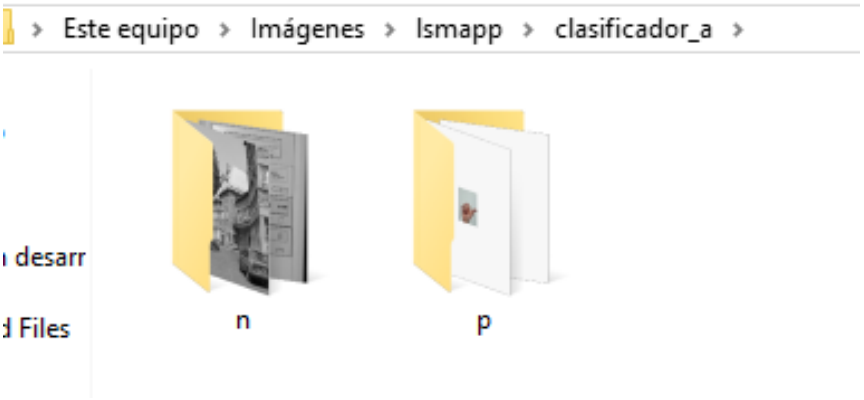


Figura 3. 8 Muestras positivas y negativas de la seña “A”

En la figura 3.9 observamos la configuración del tamaño, como se mencionó en parrafos anteriores el tamaño del clasificador no afecta al momento de la detección, ya

que método de Viola-Jones se basa en clasificadores en cascada para multiples escalas.

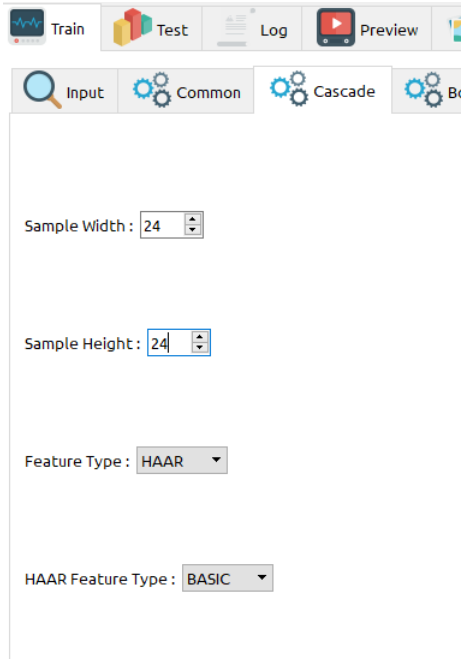


Figura 3. 9 Configuración del clasificador Haar

Los archivos generados por el programa se muestran en la figura 3.10, notese que el clasificador Haar se llama **cascade.xml**.

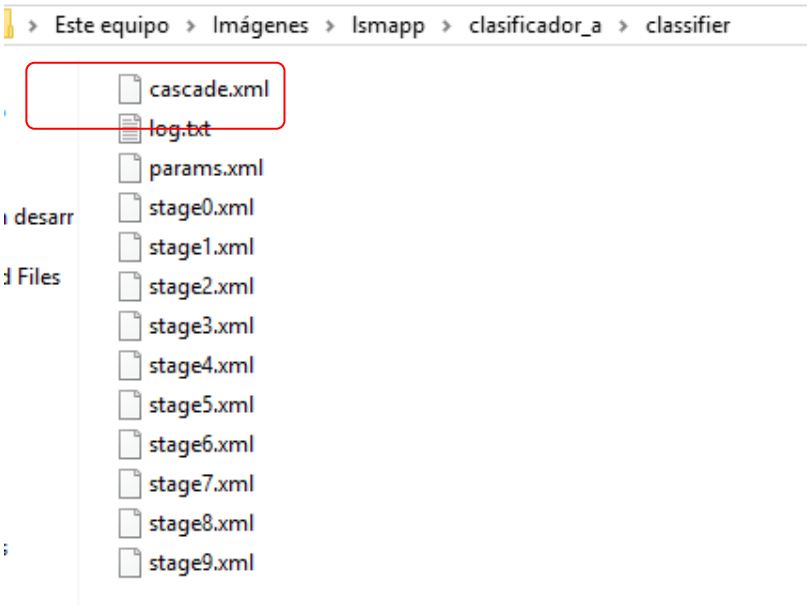


Figura 3. 10 Generación de los archivos del clasificador

Una vez generado el clasificador de la seña “A” sigue el proceso de probarlo, para ello la misma herramienta cuenta con la pestaña “Test” para probarlo en imágenes e incluso video. En la figura 3.11 se describe este proceso.

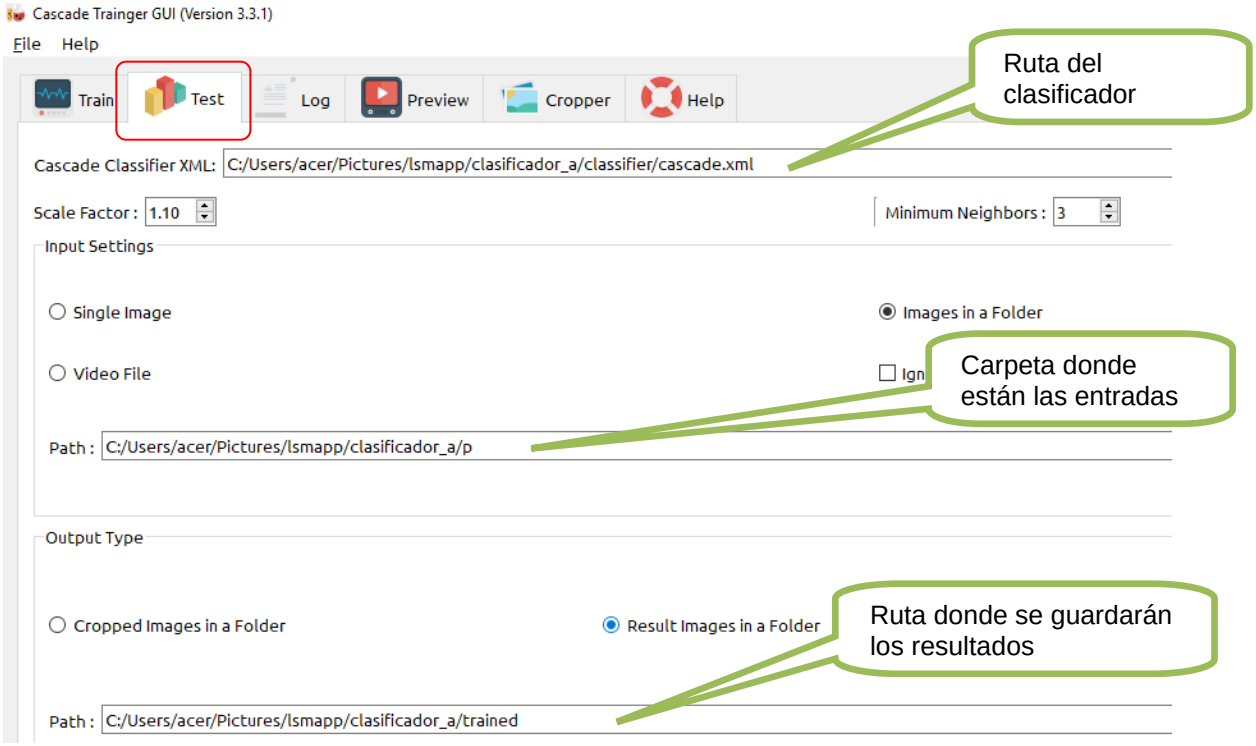


Figura 3. 11 Pruebas del clasificador

Los resultados de las pruebas se muestran en la figura 3.12, donde observamos una clasificación exitosa de la mayoría de las señas de la letra “A”. Se puede notar que algunas no las detecto para lograrlo es necesario agregar mas muestras positivas con la finalidad de mejorar la clasificación y detección.

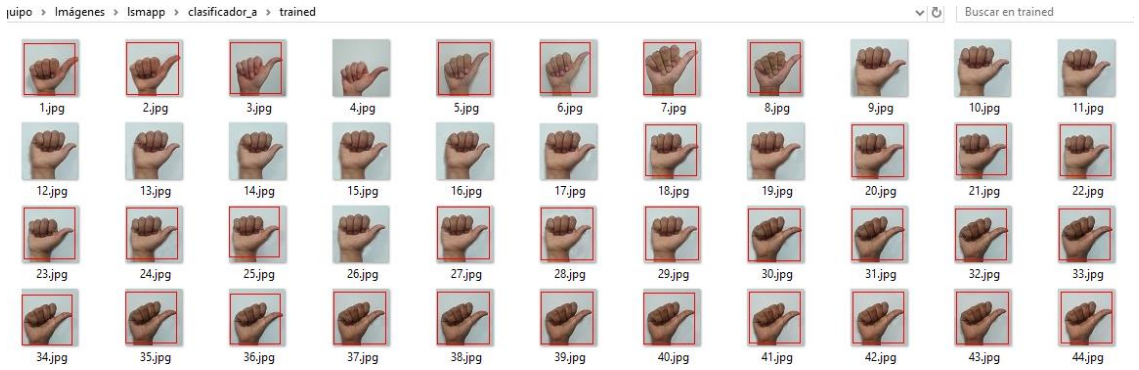


Figura 3. 12 Algunos de los resultados de las pruebas del clasificador

El proceso mostrado para este clasificador Haar de la seña de la letra “A” es el mismo que se utiliza en cada una de las señas estáticas.

3.3 Diseño de la aplicación

Los colores seleccionados fueron obtenidos con base en las recomendaciones Google para Material Design, particularmente se generación del sitio web “Material Design Palette” tal y como se muestra en la figura 3.13.

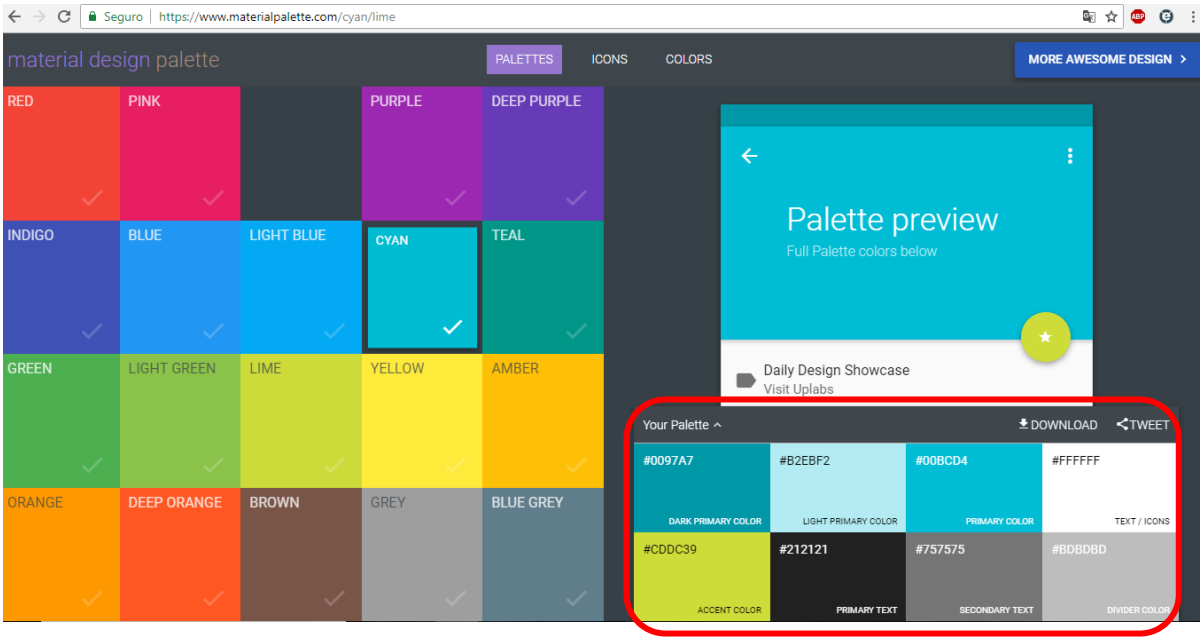


Figura 3. 13 Paleta de colores seleccionada para la Aplicación

El logo de la aplicación se diseño con los colores selectos, se buscó un aspecto simple e intuitivo sobre la intención de la app. En la figura 3.14 se muestra este logo.

Figura 3. 14 Logo/icono de la app

El diseño del SplashScreen se baso en el logo de la aplicación, por lo que su interfaz se muestra en la figura 3.15.

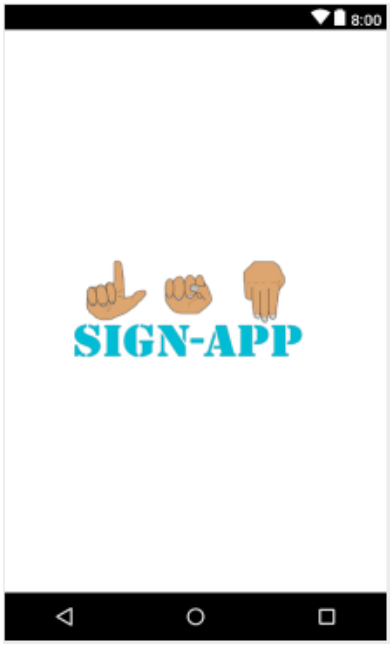


Figura 3. 15 SplashScreen de la app

Para la creación de la interfaz gráfica de la aplicación se considero un diseño simple e intuitivo, pues el propósito de la app es traducir las señas del lenguaje dactilológico de la LSM. En la figura 3.16 Se muestra el diseño propuesto, numerando los elementos.

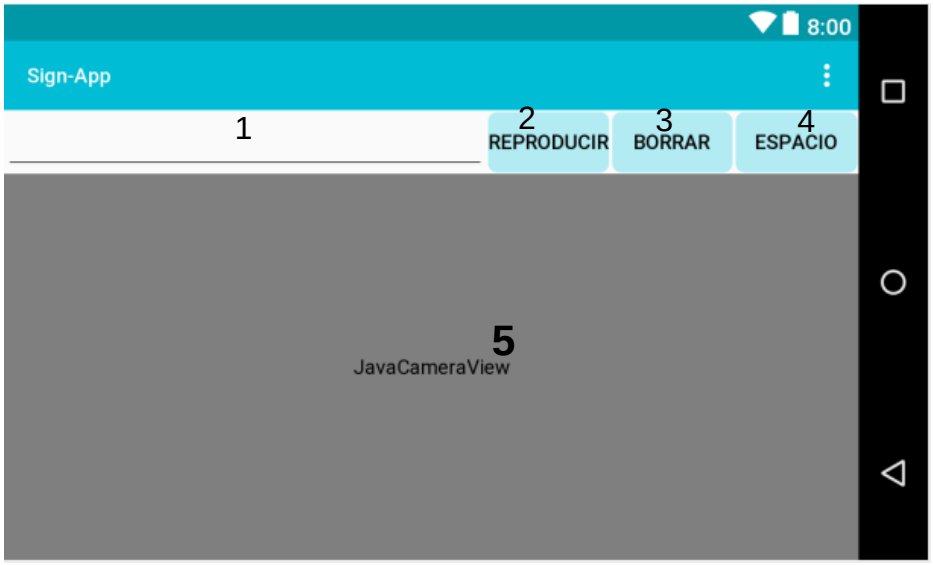


Figura 3. 16 Diseño de aplicación LSM-APP

En la imagen se muestran los componentes necesarios, los cuales se describen a continuación:

1. Este elemento es una entrada de texto (EditText), donde se colocarán las señas detectaras de manera consecutiva.
2. Es un botón (Button) que tiene el propósito de realizar la lectura con voz artificial colocado en la entrada de texto.
3. Boton para eliminar el texto traducido de la entrada de texto (EditText).
4. Es un botón (Button) que colocara un espacio de texto al ser presionado para facilitar la lectura de palabras formadas por señas. Esto debido a que la detección de es de las letras, además no se tiene una para indicar el espacio.
5. Este elemento (JavaCameraView) representa lo que la cámara del dispositivo esta “viendo” o captando, es utilizado para que la persona visualice las señas detectadas en tiempo real.

3.4 Desarrollo de la aplicación con Android Studio

Se utilizó el IDE oficial liberado por Google para el desarrollo nativo de aplicaciones móviles para la plataforma Android conocido como Android Studio Versión 3. Cabe señalar que se realizaron varias pruebas para verificar la integración adecuada de OpenCV en el proyecto.

Creación del proyecto

El primer paso consistio en crear el proyecto, tal y como se muestra en la figura 3.17.

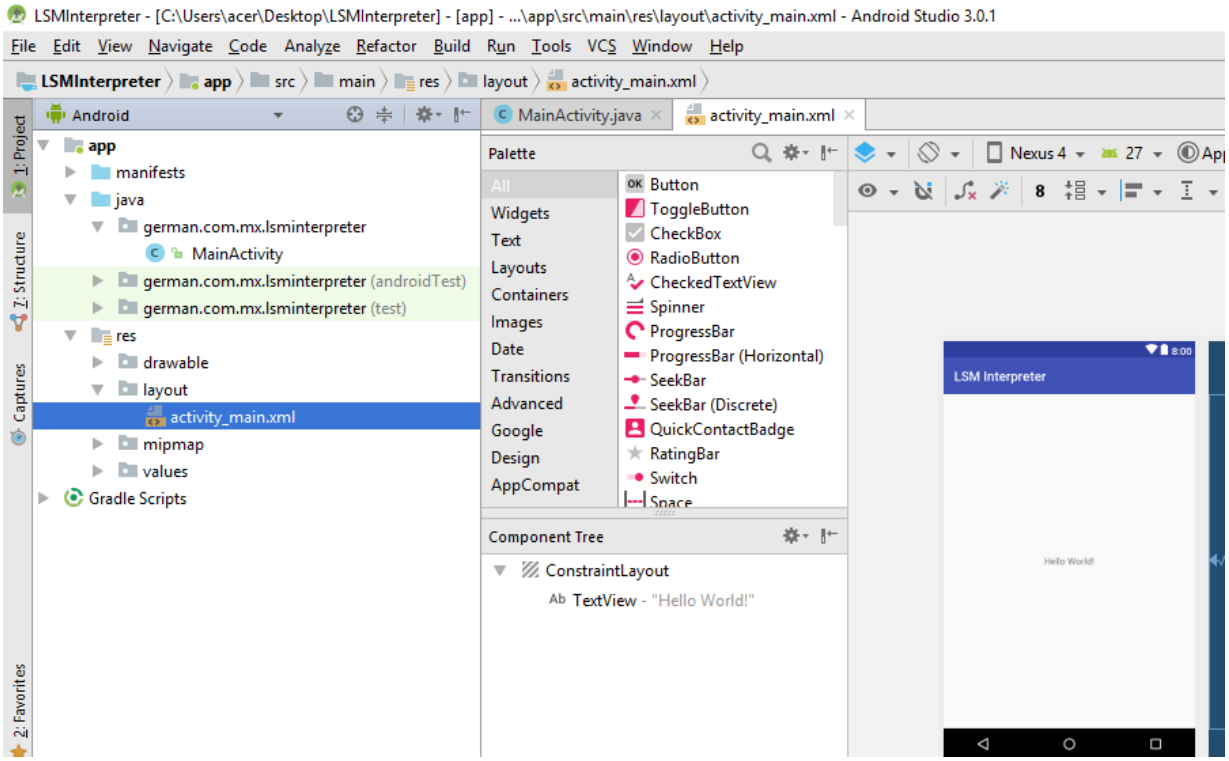


Figura 3. 17 Creación del proyecto con Android Studio

Despues es necesario integrar la biblioteca openCV como se ha mencionado. Para ello es necesario descargar la versión adecuada. Para este proyecto se eligío la versión 2.49. En la figura 3.18 se ilustra la integración adecuada de la misma al Editor Android Studio (en el apendice 1 se describe el proceso detallado de integración).

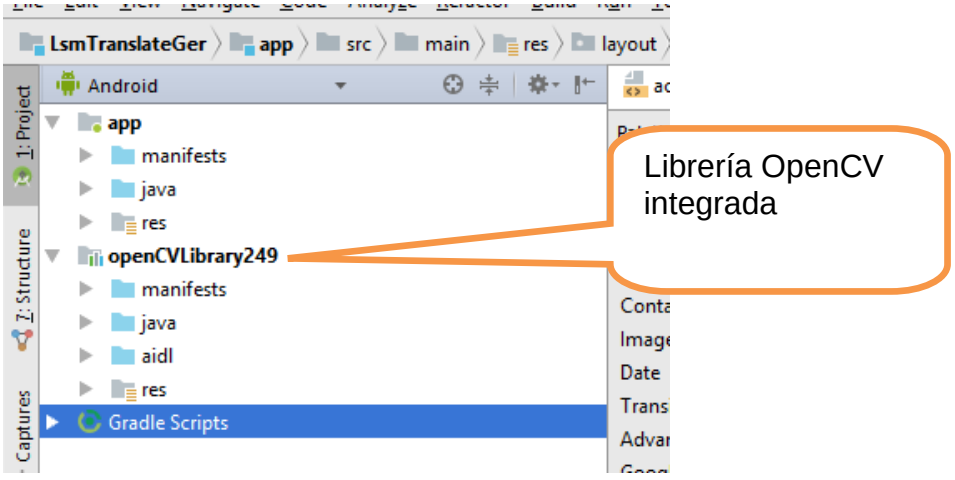


Figura 3. 18 Integración correcta de OpenCV en el proyecto

Una vez configurado OpenCV es necesario agregar los componentes (widgets) que tendrá la aplicación. En la figura 3.19 se muestra el diseño en el editor.

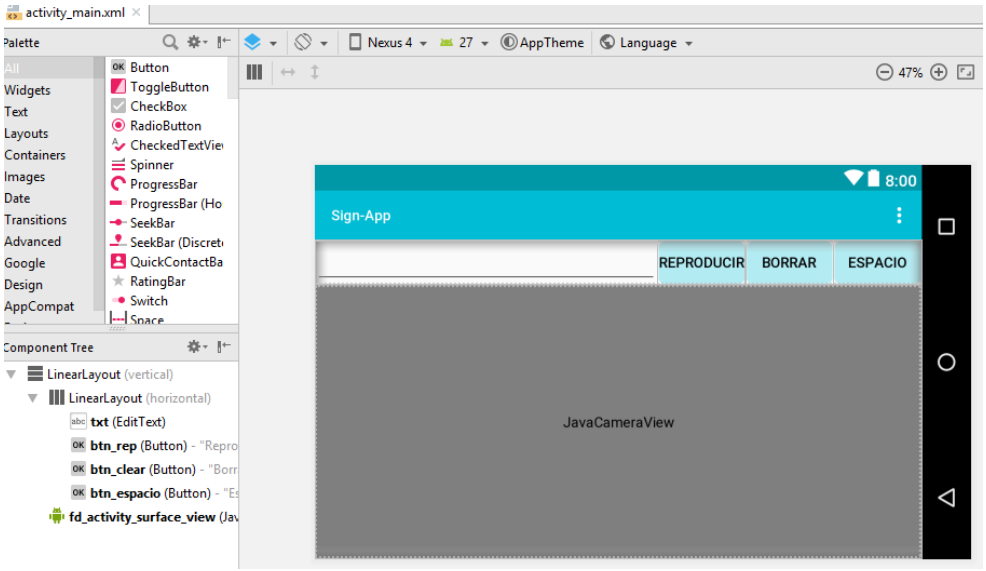


Figura 3. 19 Vista principal de la aplicación en IDE Android Studio

OpenCV proporciona una clase abstracta llamada **CameraBridgeViewBase** que representa un flujo de video de una cámara. Esta clase extiende de la clase de Android **SurfaceView**, por lo que las instancias de **CameraBridgeViewBase** pueden ser parte de la jeraquía de vistas. Ademas, una instancia de **CameraBridgeViewBase** puede mandar eventos a cualquier escuchador que implemente una de las dos interfaces **CvCameraViewListener** o **CvCameraViewListener2.** A menudo el escuchador será una actividad, como es este caso (figura 3.20).

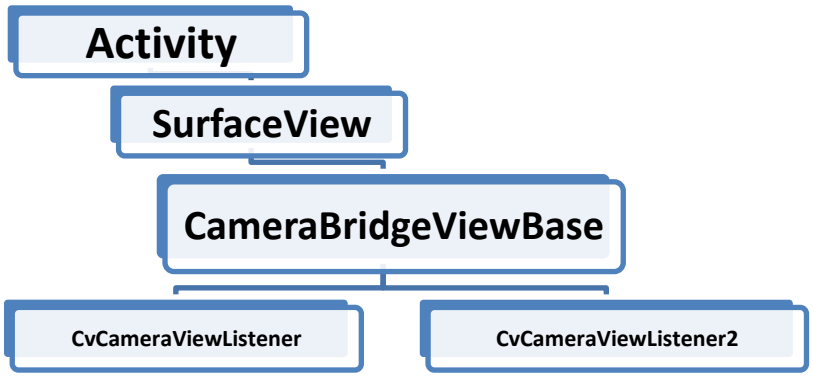


Figura 3. 20 Jerarquía de la clase CameraBridgeViewBase de OpenCV

Tanto **CvCameraViewListener** como **CvCameraViewListener2** proporcionan un mecanismo de callback para manejar el arranque y la parada de un flujo de video, asi como manejar la captura de cada fotograma. Cada vez que hay un fotograma disponible se ejecuta el método **onCameraFrame()**. Las diferencias entre ambos son los siguientes:

- **CvCameraViewListener** siempre recibe una imagen RGBA en color como un objeto de la clase Mat de OpenCV.
- **CvCameraViewListener2** recibe un objeto de la clase **CvCameraViewFrame** de OpenCV. Dicha clase permite almacenar una imagen en formato nativo de Android. A partir de este objeto es posible obtener una imagen en color o en gris de OpenCV mediante los métodos **rgba()** y **gray()** respectivamente.

Como **CameraBridgeViewBase** (clase de OpenCV) es una case abstracta, se debe realizar una implementación de la misma. OpenCV proporciona dos posibles implementaciones: **JavaCameraView** o **NativeCameraView.**

Ambas son clase de Java, pero **NativeCameraView** es un *wrapper* alrededor de una clase nativa C++. En este proyecto de utiliza **JavaCameraView**. En la figura 3.21 se muestra la utilización de esta clase en el diseño de la interfaz de la app.

```
40          <Button
41              android:id="@+id/btn_espacio"
42              android:layout_width="wrap_content"
43              android:layout_height="44dp"
44              android:layout_weight="1"
45              android:background="@drawable/myshape"
46              android:text="Espacio"/>
47
48
49      </LinearLayout>
50      <org.opencv.android.JavaCameraView
51          android:layout_width="fill_parent"
52          android:layout_height="fill_parent"
53          android:id="@+id/fd_activity_surface_view" />
54
55  </LinearLayout>
```

Figura 3. 21 Objeto JavaCameraView

Es importante señalar que para que el dispositivo pueda ejecutar aplicaciones que utilicen OpenCV es necesario tener instalada en el mismo la aplicación **OpenCV Manager**. Para soportar la interacción entre dicha aplicación y las aplicaciones cliente (nuestros programas) OpenCV proporciona una clase abstracta llamada BaseLoaderCallBack. Ésta clase declara un método callback que se ejecuta cuando OpenCV comprueba que la librería esta disponible.

Añadir permisos a la aplicación

En la app se requiere el acceso a la cámara, por lo que se añaden en archivo **AndroidManifest.xml** las líneas señaladas en la figura 3.22.

AndroidManifest.xml

manifest application activity

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="german.com.mx.lsmtranslateger">
4      <uses-permission android:name="android.permission.CAMERA"/>
5
6      <uses-feature android:name="android.hardware.camera"/>
7      <uses-feature android:name="android.hardware.camera.autofocus"/>
8      <uses-feature android:name="android.hardware.camera.front"/>
9      <uses-feature android:name="android.hardware.camera.front.autofocus"/>
```

Figura 3. 22 Permisos necesarios para la aplicación

Colores de la aplicación

En la figura 3.23 se muestran los colores utilizados en la aplicación (Material Desing).

colors.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="colorPrimary">#00BCD4</color>
4      <color name="colorPrimaryDark">#0097A7</color>
5      <color name="colorAccent">#CDDC39</color>
6  </resources>
```

Figura 3. 23 Colores utilizados en la aplicación

Añadir los clasificadores en cascada de las señas a la aplicación

Los archivos **.xml** generados, mismos que constituyen los clasificadores Haar necesarios para la detección de cada seña deben ser añadidos a la app, para ello se crea una carpeta de recursos llamada "**raw**". En la figura 3.24 se muestra esta carpeta. El archivo generado en la sección anterior se renombro como "letra_a.xml"

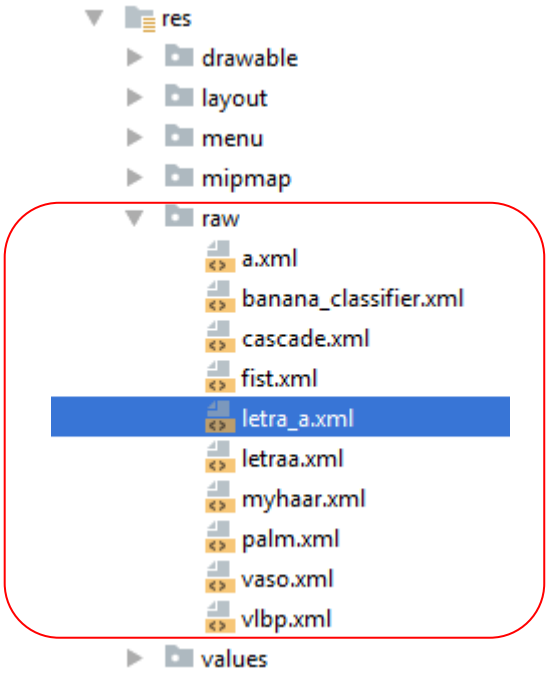


Figura 3. 24 Carpeta "raw" donde se colocan los clasificadores Haar generados

Metodología utilizada para el desarrollo de la aplicación

Mobile-D [36] es una metodología de desarrollo ágil para sistemas móviles la cual contiene una mezcla de muchas técnicas utilizadas para la elaboración de software.

El objetivo es conseguir ciclos de desarrollo muy rápidos en equipos muy pequeños. Fue creado en un proyecto finlandés en 2005, pero sigue estando vigente. Basado en metodologías conocidas pero aplicadas de forma estricta como: extreme programming, Crystal Methodologies y Rational Unified Process.

Un ciclo de proyecto con la metodología Mobile-D está compuesto por cinco fases (figura 3.25):

- Fase de Exploración

Esta fase es la encargada de la planificación y educción de requisitos del proyecto, donde se tiene la visión completa del alcance del proyecto y también todas las funcionalidades del producto.

- Fase de inicialización

Es la implicada en conseguir el éxito en las próximas fases del proyecto, donde se preparará y verificará todo el desarrollo y todos los recursos que se necesitarían. Esta fase se divide en cuatro etapas: la puesta en marcha del proyecto, el dia de planificación inicial, el día de prueba y día de salida.

- Fase de producción

Se vuelve a repetir la programación de los tres días, iterativamente hasta montar (implementar) las funcionalidades que se desean. Aquí se utiliza el desarrollo dirigido por pruebas (TDD), para verificar el correcto funcionamiento de los desarrollos.

.

- Fase de estabilización

Se llevarán a cabo las últimas acciones de integración donde se verificará el completo funcionamiento del sistema en conjunto. De toda la metodología, esta es la fase más importante de todas ya que es la que asegura la estabilización del desarrollo. También se puede incluir en esta fase, toda la producción de documentación.

- Fase de pruebas

Es la fase encargada de las pruebas de la aplicación una vez terminada. Se deben realizar todas las pruebas necesarias para tener una versión estable y final. En esta fase, si se encuentra algún tipo de error, se debe proceder a su arreglo pero nunca se han de realizar desarrollos nuevos de última hora, ya que rompería todo el ciclo.

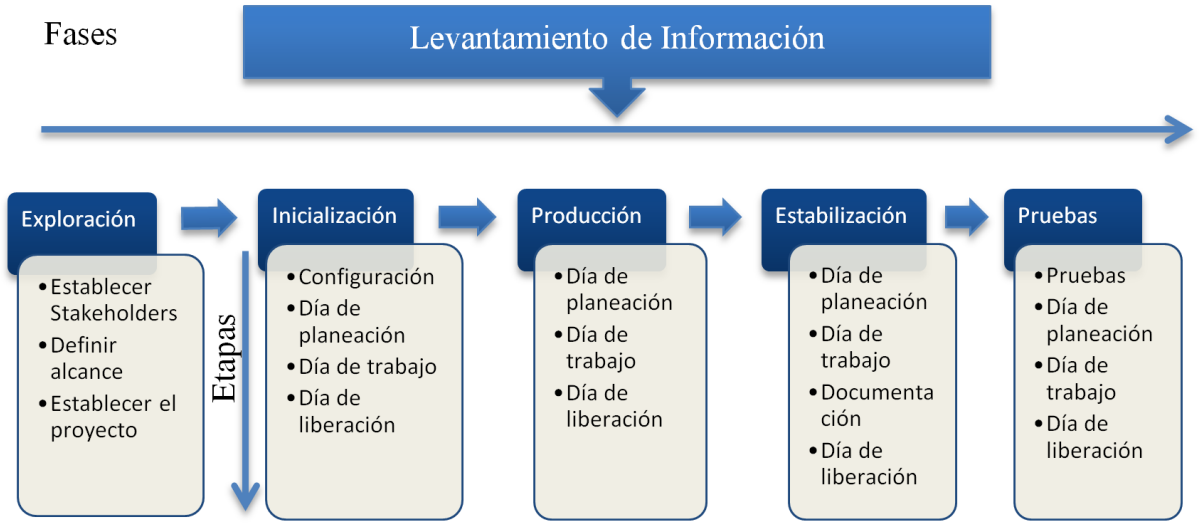


Figura 3. 25 Fases de la metología Mobile-D

Fuente: http://pegasus.javeriana.edu.co/~PA133-05-PMovVidaAutomotor/Metodologia.html

Programación de la funcionalidad de la aplicación

Se utiliza el paradigma de Programacipon Orientado a Objetos (OOP) sobre el que esta basado el lenguaje de programación Java. Mismo que es utilizado para el desarrollo de aplicaciones nativas con Android Studio (en la versión 3 del IDE se incluye el soporte para Kotlin, no obstante en este proyecto se utiliza Java).

Una vez creado el diseño de la vista, sigue la programación de la lógica de la aplicación. En la actividad (Activity) donde tenemos el objeto JavaCameraView agregamos las instrucciones necesarias tanto de programación nativa con Android como de OpenCV.

El primer paso para la detección de señas a través de OpenCv es implementar la interfaz **CvCameraViewListener2:**

```
public class MainActivity extends AppCompatActivity implements CvCameraViewListener2
```

Despues de necesario cargar los clasificadores Haar, en la figura 3.26 de muestra como ejemplo el código para cargar el clasificador de la letra A de la carpeta **raw**.

```
// cargar el clasificador en cascada de los recursos
InputStream iseA = getResources().openRawResource(R.raw.letra_a);
File cascadeDirA = getDir( name: "cascade", Context.MODE_PRIVATE);
mCascadeFileA = new File(cascadeDirA, child: "letra_a.xml");
FileOutputStream osA = new FileOutputStream(mCascadeFileA);

while ((bytesRead = iseA.read(buffer)) != -1) {
    osA.write(buffer, off: 0, bytesRead);
}
iseA.close();
osA.close();
////////////////////////////////////////////////////////////////////////////////////////////////
mJavaDetectorA = new CascadeClassifier(mCascadeFileA.getAbsolutePath());
if (mJavaDetectorA.empty()) {
    Log.e(TAG, msg: "Failed to load cascade classifier for eye");
    mJavaDetectorA = null;
} else
    Log.i(TAG, msg: "Loaded cascade classifier from " + mCascadeFileA.getAbsolutePath());
```

Figura 3. 26 Carga del clasificador de la letra A en la app

Al cargar la interfaz (activity) es necesario inicializar el visor de la cámara para poder obtener las imágenes captadas en la aplicación, para ello inicializamos el objeto JavaCameraView (figura 3.27) donde adicionalmente configuramos un índice para saber si se utiliza la cámara frontal o la trasera.

```
mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.fd_activity_surface_view);
mOpenCvCameraView.setCvCameraViewListener(this);
if (savedInstanceState != null) {
    indiceCamara = savedInstanceState.getInt(STATE_CAMERA_INDEX, defaultValue: 0);
} else {
    indiceCamara = 0;
}

mOpenCvCameraView.setCameraIndex(indiceCamara);
```

Figura 3. 27 Inicializar objeto JavaCameraView

Ya que se cargaron los clasificadores es necesario habilitar el visor de la cámara para esto usamos la intruccion: **mOpenCvCameraView**.enableView();

Despues es necesario detectar cada frame (imagen) obtenida de la cámara de móvil para y procesarla para obtenerla en escalda de grises, con la que trabaja el casificador Haar. En la figura 3.28 se muestra este código, facilitado con OpenCV.

```
public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {

    mRgba = inputFrame.rgba();//Guardar imagen recibida a color
    mGray = inputFrame.gray();//Guardar imagen recibida en escala de grises
```

Figura 3. 28 Obtener frame y procesarlo a escala de grises con OpenCV

Una vez procesado el frame es necesario buscar la seña en él, para esto se utiliza el método **detectMultiScale()** del objeto CascadeClassifier (figura 3.29).

```
//intentar detectar la segunda seña: la letra A
MatOfRect letrasA = new MatOfRect();
if (mDetectorType == JAVA_DETECTOR) {
    if (mJavaDetectorA != null)
        mJavaDetectorA.detectMultiScale(mGray, letrasA, scaleFactor: 1.1, minNeighbors: 2, flags: 2,
                new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new Size());
} else {
    Log.e(TAG, msg: "No hay un metodo de deteccion!");
}
Rect[] AArray = letrasA.toArray();
Log.e(TAG, msg: "Num de A detectadas: " + AArray.length);
```

Figura 3. 29 Buscar la seña A en al Frame recibido

Si se detecta alguna seña en el frame se lanza un Hilo en segundo plano para colocar la letra “A” en el EditText ubicado en la parte superior de la aplicación para poder formar alguna palabra y posteriormente reproducirla con voz artificial. El código se muestra en la imagen 3.30.

```
if (AArray.length > 0) {//encontro una A
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            String curr = txt.getText().toString();

            if(curr!=null && !"".equals(curr)) {

                int p = curr.length() - 1;
                if (curr.charAt(p) != 'a') {
                    txt.setText(curr + "a");
                }
            }else{
                txt.setText("a");
            }
        }
    });
}
```

Figura 3. 30 Colocar la letra detectada en el EditText

Como parte del proceso de detección también se le indica al usuario en la pantalla de la aplicación la seña detectada, resaltada con un cuadro y un texto descriptivo. Esta descripción se realiza sobre la imagen original (a color). En la figura 3.31 se muestra el código necesario.

```
for (int i = 0; i < AArray.length; i++) {
    Core.rectangle(mRgba, AArray[i].tl(), AArray[i].br(),
            FACE_RECT_COLOR, thickness: 3);

    Core.putText(mRgba, text: "a", AArray[i].tl(), fontFace: 1, fontScale: 2,
            new Scalar( v0: 255, v1: 0, v2: 0, v3: 255), thickness: 3);
    // xCenter = (facesArray[i].x + facesArray[i].width + facesArray[i].x)
    //yCenter = (facesArray[i].y + facesArray[i].y + facesArray[i].height)
    //Point center = new Point(xCenter, yCenter);
}
////////////////////////////////////////
return mRgba;
```

Figura 3. 31 Mostrar la seña detectada en la pantalla del móvil

La visualización de la seña “a” una vez detectada en la aplicación móvil se muestra en la figura 3.32

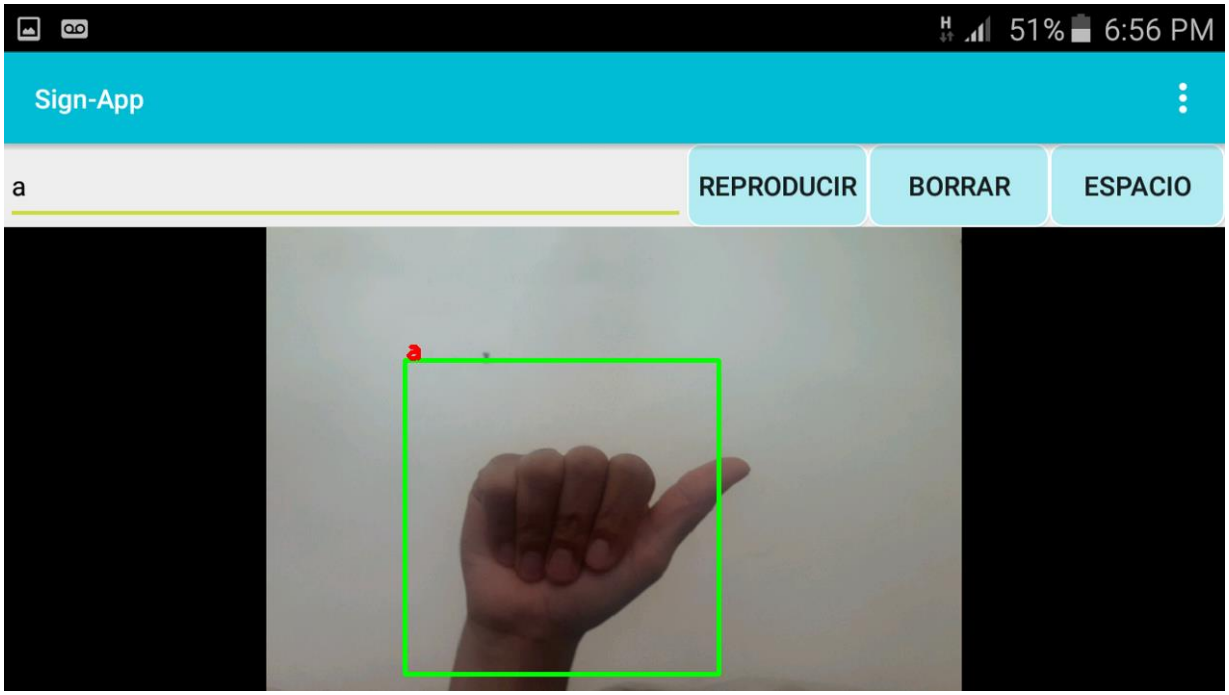


Figura 3. 32 Detección de señas en tiempo real (letra A)

Para cada una de las señas que se busque detectar es necesario el mismo proceso:

1. Colocar el clasificador Haar (xml) generado en el proyecto, es decir, al IDE Android Studio.
2. Cargar el clasificador al cargar o iniciar la aplicación.
3. Realizar la búsqueda de la seña en los frames (imágenes) obtenidos de la cámara del móvil.
4. Una vez detectada la seña colocar el texto de dicha letra en el campo de entrada.
5. Una vez detectada la seña mostrar un cuadro indicando a la persona la seña encontrada.
6. Si se quiere reproducir el texto colocado en el campo de entrada hacer clic en el botón “Reproducir”.

Al estar integrada la librería OpenCV solo es necesario incluir los clasificadores y manipular sus resultados tal como se mostro en los pasos anteriores.

3.5 Implementación de Voz artificial en la aplicación

Como se mencionó anteriormente la lógica interna utilizada por el motor de síntesis de voz es bastante compleja. Sin embargo, la plataforma Android permite integrar en cualquier aplicación este sistema de una forma sencilla sin necesidad de que el desarrollador conozca su metodología. Para la implementación de una aplicación basada en la síntesis de texto a voz, únicamente es necesario que ésta se encargue de enviar el texto que se quiere leer en voz alta al motor de síntesis para que lo reproduzca. Esta funcionalidad se consigue a partir del paquete **android.speech.tts** de la API de Android, en particular, la clase **android.speech.tts.TextToSpeech**. Para usar esta clase, necesita instanciar un objeto de esta clase y también especificar el "initListener", en la figura 3.33 se muestra la declaración e inicialización de este objeto.

```
45      TextToSpeech tts;
169     tts = new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {
170         @Override
171         public void onInit(int status) {
172
173             if (status != TextToSpeech.ERROR) {
174                 Locale locSpanish = new Locale( language: "spa",  country: "MEX");//lenguaje español
175                 tts.setLanguage(locSpanish);
176             }
177         }
178     });
```

Figura 3. 33 Inicialización del motor TTS en la aplicación

Para realizar la lectura de texto con voz artificial se manda llamar el método **speak()** del objeto **TextToSpeech**. En este caso se invoca a este método al hacer clic en el botón "hablar" de la interfaz. En la figura 3.34 se muestra este proceso.

```
    btn_rep.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String texto = txt.getText().toString();//texto traducido de las señas
            tts.speak(texto, TextToSpeech.QUEUE_FLUSH,  params: null);//voz artificial
        }
    });
}
```

Figura 3. 34 Reproducción de texto con voz artificial

Para cada una de las señas se aplica el mismo procedimiento. Si el usuario de la aplicación desea dar un espacio una vez formada alguna palabra solo deberá presionar el botón “Espacio”. Para borrar el texto solo deberá presionarl el botón “Borrar”.

Adicionamente la aplicación cuenta con un menú de opciones, el que la persona puede cambiar la resolución de pantalla y cambir la cámara utilizada para captar las señas (figura 3.35).

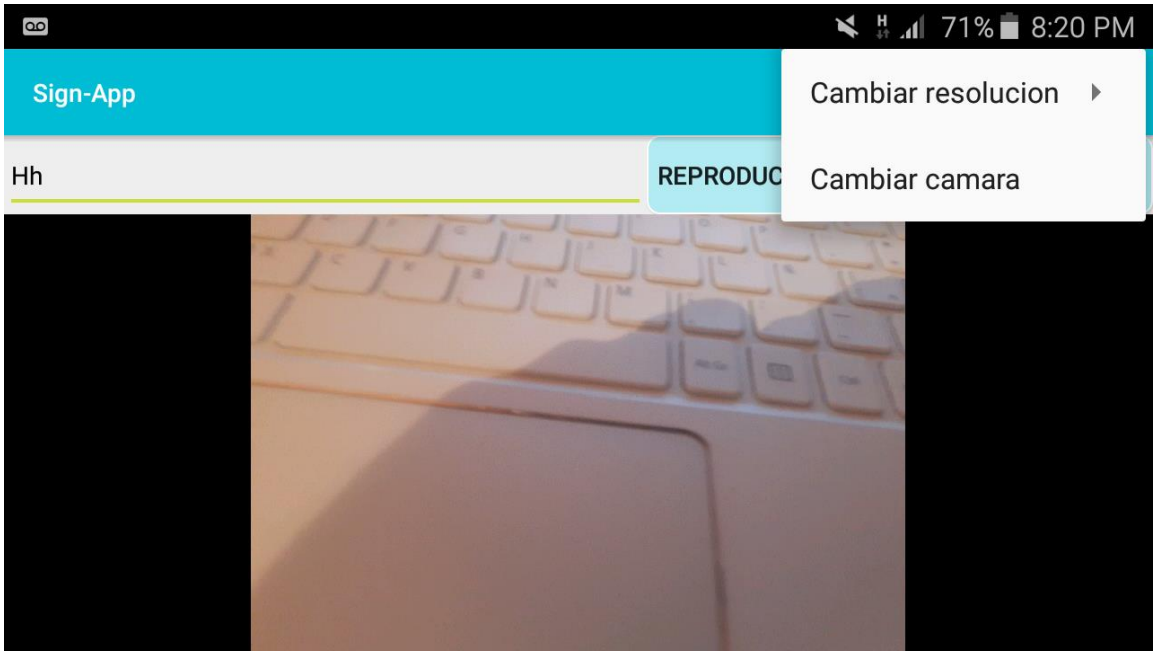


Figura 3. 35 Menu de opciones de la app

Las diferentes resoluciones de pantalla de muestran en la figura 3.36. La finalidad de estas opciones es facilitar al usuario el cambio de calidad para mejorar el procesamiento y captación de las imágenes de la cámara.

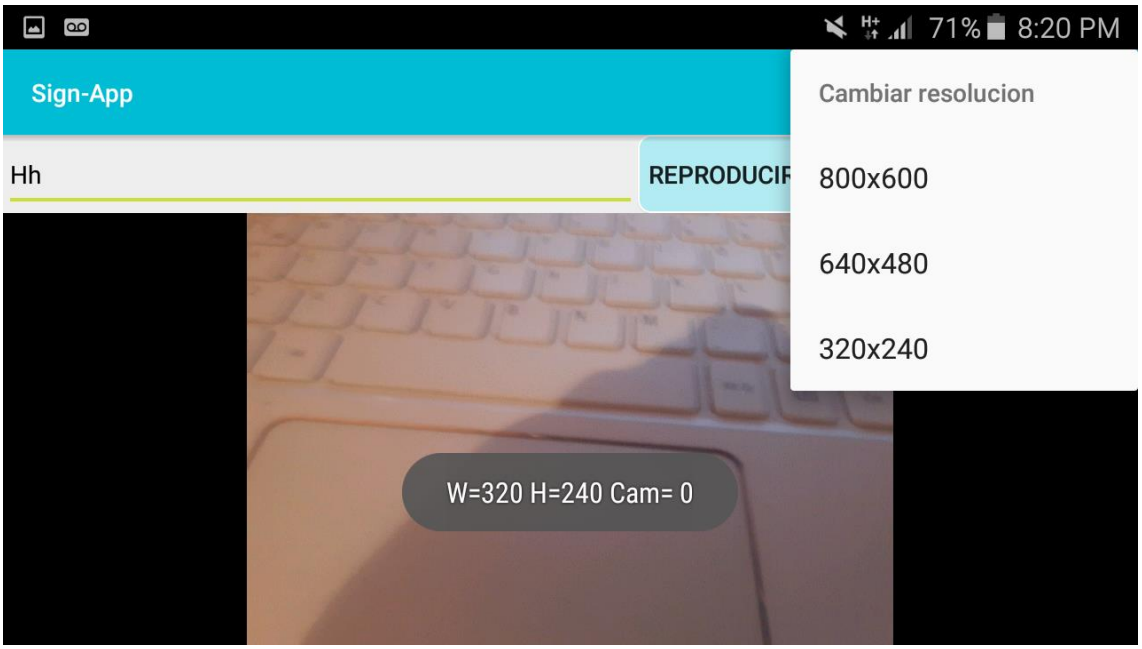


Figura 3. 36 Diferentes resoluciones de pantalla disponibles

Capítulo 4 ANALISIS Y RESULTADOS

En este capitulo se muestran los resultados del uso de los clasificadores Haar y de la aplicación propuesta, se describen dos tipos de pruebas: En primer lugar se analiza la detección con la herramienta “Cascade Trainer-GUI” utilizada para crear el clasificador, dicho programa permite ademas probar el clasificador tanto en imágenes con en video. Y en segundo lugar se describen las pruebas realizadas mediante capturas de la aplicación en tiempo de ejecución, es decir, se prueba el clasificador a través de la app.

Dichas pruebas se realizaron con diferentes fondos claros, esto con la finalidad de verificar la correcta detección de las señas generadas. Para efectos de estar en condiciones de formar palabras básicas, basada en señas estaticas se generaron los clasificadores de las letras A, M, H, L, y O.

Es importante considerar que al abrir la app se inicia la recepción de un flujo continuo de frames o imágenes obtenidad por la cámara (la cantidad de frames por segundo o FPS es un valor que depende del dispositivo en cuestión) . El proceso de detección se aplica en cada una de estas imágenes, es decir, en cada frame se busca cada patron de la señas, una vez detectada alguna ya no se buscan las demás.

Por lo anterior se tiene que considerar que solo se coloque la letra una vez (aunque se encuentre en varios frames), para lograrlo se coloca el texto de la letra solo si es distinta a la ultima detectada, por ejemplo, si ya se detecto una A, no se debe volver a colocar, a menos que haya un espacio antes. No obstante esta condición requiere algunas consideraciones, tales como calidad de la detección y asegurar en que la seña se encuentre en varios frames para considerla como una detección exitosa.

Ahora se desciben los resultados de detección de las señas mencionadas, es decir, de las señas que se creo un clasificador Haar. Se analizan por separado para visualizar

mejor la interfaz de la app y proporcionar algunos detalles específicos de cada una de ellas.

Letra L

El clasificador de esta letra fue generado con 63 imágenes positivas y 1000 negativas. A través del programa Cascade Trainer-GUI el clasificador fue probado, para esto se colocarón varias imágenes con la seña, sobre las que el clasificador busca la seña. En la figura 4.1 se muestra esta prueba, en la que observamos que algunas de las señas no pudieron ser detectadas, este proceso depende de la cantidad de muestras utilizadas para crear el clasificador.

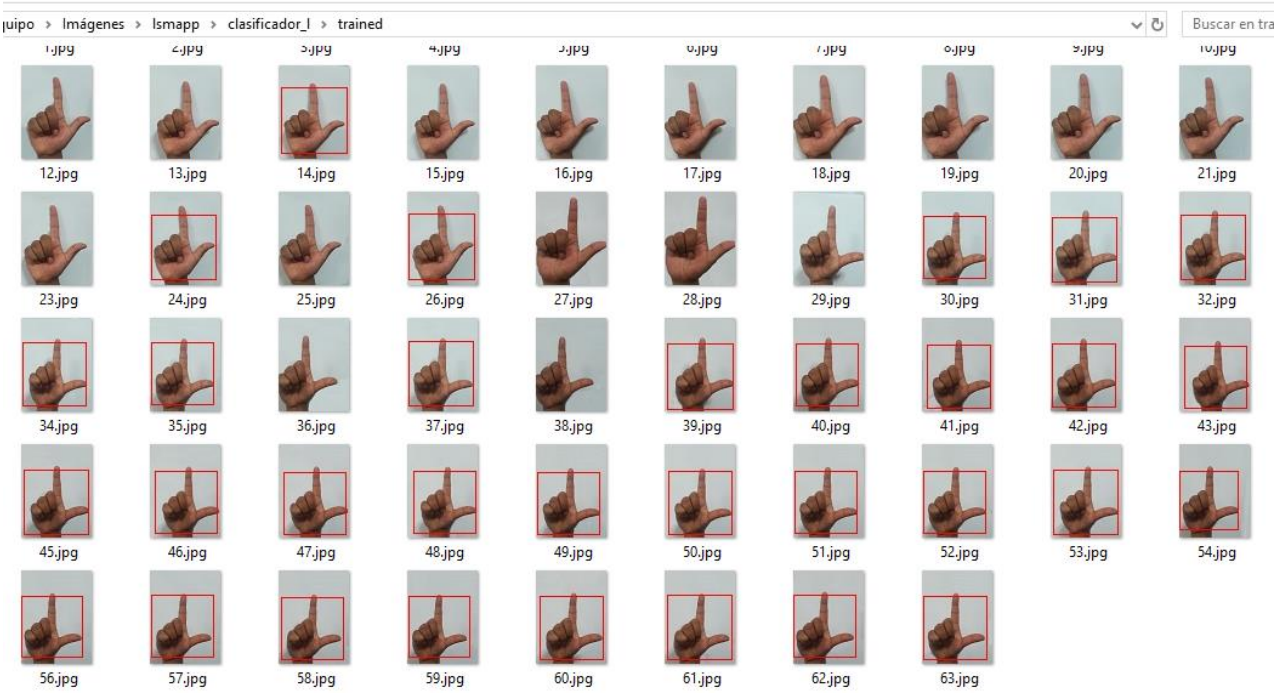


Figura 4. 1 Prueba en imagenes del clasificador de la letra L

Una vez incluido el clasificador en el dispositivo móvil, se probaron con varios fondos en 5 personas con el propósito de probar la aplicaion móvil.Como resultados de las pruebas se tienen los siguientes resultados.

Fondo 1: Uso de fondo blanco, este es el fondo básico y mas importante.

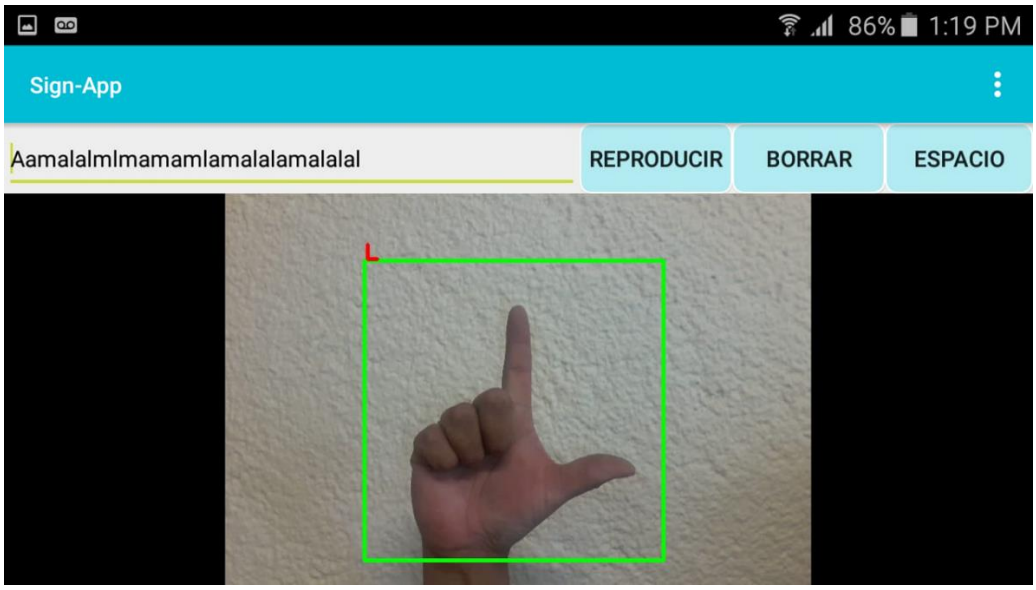


Figura 4. 2 Deteccion de letra L en fondo 1

Fondo 2: En este caso se utilizó un fon con color rosa (figura 4.3).

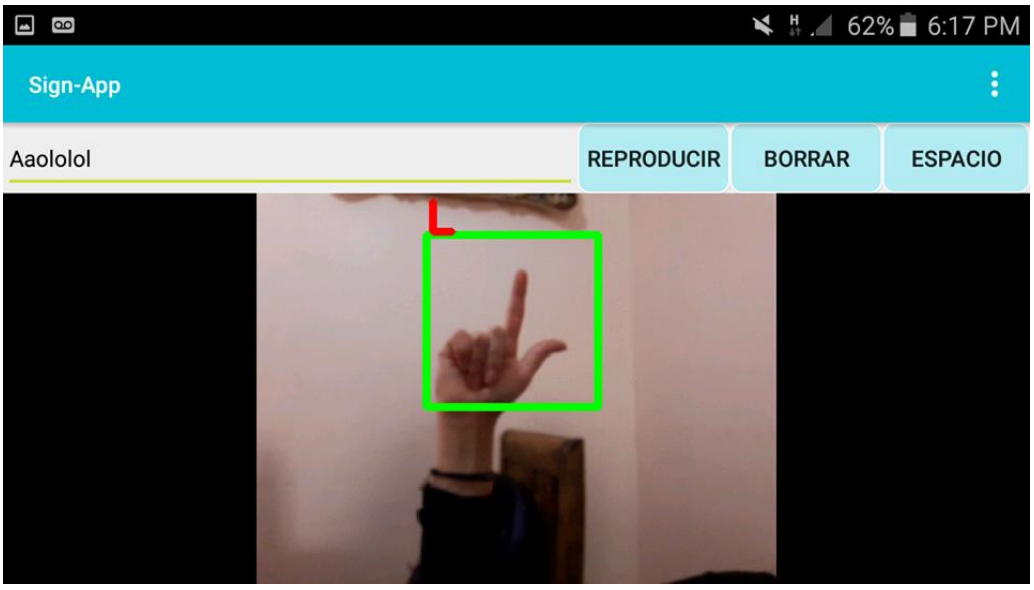


Figura 4. 3 Deteccion de letra L en fondo 2

Fondo 3: aquí se utiliza un fondo amarillo con poca luz (figura 4.4)

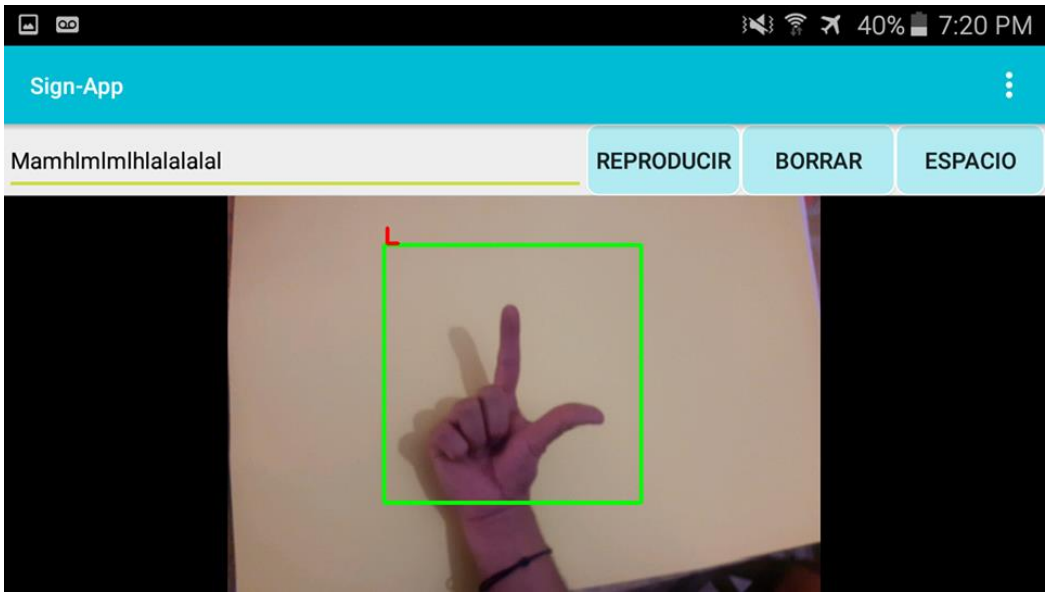


Figura 4. 4 Deteccion de letra L en fondo 3

Fondo 4: Para este fondo se definio un colo azul (figura 4.5)

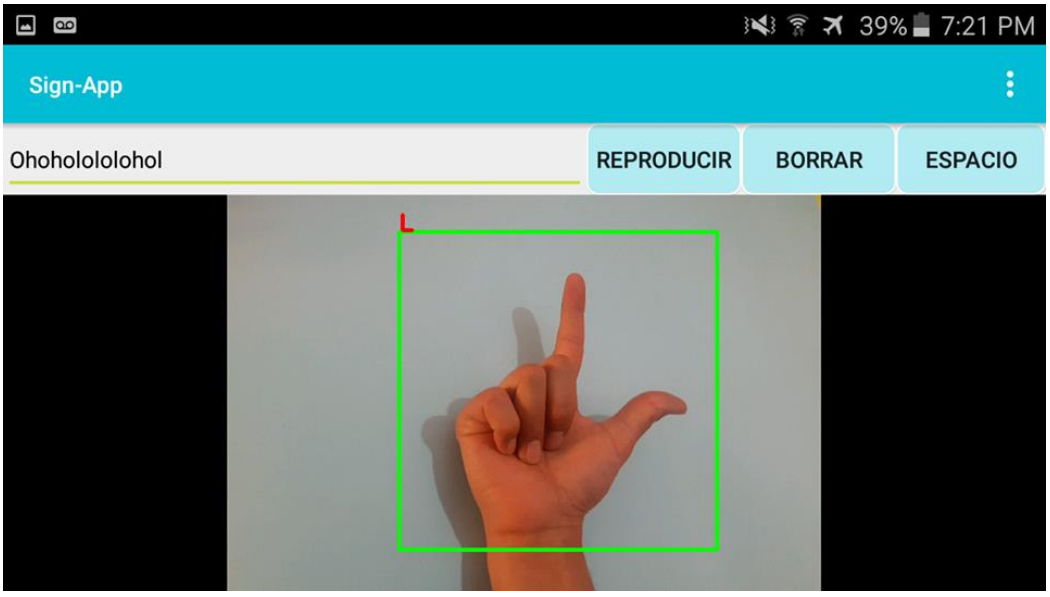


Figura 4. 5 Deteccion de letra L en fondo 4

Fondo 5: En este caso se utilizó un fondo con color blanco (figura 4.6).

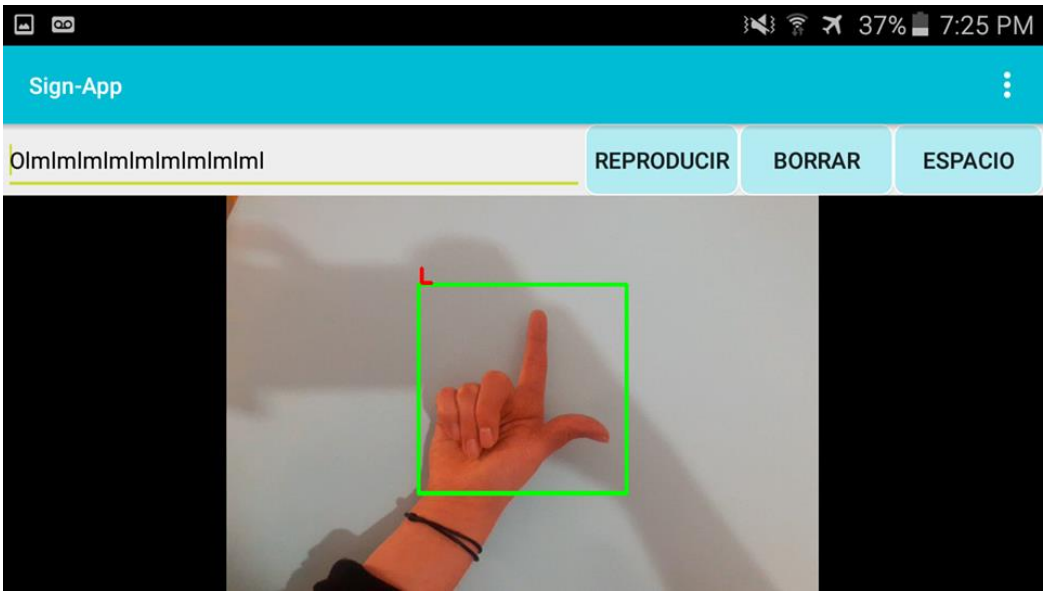


Figura 4. 6 Deteccion de letra L en fondo 5

Se observa una detección exitosa en fondos claros, esto se debe a la utilización de este color para la creación del clasificador. Para el proceso de detección OpenCV utiliza un fondo en escala de grises, por lo que resulta mas sencillo y rápido utilizar este color para verificar la prueba de detección.

Letra H

El clasificador de esta letra fue generado con 61 imágenes positivas y 1000 negativas. Las pruebas del clasificador en el software Cascade Trainer-GUI se muestran en la imagen.

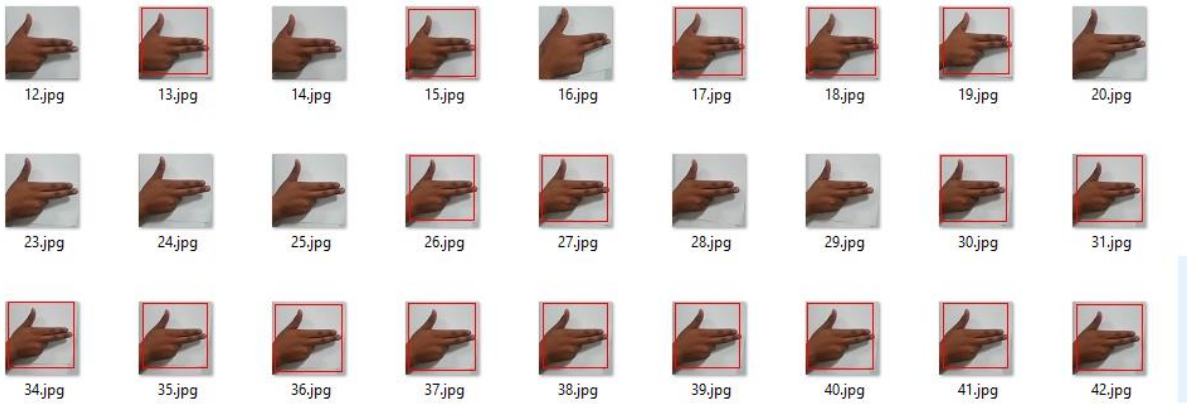


Figura 4. 7 Prueba en imagenes del clasificador de la letra H

Como resultados de las pruebas en la aplicación se tienen los siguientes resultados:

Fondo 1: pared blanca con relieves (Figura 4.8)

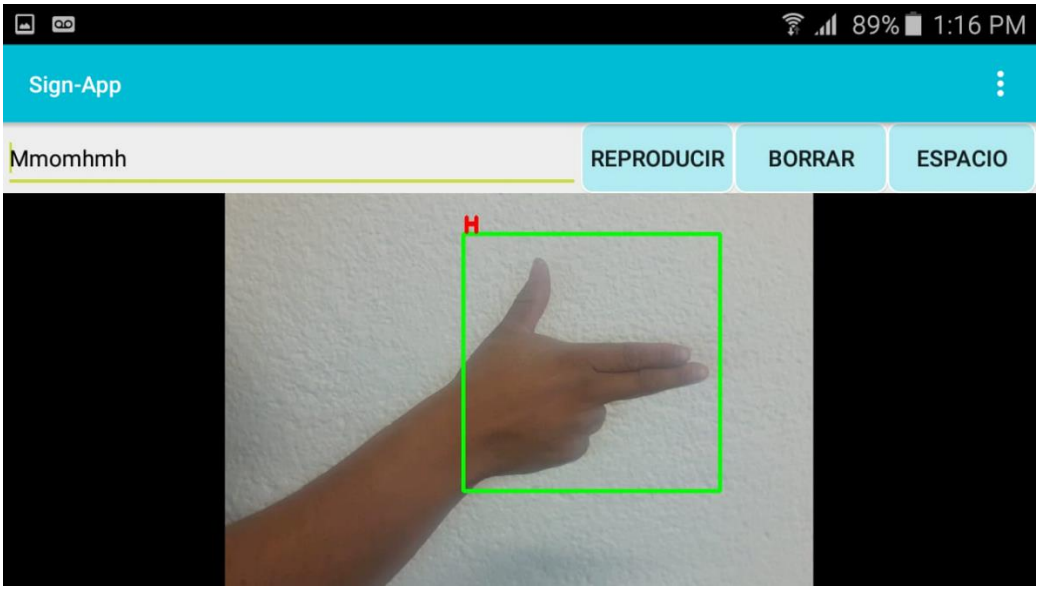


Figura 4. 8 Deteccion de letra H en fondo 1

Fondo 2: En este caso se utilizó una pared con color rosa (figura 4.9).

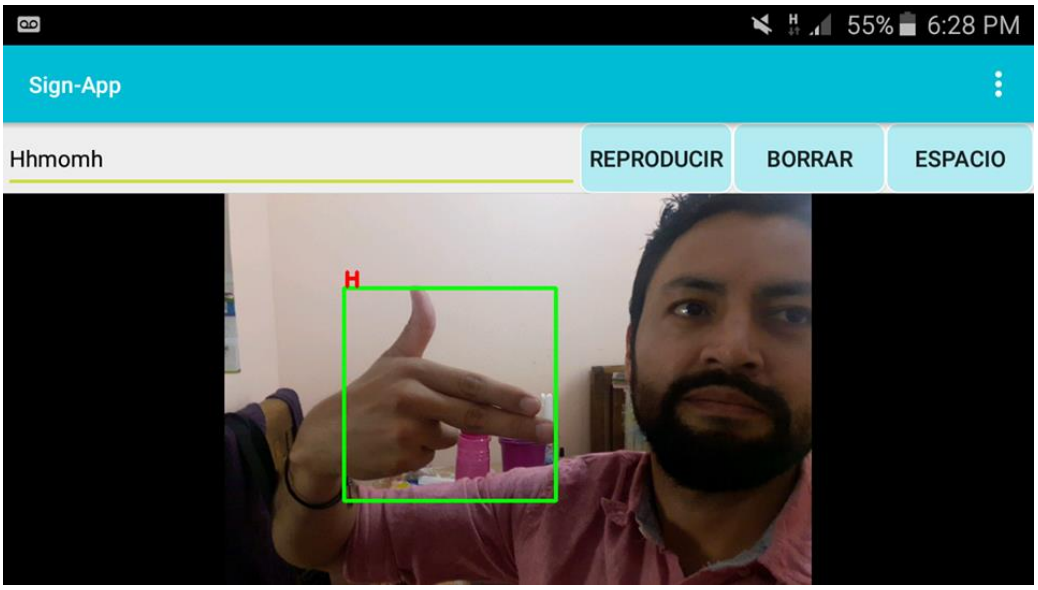


Figura 4. 9 Deteccion de letra H en fondo 2

Fondo 3 En este caso se utilizó un fondo con color blanco (figura 4.10).

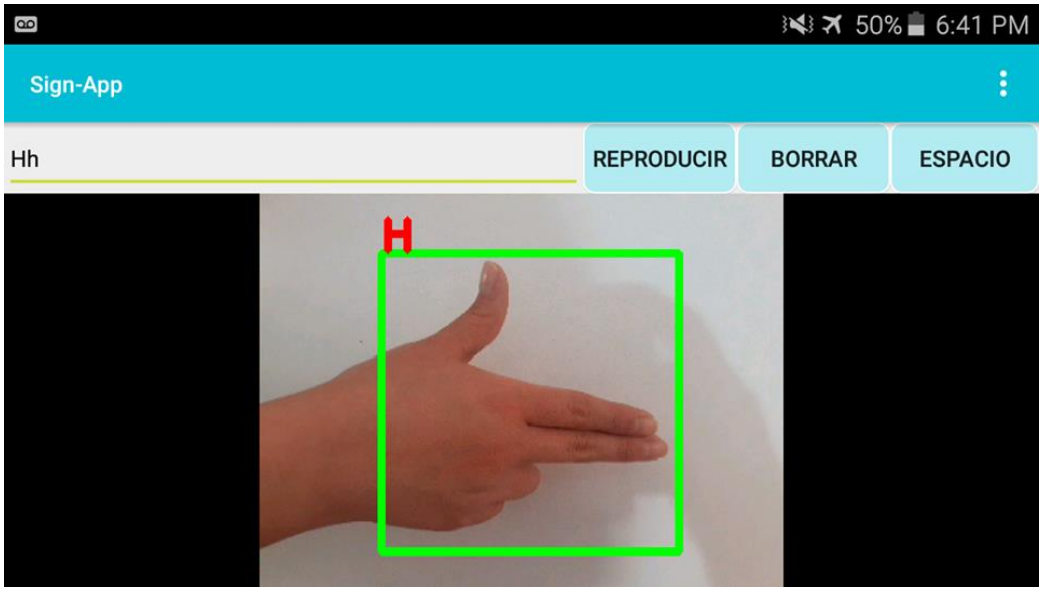


Figura 4. 10 Deteccion de letra H en fondo 3

Fondo 4: Para este fondo se definio un colo azul (figura 4.22)

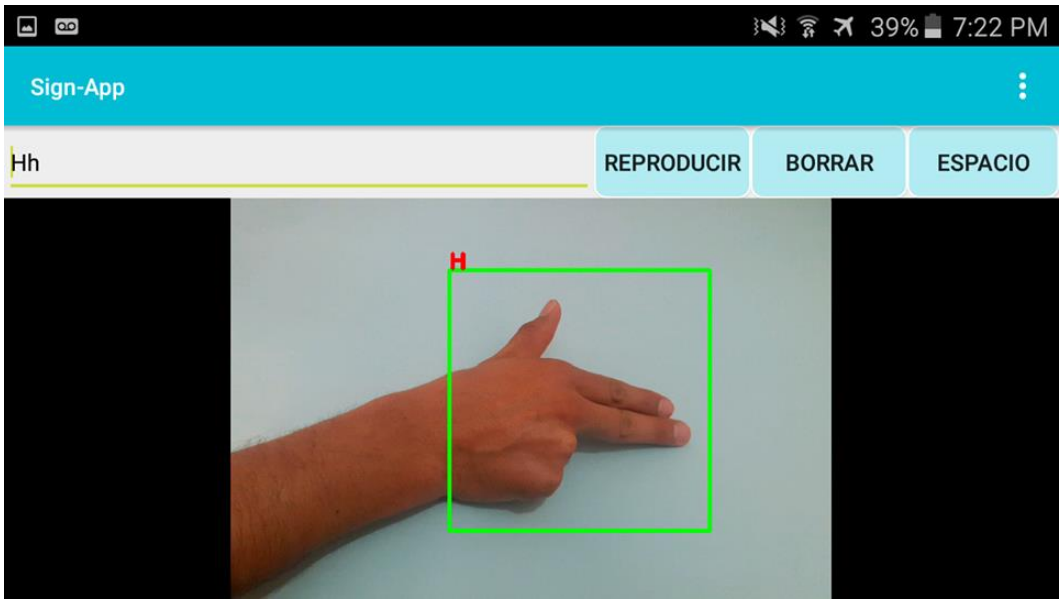


Figura 4. 11 Deteccion de letra H en fondo 4

Fondo 5: Aquí se utilizó el fondo amarillo (Figura .12)

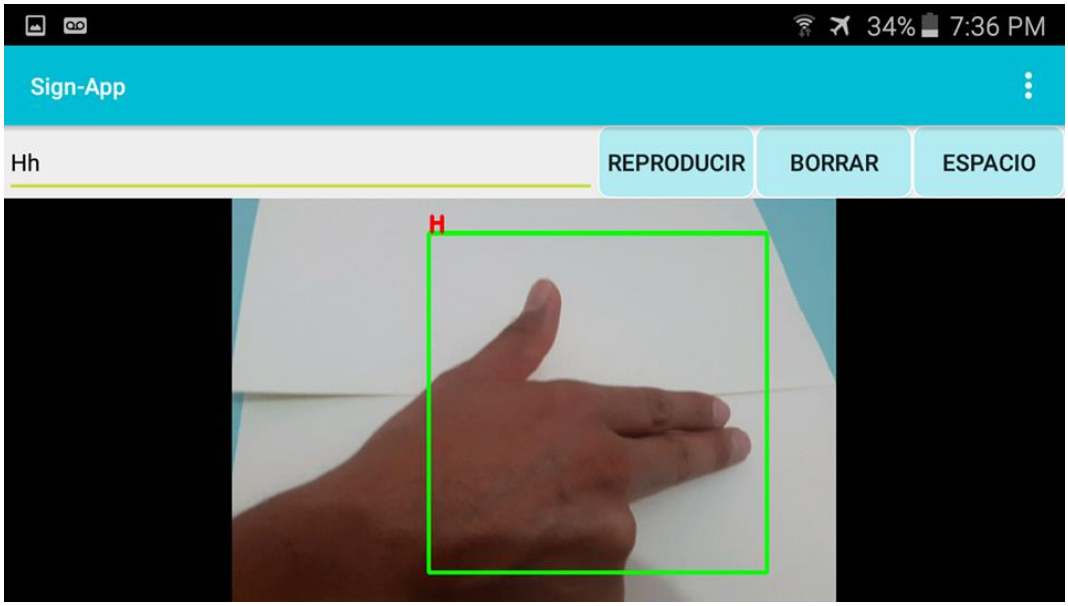


Figura 4. 12 Deteccion de letra H en fondo 5

Letra A

El clasificador de esta letra fue generado con 102 imágenes positivas y 500 negativas. El resultado de las pruebas del clasificador en imágenes de ejemplo se muestran en la figura 4.13.

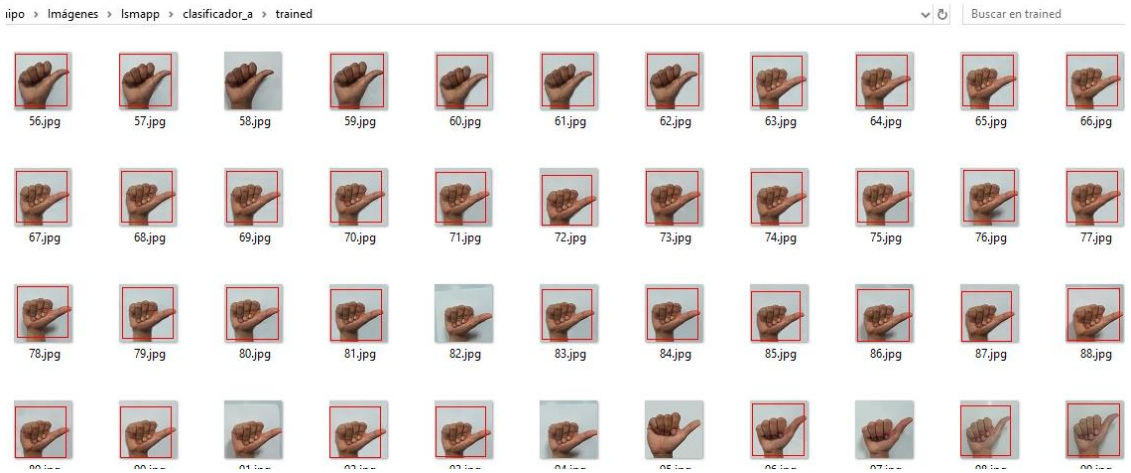


Figura 4. 13 Prueba en imagenes del clasificador de la letra A

En las pruebas desde la aplicación móvil Android se tienen los siguientes resultados:

Fondo 1: Este fondo es blanco, con un poco de brillo por la luz.

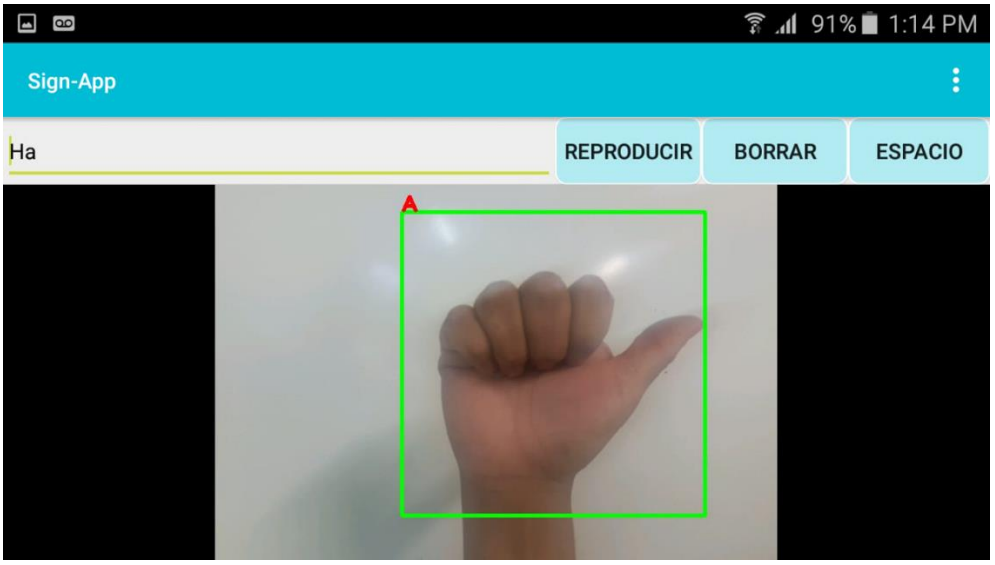


Figura 4. 14 Deteccion de letra A en fondo 1

Fondo 2: Este fondo es una pared rosa con un poco de objetos adicionales (Figura 4.15).

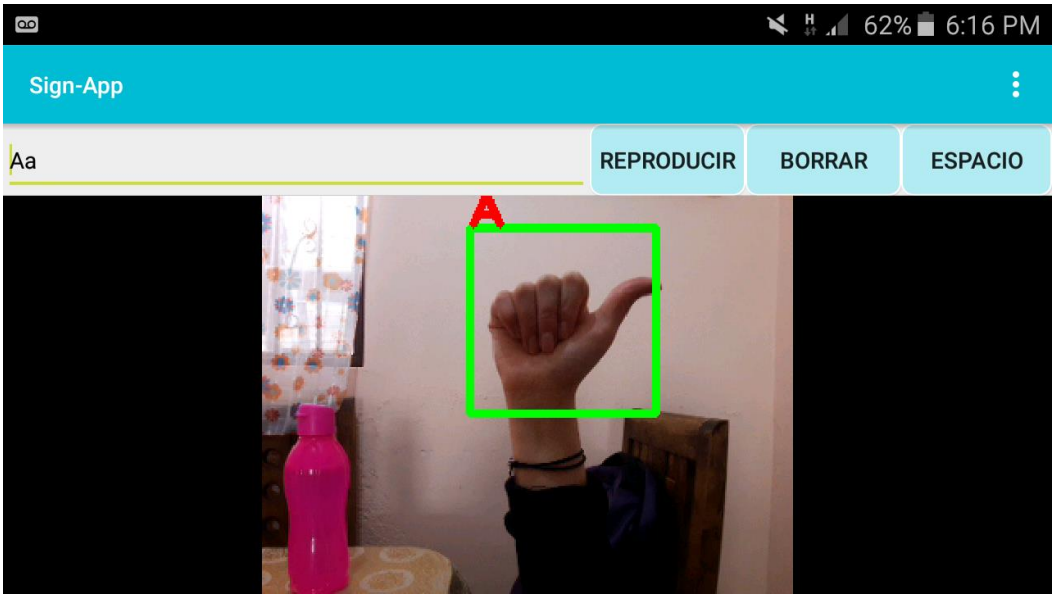


Figura 4. 15 Deteccion de letra A en fondo 2

Fondo 3: En este caso es un fondo azul claro (figura 4.16).

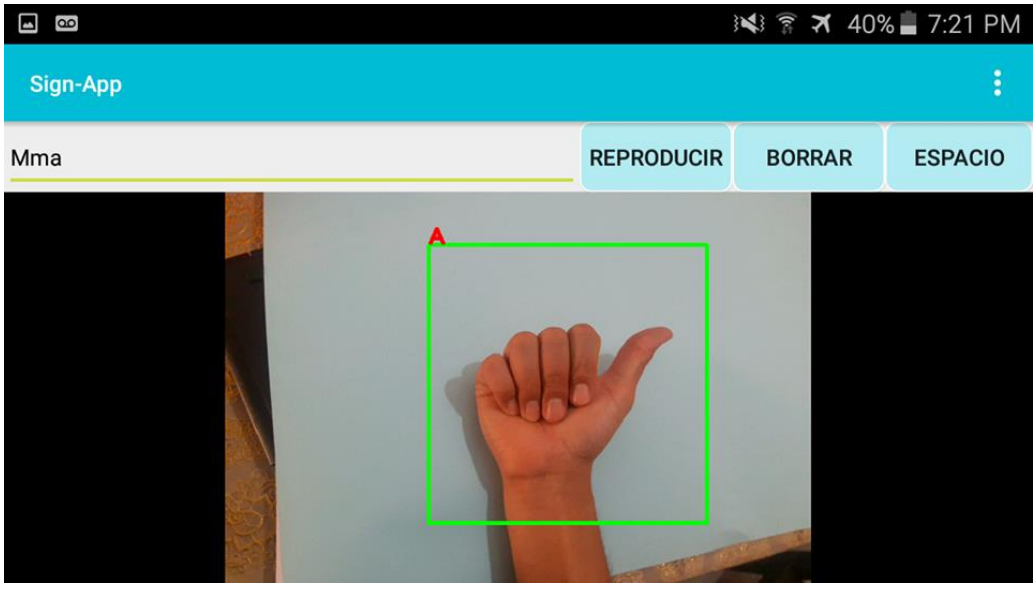


Figura 4. 16 Deteccion de letra A en fondo 3

Fondo 4: En esta ocasión se prueba sobre un fondo blanco (figura 4.17).

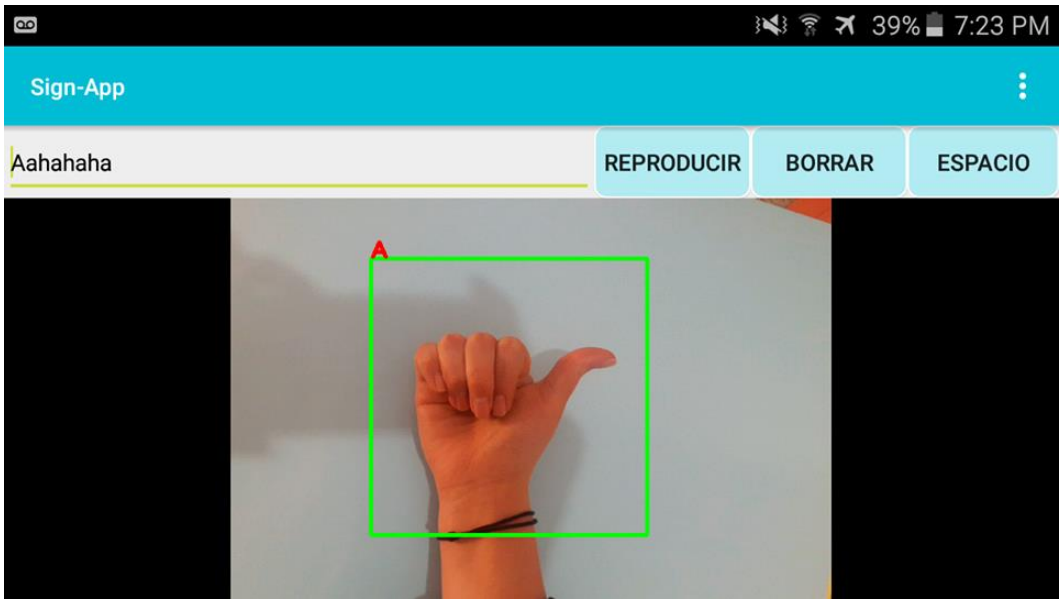


Figura 4. 17 Deteccion de letra A en fondo 4

Fondo 5: En este caso se prueba con un fondo amarillo o beige (figura 4.18).

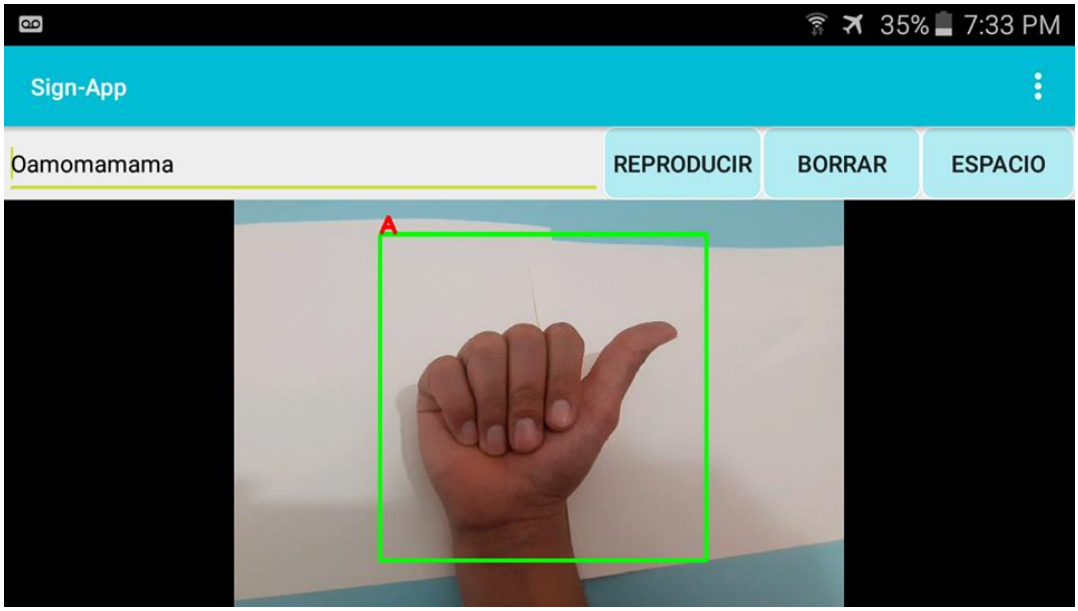


Figura 4. 18 Deteccion de letra A en fondo 5

Letra O

El clasificador de esta letra fue generado con 63 imágenes positivas y 1000 negativas. Los resultados de la pruea del clasificador sobre una lista de imágenes predefinidas y fijas se muestran en la siguiente figura:

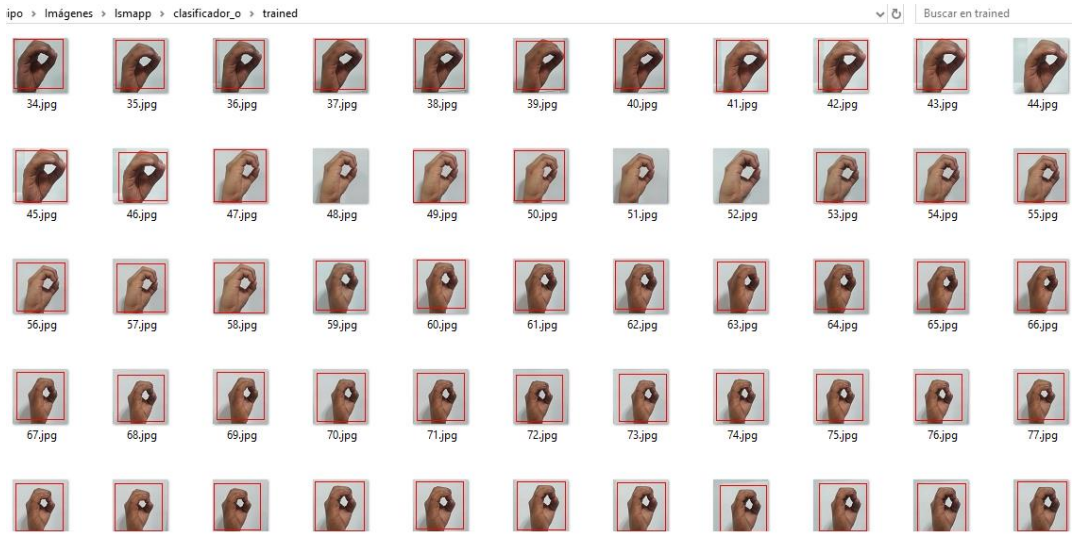


Figura 4. 19 Prueba en imagenes del clasificador de la letra O

Los resultados de las pruebas en la aplicación se muestran en las figuras siguientes.
Fondo 1: Aquí se usa un fondo blanco con un poco de brillo (figura 4.20).

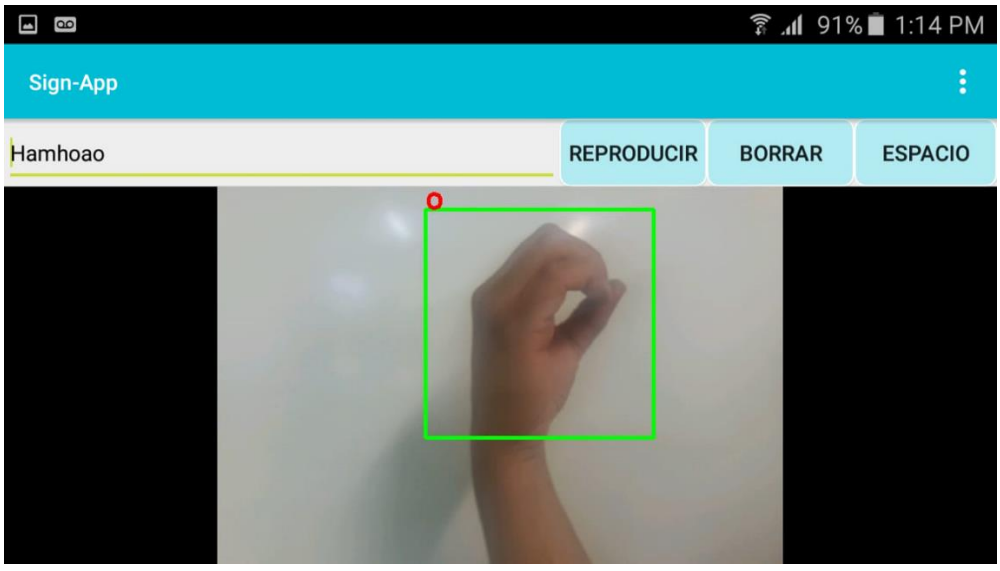


Figura 4. 20 Deteccion de letra O en fondo 1

Fondo 2: Este fondo es una pared rosa con un poco de objetos (Figura 4.21).

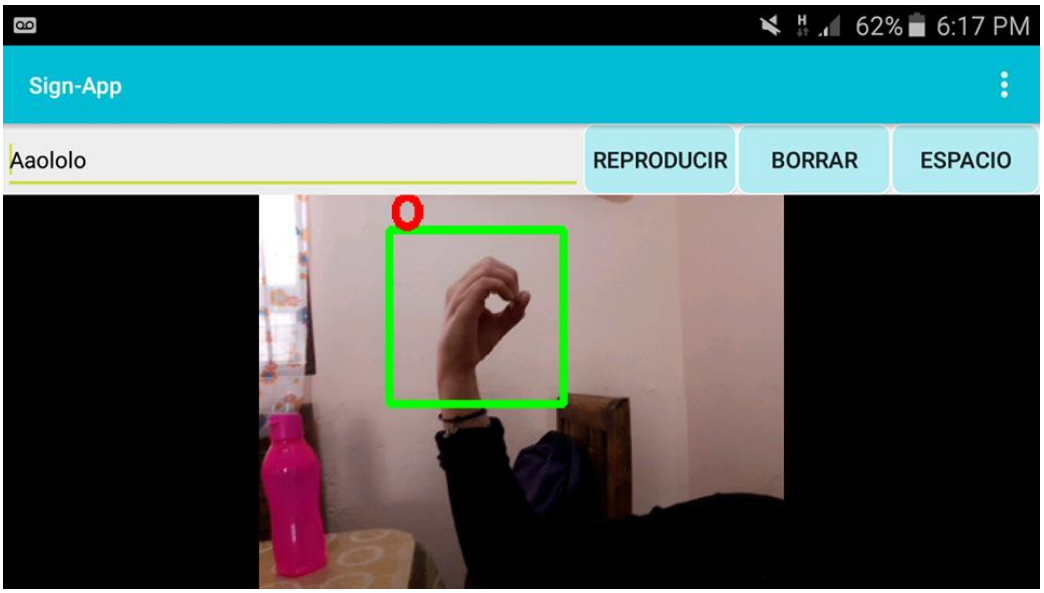


Figura 4. 21 Deteccion de letra O en fondo 2

Fondo 3: En este caso se utiliza un fondo blanco con poca luz (figura 4.22).

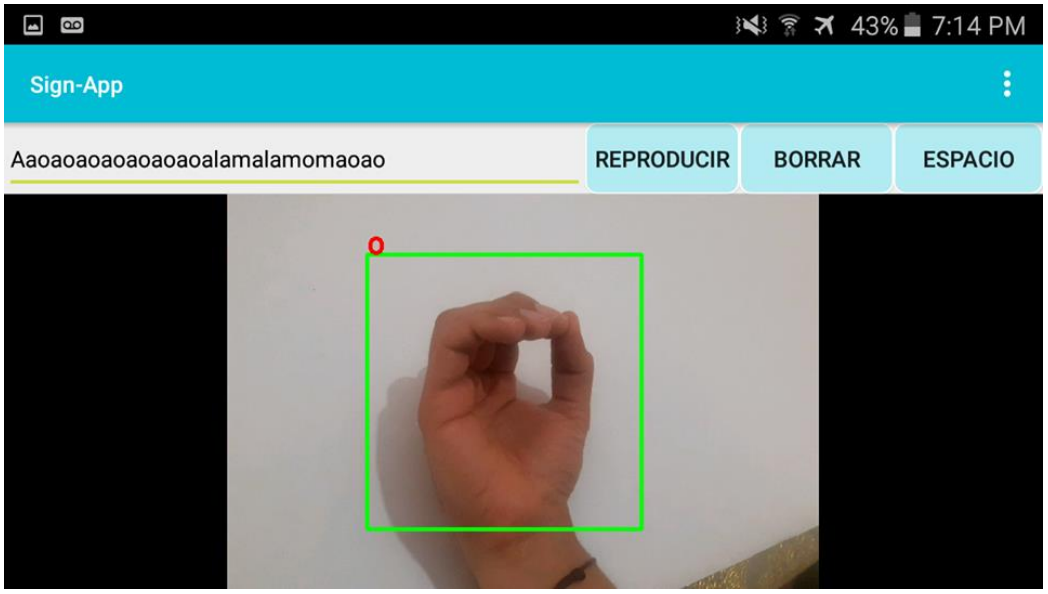


Figura 4. 22 Deteccion de letra O en fondo 3

Fondo 4: En esta caso se muestra la detección con un fondo azul claro (figura 4.23)

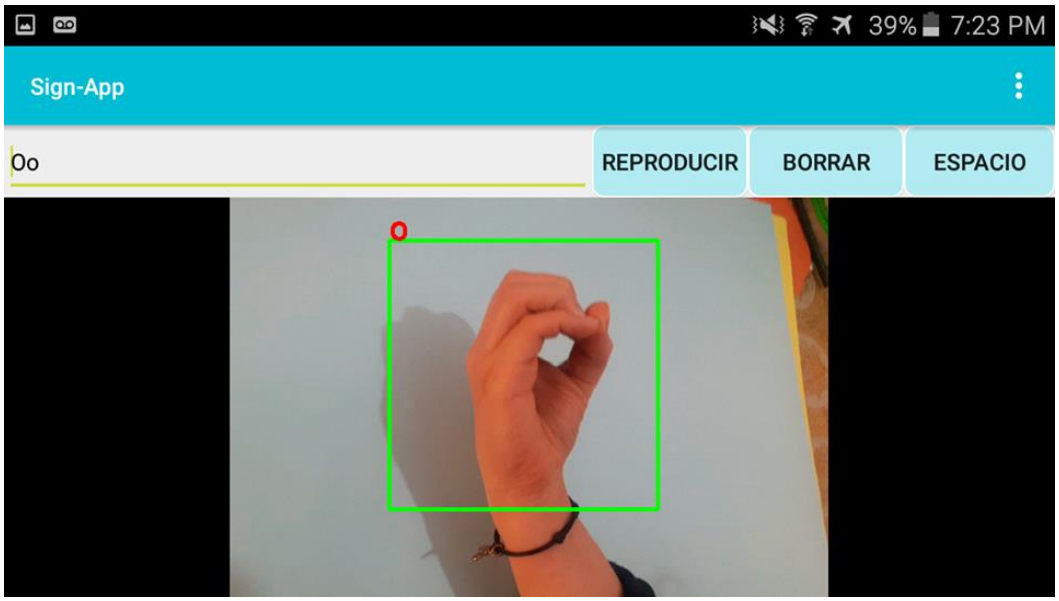


Figura 4. 23 Deteccion de letra O en fondo 4

Fondo 5: Para esta prueba se usa un fondo amarillo o beige (figura 4.24)

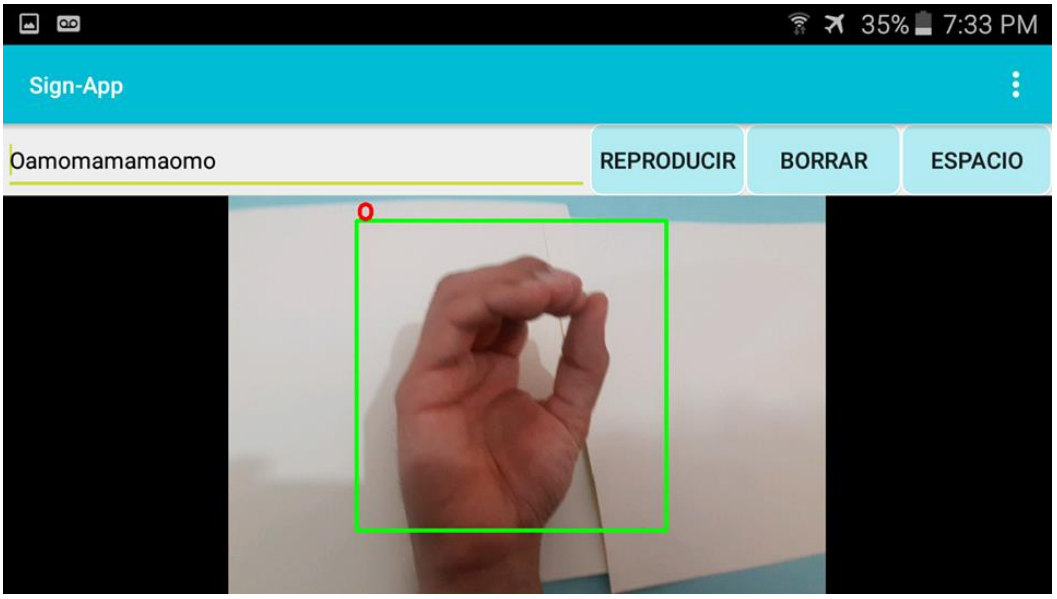


Figura 4. 24 Deteccion de letra O en fondo 5

Letra M

El clasificador de esta letra fue generado con 63 imágenes positivas y 1000 negativas. Los resultados de la prueba del clasificador con una lista de imágenes de la seña en una carpeta predeterminada se muestran en la siguiente figura:

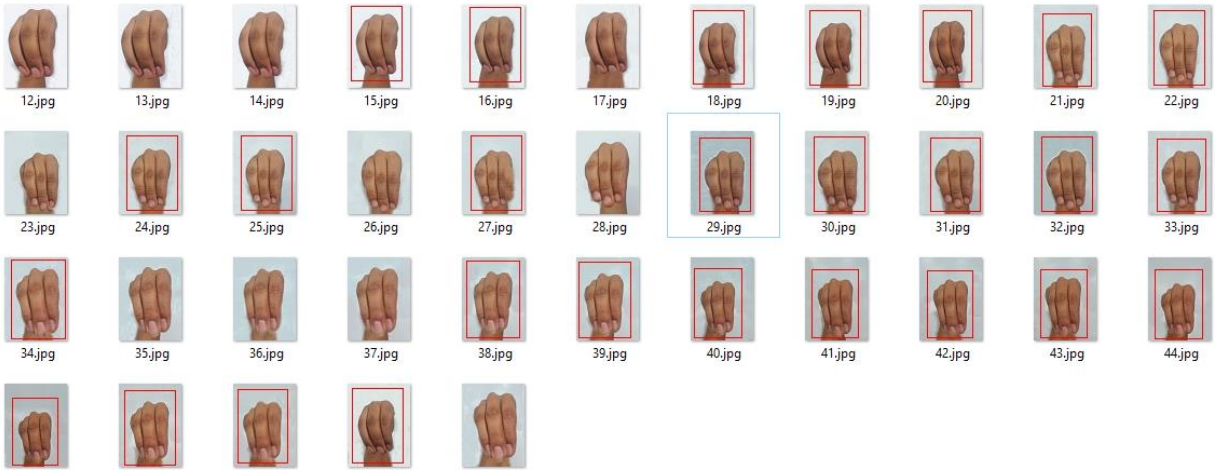


Figura 4. 25 Prueba en imagenes del clasificador de la letra M

Los resultados de las pruebas en la app se muestran en las imágenes siguientes.
Fondo 1: fondo blanco con relieve (figura 4.26).

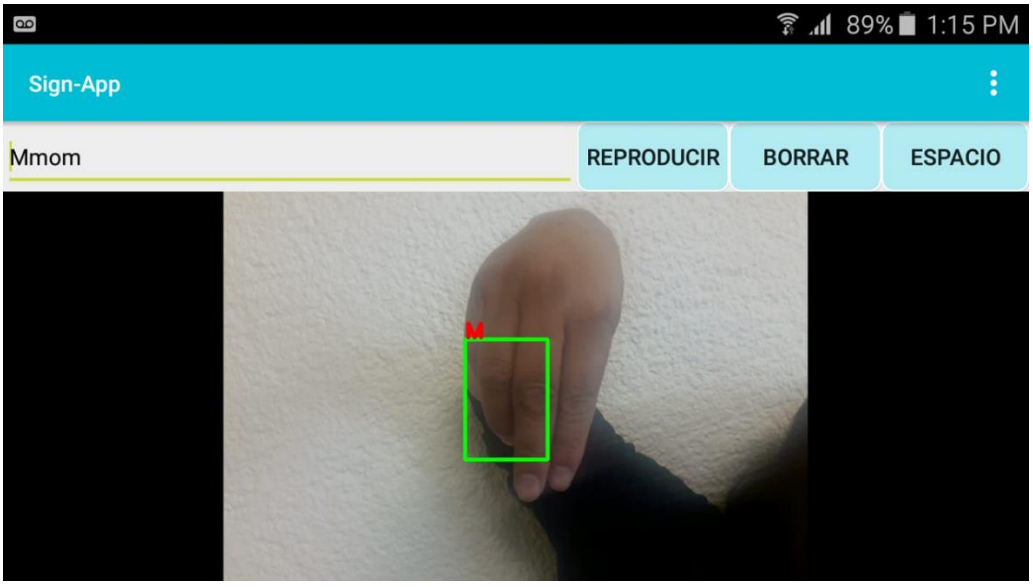


Figura 4. 26 Deteccion de letra M en fondo 1

Fondo 2: Para el caso en cuestión se usa el color blanco (figura 4.27).

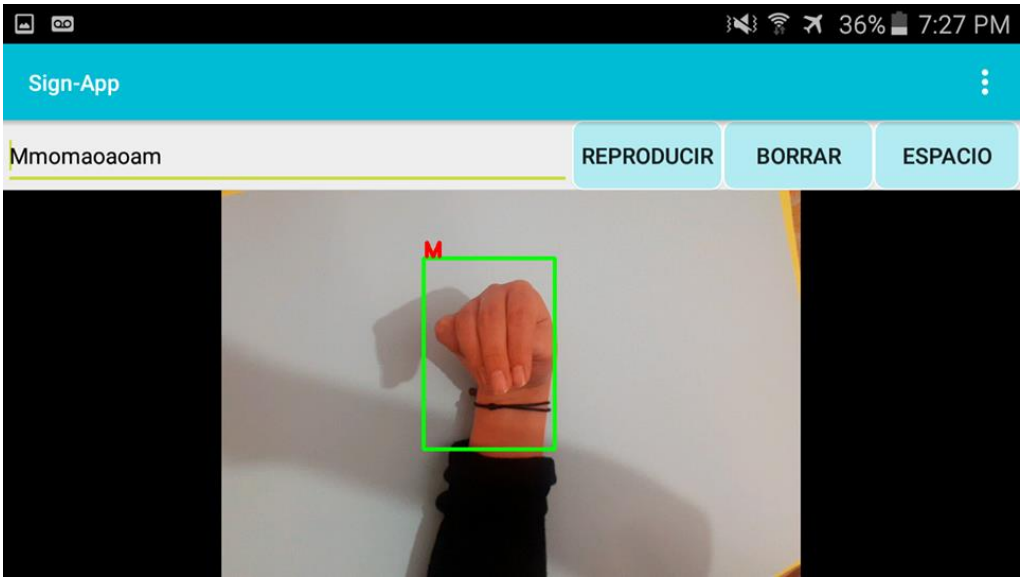


Figura 4. 27 Deteccion de letra M en fondo 2

Fondo 3: En este caso es utilizado un color amarillo o beige (figura 4.28).

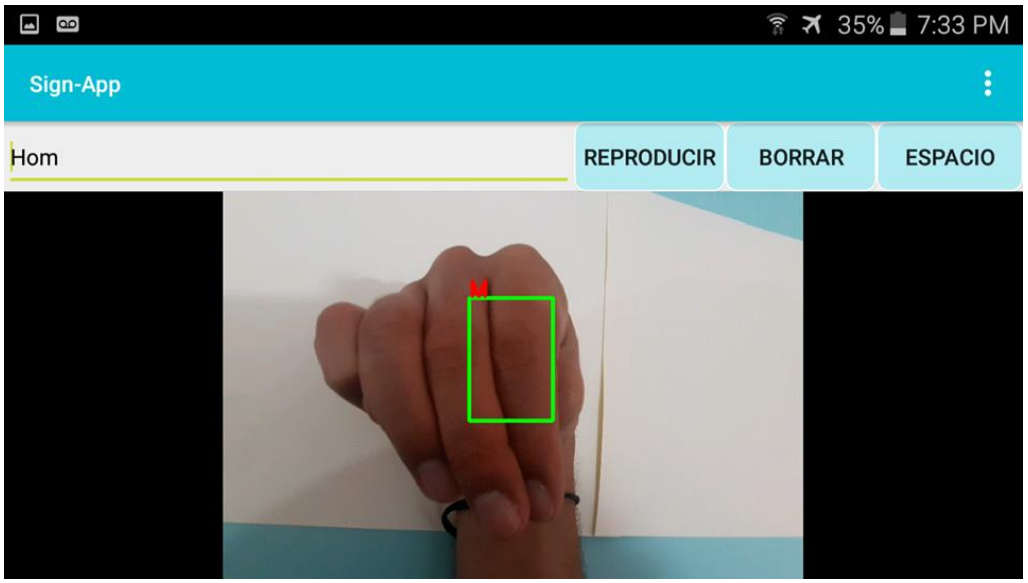


Figura 4. 28 Deteccion de letra M en fondo 3

Fondo 4: Aquí se usa un fondo azul, inclusive se observan objetos adicionales de fondo (figura 4.29).

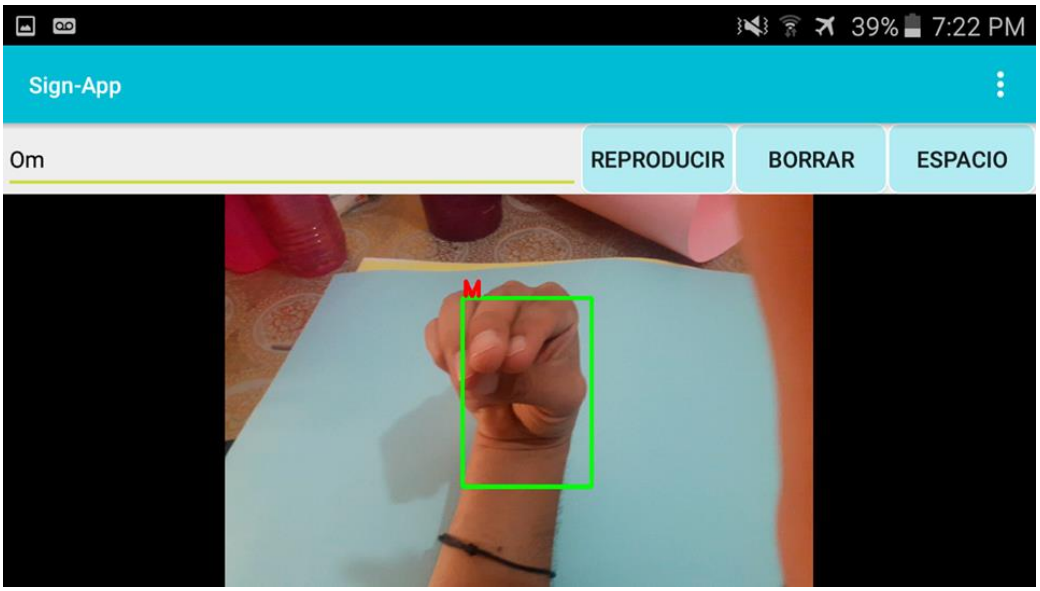


Figura 4. 29 Deteccion de letra M en fondo 4

Fondo 5: para este caso se utiliza un fondo blanco con poca luz (figura 4.30).

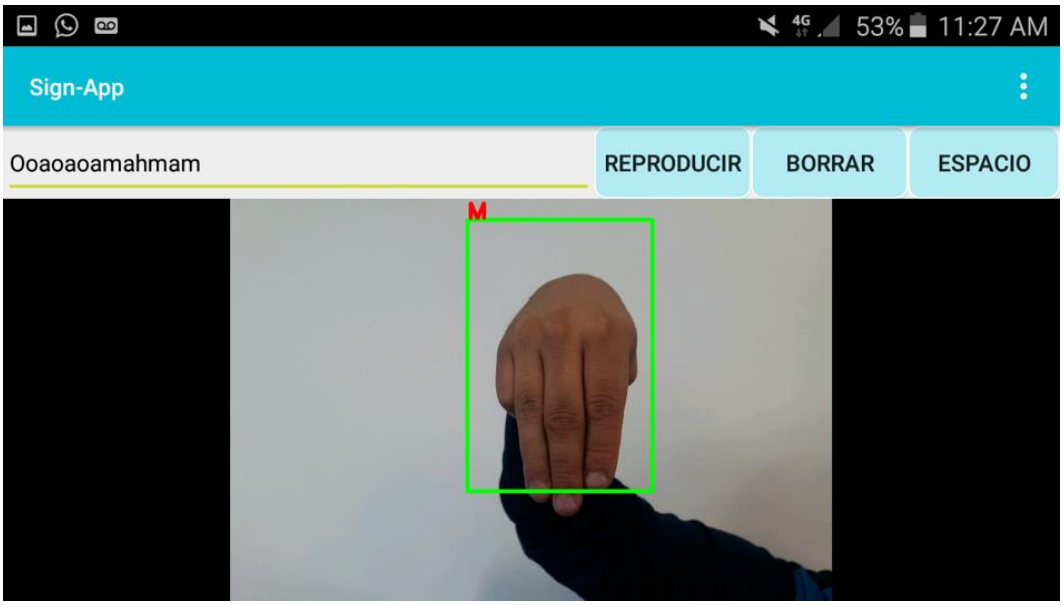


Figura 4. 30 Deteccion de letra M en fondo 5

Resumen de resultados

Se utilizó el IDE Android Studio 3.0.1 para el desarrollo de una aplicación móvil nativa para el SO Android, en la cual se implemento correctamente la librería OpenCV versión 2.4.9 en el proyecto y posteriormente se instaló correctamente en el dispositivo móvil. Especificamente se probo en dispositivos con las versiones del SO Android 5.1, 7.0 y 8.0.

Se desarrollaron los clasificadores Haar de las las letras del Alfabeto A, H, L, M y O por medio de la biblioteca OpenCV mismos que son cargados por la aplicación móvil para la detección en tiempo real de señas en imágenes captadas por la cámara del móvil. Se puede utilizar la cámara frontal y la trasera del dispositivo, además se puede cambiar la resolución de pantalla.

Al realizar la detección de la seña correspondiente se coloca la letra asociada en un espacio de la interfaz de la app. Asi mismo se señala con un cuadro en la pantalla con la letra detectada para mejorar la interacción con el usuario de la aplicación.

Se integró con exito en la interfaz de la aplicación un botón para la reproducción con voz artificial el texto formado por la concatenación de las letras de las señas detectadas. Este proceso de realizado por medio del Conversor Texto-Voz de Google integrado en los dispositivos con el SO Android.

Se comprobó la detección de las cinco señas a travás de la aplicación móvil en distintos fondos con el propósito de verificar el funcionamiento de la app y de los clasificadores Haar. La detección es correcta en cada uno de ellos. Se coloca el texto de la letra asociada correctamente, no obstante debido a la cantidad de fotogramas obtenidos constantemente durante el funcionamiento el proceso de colocar el texto de cada seña requiere algunas optimizaciones, ya que la entrada de dichas imágenes o fotogramas es continua.

Capítulo 5 CONCLUSIONES Y TRABAJO FUTURO

El alfabeto de la Lengua de Señas Mexicana (LSM), también denomindao dactilología, esta formado por la representación de las letras y números a través de señas con la mano derecha. La mayoría de dicho signos son estáticos. Esto significa que es posible detectar una seña en una imagen, que puede ser obtenida por medio de una cámara con la finalidad de realizar un comunicación con personas sordas.

Existen librerias de código abierto que permiten el procesamiento de imágenes con la finalidad de extraer datos de ella, por ejemplo se podría convertir a blanco y negro, resaltar los bordes, detectar ciertos colores, entre otros. Adicionalmente estas biliotecas permiten la posibilidad de realizar procesos sobre la imagen para detectar ciertos elementos u objetos en ella.

La librería OpenCV es una de las herramientas de código abierto mas utilizadas actualmente para el procesamiento de imagen, ya que tiene muchos módulos, que permiten aplicar filtros, histogramas, editar imágenes, cambios de color, etc. También integra las funciones y algoritmos para la detección y búsqueda de patrones en cualquier imágen. Adicionalmente permite su utilización en varios sistemas operativos y lenguajes de programación.

Los clasificadores Haar son una de la herramientas que integra la librería OpenCV como parte del módulo *ObjDetec*; permite tanto la construcción de dichos clasificadores como la detección de objetos en imágenes por medio de ellos. Estas imágenes pueden ser obtenidas desde un video o individualmente.

En este proyecto se obtuvieron como muestra cientos de fotos de cinco señas para generar sus clasificadores Haar e integrarlos a una aplicación móvil con la finalidad de realizar la detección de dichas señas a través de la librería OpenCV en el móvil. Cabe señalar que ésta app utiliza estos clasificadores para detectar algunas de las señas

definidas en la LSM en tiempo real y coloca el texto (letra del alfabeto) correspondiente a la seña en un espacio de la interfaz, por lo tanto realiza un atraducción de la LSM a texto. Adicionalmente se permite la reproducción con voz artificial del texto traducido.

La utilización de clasificadores Haar (Haar Cascade Classifier) para generar un patrón de búsqueda en imágenes de entrada por la cámara del móvil, es un método útil para detección de señas. En este proyecto de considero un fondo claro con la finalidad de hacer más eficiente el proceso.

En este proyecto se utilizo la plataforma Android ya que es el Sistema Operativo mas utilizado y de código abierto. No obstante los clasificadores son utilizables en cualquier sistema operativo y lenguaje que este soportado por OpenCV, por lo tanto resuta totalmente factible su implementación en otras plataformas y lenguajes de programación.

El porcentaje de detección exitosa depende de la cantidad de las muestras postivas y negativas. En esta aplicación se utilizaron pocas imágenes debido a que el proceso de generación de un clasificador Haar es proporcional a la cantidad de dichas imágenes de entrada, lo que lo hace muy lento, además se crearon con la finalidad de probar el funcionamiento de los clasificadores generados en una plataforma móvil.

Para incrementar y mejorar el proyecto es necesaria la obtención de miles de muestras de cada seña para estar en condiciones de generar un clasificador que detecte la seña con cualquier fondo, no obstante el tiempo de creación de estos clasificadores se incrementa considerablemente por cada seña.

Una opción adicional para mejorar la aplicación es la generación de los denominados 3D-Like features, es decir, clasificadores que consideran la profundidad en las imágenes. Sin embargo su tiempo de creación y detección es mucho mayor ya que el dispositivo móvil debe procesar en tiempo real mas clasificadores.

BIBLIOGRAFÍA

[1] A. Koon & de la Vega. (2000, Aug.) "El impacto tecnológico en las personas con discapacidad". Red CDPD [On-line]. 1, pp. 1-19. Available: http://www.repositoriocdpd.net:8080/bitstream/handle/123456789/363/Pon_KoonRA_ImpactoTecnologicoPersonas_2000.pdf?sequence=1 [Dec. 1 2017]

[2] CONADIS (2016, Aug.). " Lengua de Señas Mexicana (LSM)". CNADIS [On-line] 1, pp. 1-2. Available: https://www.gob.mx/conadis/articulos/lengua-de-senas-mexicana-lsm [Sep 27 2017]

[3] A. Valenzuela, E. Aguero, & Beguerí. (2011,Oct.). (2000, Aug.) "Desarrollo de entornos interactivos para usuarios sordos". CACIC 2011 [On-line]. 1, pp. 1-10. Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/18787/Documento_completo.pdf?sequence=1 [Oct. 1 2017].

[4] Acevedo, Flores, Lima, & Alducin. (2009, Jul.). "Lenguaje de señas por celular". CISCI 2009 [On-line]. 1, pp. 1-2. Available: http://www.iiis.org/cds2009/cd2009csc/cisci2009/paperspdf/c828ug.pdf [Sep. 23 2017]

[5] A. Medhi., M. Doshi, P. Pawar, A. Kesarkar & R. Dalvi. (2017, Feb.). "Real-Time Vision Based Sign Language Recognition System". IJIRCCE [On-line] 1, pp. 1-6. Available: https://www.ijircce.com/upload/2017/february/211_SIGN%20PAPER_IEEE.pdf [Sep. 8 2017]

[6] E. Chiguano, N. Moreno & L. Corrales (2011). "Diseño e implementación de un sistema traductor de lenguaje de señas de manos a un lenguaje de texto mediante visión artificial en un ambiente controlado". Escuela Politecnica Nacional [On-line]. 1, pp. 1-7. Available: http://bibdigital.epn.edu.ec/bitstream/15000/4924/1/PAPER_CHIGUANO_MORENO.pdf [Aug. 15 2017]

[7] J.L Raheja., A. Singhal & A. Chaudhary (2015). "Android based Portable Hand Sign Recognition System". Dept. of Computer Science Truman State University . [On-line]. 1, pp. 1-19. Available: https://arxiv.org/ftp/arxiv/papers/1503/1503.03614.pdf [Aug. 15 2017]

[8] Vintimilla Sarmiento, M. G. (2014, Jun). "Desarrollo e implementación de una aplicación que traduzca el abecedario y los números del uno al diez del lenguaje de señas a texto para ayuda de discapacitados auditivos mediante dispositivos móviles Android". Universidad de las Fuerzas Aramdas. [On-line]. 1, pp. 1-19. Available: https://repositorio.espe.edu.ec/bitstream/21000/8873/1/T-ESPE-048054.pdf [Aug. 20 2017]

[9] M. Kolsch, M.Turk, Robust hand detection, Proc. 6th Int. Conf. on Automatic Face and Gesture Recognition, (2004) pp. 614 – 619, Seoul, Korea.

[10] Y. Fang, K. Wang, J. Cheng, H. Lu, A Real-Time Hand Gesture Recognition Method, Proc. 2007 IEEE Int. Conf. on Multimedia and Expo, (2007) pp. 995-998

[11] Tomas Girones, J. (2017). El gran libro de android. Ed. Alfaomega, Marcombo. 6ª Edición. España.

[12] Instituto Nacional de Estadística y Geografía. “La discapacidad en México, datos al 2014” Internet: http://conadis.gob.mx/gob.mx/transparencia/transparencia_focalizada/La_Discapacidad_en_Mexico_datos_2014.pdf, [Jan. 05 2018].

[13] B’FAR R. “Mobile Computing Principles” 1st ed., Cambridge, 2005.

[14] Manilal Patel M., KaushikP V. “Mobile Computing.” International Journal of Innovative Science, Engineering & Technology, Vol. 1, pp.296-299, Sep. 2014.

[15] Kjeldskov,J. (2013). ”Mobile Computing”. The Encyclopedia of Human-Computer Interaction [On-line] 1, pp. 1-17. Available: http://people.cs.aau.dk/~jesper/pdf/books/Kjeldskov-B5.pdf [Sep 29 2017].

[16] R. Zambrano. (2014, Aug) Desarrollo de aplicaciones móviles: ¿nativas, multiplataforma, HTML5 o híbridas?. [Onine]. Available: http://www.mobidoo.es/desarrollo-movil/desarrollo-de-aplicaciones-moviles-nativas-multiplataforma-html5-hibridas/

[17] J. Tomas, Jirones. (2017, May 14). Máster en Desarrollo de Aplicaciones Android (2nd Ed). [Online]. Available: http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo.

[18] Serafín de Fleischmann & González Pérez (2011). “Manos con Voz”. CONAPRED [On-line] 1, pp. 1-17. Available: http://www.conapred.org.mx/documentos_cedoc/DiccioSenas_ManosVoz_ACCSS.pdf [Nov. 3 2017].

[19] A, Dorta. (2018, Feb 15). Mexicanos crean app que traduce texto y voz en lenguaje de señas. [Online]. Available: http://www.mexiconewsnetwork.com/es/noticias/mexicanos-crean-app-traduce-texto-voz-lenguaje-selas/

[20] Buap.mx (2016, Oct 16). Escuchame. Una app que Facilita la inclusion social. [Online]. Available: https://www.buap.mx/content/esc%C3%BAchame-una-app-que-facilita-la-inclusi%C3%B3n-social.

[21] Sucar, L. E. & Gomez G. (2011). “Visión computacional”. INAOE [Online] 1, pp. 1-18. Available: https://ccc.inaoep.mx/~esucar/Libros/vision-sucar-gomez.pdf [Nov. 8 2017]

[22] Medhi A, Doshi, M., Pawar P, Kesarkar A. & Dalvi R . (2017, Feb.). “Real-Time Vision Based Sign Language Recognition System”. IJIRCCE [Online] 1, pp. 1-6. Available: https://www.ijircce.com/upload/2017/february/211_SIGN%20PAPER_IEEE.pdf [Sep. 8 2017]

[23] W. Garaje & G. Brasdki. (2017, Oct 1). OpenCV. [Online]. Available: https://opencv.org/

[24] P. Viola & M. J. Jones. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. [Online] Available:http://research.microsoft.com/en-us/um/people/viola/Pubs/Detect/violaJones_CVPR2001.pdf

[25] W. Garaje & G. Brasdki. (2017, Nov 03 16.). Cascade Classfication. [On-line].Available: https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html?

[26] H. Duxans I Barrobés, “Voice Conversión Applied to Text-To-Speech Systemas”, Ph.D. Thesis, Universidad Poltecnica de Cataluña. Barcelona, España, 2006

[27] H. Duxans Barrobés y M. Ruiz Costa-jussá, «Síntesis del habla,» de Procesamiento de audio, Barcelona, Eureca Media, 2012.

[28] A. Alondo. I. Sainz. D. Erro. E. Navas and and I. Hernaez (2013, Sep). “Sistema de Conversión Texto a Voz de Código Abierto Para Lenguas Ibéricas”. Sociedad Española Para el Procesamiento del Lenguaje Natural. [Online]. 1, pp. 1-7. Available: https://pdfs.semanticscholar.org/d68d/22a10598ebe83c94c32e436d90f8e7685569.pdf

[29] A. Raya. (2013, Jun, 13). La app de síntesis de voz de Android llega a Google Play. [Online]. Available: https://elandroidelibre.elespanol.com/2013/11/la-app-de-sintesis-de-voz-de-android-llega-a-google-play.html

[30] www.tutorialspoint.com. (2018). Android Text To Speech. [online] Available at: https://www.tutorialspoint.com/android/android_text_to_speech.htm [Accessed 30 Jun. 2018]

[31] Q. Chen, N.D. Georganas, E.M. Petriu, Real-time Vision-based Hand Gesture Recognition Using Haar-like Features, Proc. Instrumentation and Measurement Technology Conf. – IMTC 2007, (2007)Warsaw, Poland

[32] P. Hoshi, D. Millán Escrivá and V. Godoy. *OpenCV by Example*. Packt Publishing Birmingham. UK. 2016

[33] L. Joseph. (2011, August, 07). How to do OpenCV Haar Training. [Online]. Available: http://www.technolabsz.com/2011/08/how-to-do-opencv-haar-training.html

[34] T. Ball. (2013, July, 23). Train your own OpenCV Haar Classifier. [Online]. Available: https://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html

[35] A. Ahmadl. (2017, July, 05) Cascade Trainer GUI. [Online]. Available: http://amin-ahmadi.com/cascade-trainer-gui/

[36] M. Guerrero. (2015, November, 11). Metodología Mobile-D: Para desarrollos de aplicaciones móviles. [Online]. Available: http://manuelguerrero.blogspot.es/1446543763/metodologia-mobile-d-para-desarrollos-de-aplicaciones-moviles/

ÍNDICE DE FIGURAS

GLOSARIO DE TÉRMINOS

Alfabeto Dactilológico: Es el alfabeto que usan las personas sordas. Consiste en la representación manual de las letras y números que componen el alfabeto de la lengua oral.

Android: Es una pila software para dispositivos móviles que incluye un Sistema Operativo, un conjunto de aplicaciones clave y una interfaz de comunicación entre ellos. Tradicionalmente se asocia al sistema operativo para dispositivos móviles respaldado por la denominada Open Handset Alliance.

Android Studio: Es el Entorno de Dearrollo Integrado (IDE) oficial para el desarrollo de aplicaciones para el Sistema Operativo Android.

API (Application Programming Interface): La Interfaz de programación de aplicaciones es conjunto de procedimientos, métodos y funciones, que ofrece cierta biblioteca para ser utilizado por otro software.

Aplicación móvil (app): Es un programa o software que tiene la característica de estar dirigido y desarrollado específicamente para ser utilizado en dispositivos móviles.

Apk (Android Application Package): Extensión de archivo. Es una variante del formato JAR de Java que permite distribuir e instalar aplicaciones y componentes de software para el Sistema Operativo Android.

Clasificador en cascada: Es un detector de objetos que hace uso del algoritmo Viola-Jones para la detección de características Haar-like en una imagen, es decir, un clasificador en cascada básicamente permite buscar las caracteristicas de un objeto en las imágenes que se le den como entrada. Se denomina en cascada porque realiza la búsqueda por secciones o áreas, descartando aquellas menos probables.

Clasificadores Haar: Es un detector de objetos que se basa en la búsqueda de las características Haar (Haar features) en una imagen digital.

Clase (Lenguaje Java): Es la unidad básica del lenguaje de programación Java. Agrupa las instrucciones y permite la creación de programas y aplicaciones para la plataforma Android.

Componente Principal de Analisis (PCA): Es una técnica que describe un conjunto de datos en términos de nuevas variables no correlacionadas.

Computadora: Dispositivo electrónico que ejecuta millones de operaciones por segundo; requiere de un software para su funcionamiento (instrucciones lógicas detalladas de las actividades a realizar).

Dactilología: Es un sistema de comunicación que permite el intercambio de mensajes o información por medio del uso principalmente de los dedos y el movimiento de las manos.

Dispositivo móvil: Aparato electronico de tamaño pequeño que cuenta con capacidad de procesamiento, conexión a una red, memoria limitada y permite la ejecución de funciones similares a las de una computadora. Esta asociado a un uso individual o personal.

Firmware: Conjunto de instrucciones (de programación), registradas en una memoria ROM o similar que controla los circuitos electrónicos de algún dispositivo.

Haar features: Son características de una imagen digital utilizadas en el reconocimiento de objetos.

Hardware: Se refiere a los componentes físicos de un equipo de cómputo o de un dispositivo.

HTML (Hypertext Markup Language): Lenguaje de marcas de hipertexto. Es el lenguaje para el diseño de páginas web.

IDE (Integrated Development Environment): Entorno de Desarrollo Integrado. Es un programa o aplicación compuesta por un conjunto de herramientas útiles para un programador.

Ideograma: En la lengua de señas los ideogramas se refieren a la representación de una palabra o frase con una o varias configuraciones o movimientos de manos.

Imagen: Es una representación que manifiesta la apariencia de una objeto real o una parte pequeña del entorno.

Imagen (en computación): Es la representación bidimensional de una imagen a partir de una matriz numérica. Frecuentemente en binario (ceros y unos). Generalemnte es tratada, generada y manipulada por dispositivos electrónicos.

Internet: Es la red de computadoras mundial interconectadas a través de medios y dispositivos de red que permiten el intercambio electrónico de datos y que utilizan la familia de protocolos TCP/IP.

Java: Lenguaje de programación Orientado a Objetos de alto nivel cuyo propietario es Oracle.

JDK (Java Kit Development): Kit de Desarrollo de Java. Es un software que provee herramientas de desarrollo para la creación de programas en el lenguaje de programación Java.

Lengua de señas: Es una lengua que se basa en la configuración de gestos y movimientos realizados con las manos, adicionalmente también se incluyen gestos con el rostro. Es utilizada por las personas sordas.

Lengua de señas Mexicana: Es la lengua de señas que utilizan las personas sordas en México.

Lenguaje de programación: Es un lenguaje artificial y formal que permite la creación de tareas o actividades que serán ejecutadas por las computadoras o dispositivos similares a través de instrucciones especificas con una sintaxis y una semántica.

Lenguaje natural: El lenguaje natural es el lenguaje hablado o escrito por humanos para propósitos generales de comunicación.

Material Design: Es una guía integral para el diseño visual, de movimientos y de interacción en distintas plataformas y dispositivos, es decir, es una normativa de diseño de interfaces gráficas enfocado en la visualización e interacción fácil e intuitiva con el usuario.

Maquina de Soporte Vectorial (SVM): Un conjunto de algoritmos de aprendizaje supervisado, es decir, algoritmos para extracción o deducción de una función a partir de datos de entrenamiento.

Maquina virtual (en lenguaje de programación Java): Es un entorno de ejecución que permite ejecutar instrucciones del lenguaje Java sin compilar todo el programa, es decir, realiza la traducción de codigos de bytes (bytecode) a código binario en tiempo real.

OpenCV(Open Source Computer Vision Library): Es una biblioteca abierta de software de visión computacional y de aprendizaje automático.

Paquete (en Lenguaje Java): Es un elemento que agrupa un conjunto de clases con funcionlaidad lógica predefinida y relacionada, es decir, permite agrupar instrucciones de programación relacionadas y listas para utilizarse en algún programa que lo incluya.

SDK (Software Development Kit): Conjunto de herramientas para el desarrollo de software que le permite crear aplicaciones para un sistema concreto.

Sistema operativo: Software que permite la utilización de la computadora y proporciona la forma o interfaz para comunicar el dispositivo con la persona.

Software: Conjunto de programas que permiten el funcionamiento de una computadora o dispositivo móvil actual (SmartPhone).

TTS (Text-To-Speak, en español Conversor de Texto a Voz o CTV): Es la creación por medios automáticos de una voz artificial que genera el sonido similar al producido por una persona al leer un texto cualquiera en voz alta.

Voz artificial: proceso de creación de habla de manera artificial (por medios mecánicos, electrónicos, etc.).

Wrapper (computación): Es un componente de software que facilita la utilizacion de otros programas complejos a traves de instrucciones mas fáciles y accesibles, facilita el desarrollo y reutilización de software.

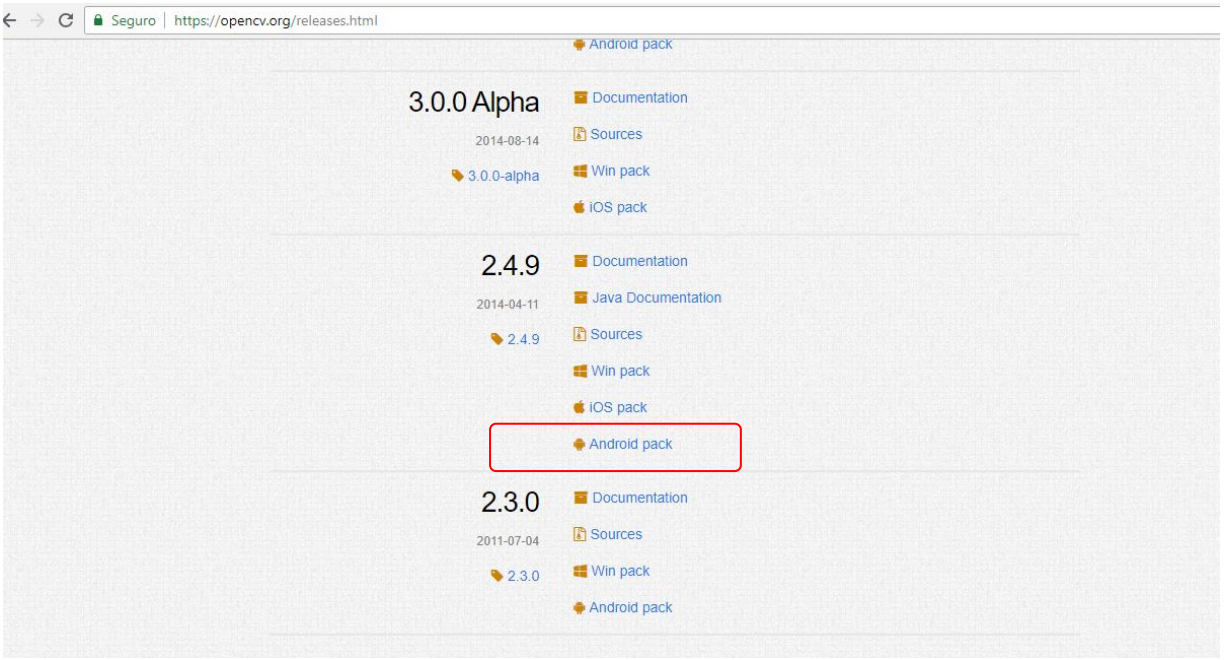
XML (Extensible Markup Language): Es un metalenguaje que permite definir lenguajes de marcado para usos específicos.

Wearable (computación): Hace referencia a los aparatos y dispositivos electrónicos que se incorporan en alguna parte del cuerpo interactuando de forma continua con el usuario y otros dispositivos.

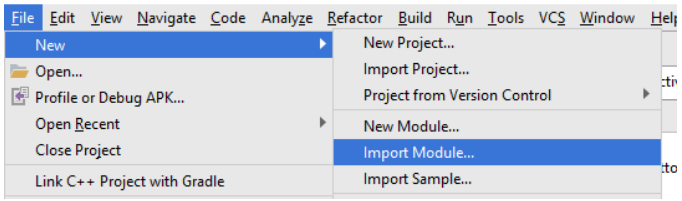
WWW (World wide Web): Hace referencia a un sistema que permite visualizar la información de internet a través de varias tecnologías, principalmente un programa especial llamado navegador o explorador de internet y las denominadas páginas web, diseñadas con el lenguaje HTML.

Apéndice 1 Integración de OpenCV a Android Studio

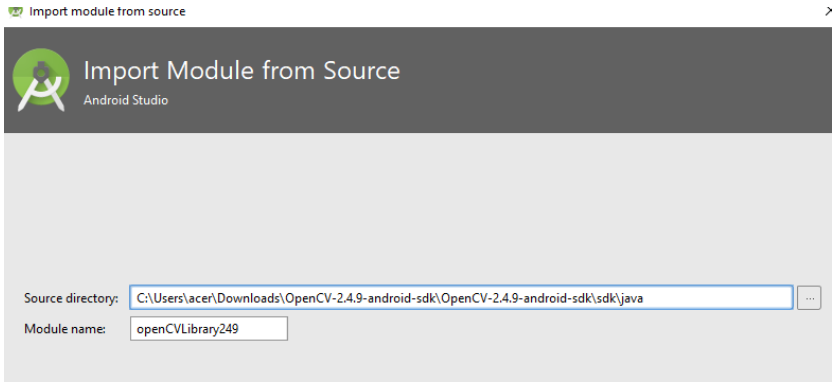
Descarga de OpenCV para Android. Versión 2.49



Una vez descargada la librería procedemos a integrarlo al proyecto de Android Studio. Para este proceso es necesario agregar un modulo, mismo que será importado y asociado a los recursos de la biblioteca mencionda, en la figura siguiente se muestra el menú para iniciar la importación.

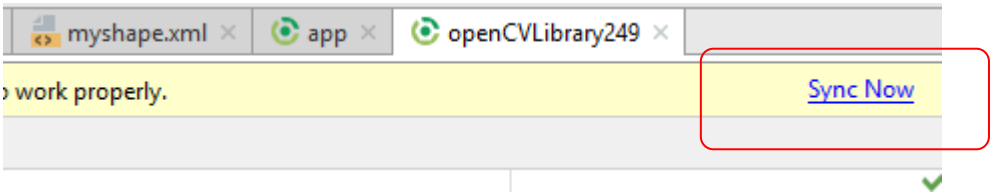


Despues es necesario especificar la ruta en la cual están los recursos de openCV en su versión para Android (Java), tal como se muestra a continuación.

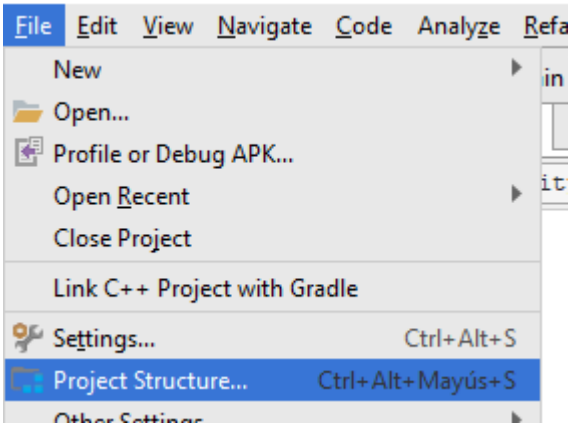


Luego actualizamos l configuración de Gradle del módulo OpenCV, es porque es necesario que coincidan con la configuración de la app.

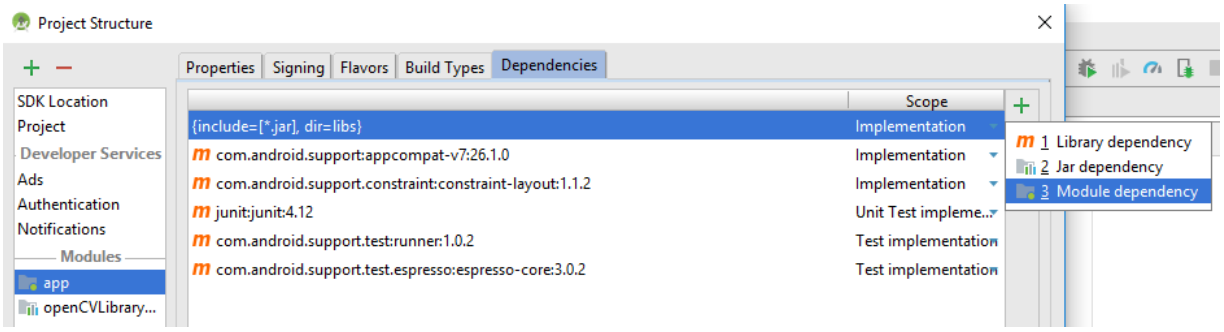
Posteriormente haremos clic en “Sync Now” para reconstruir el proyecto.



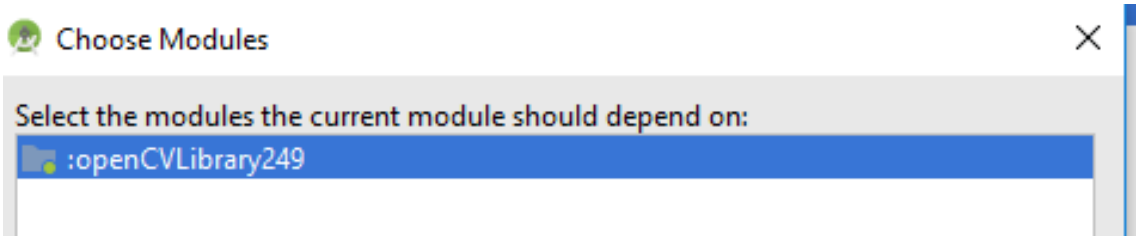
Despues en el seleccionamos la opción “Project Structure…” del menú File.



En el cuadro que nos aparece seleccionamos el modulo “app”, después la pestaña “Dependencies”, luego hacemos clic en el icono “+” y seleccionamos “Module Dependency”.



En el cuadro de dialogo siguiente seleccionamos el módulo “openCVLibrary249” y hacemoc clic en “OX”.



Finalmente hacemos clic en OK nuevamente y esperamos que Gradle sincronice el proyecto.