DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGAIÓN

“Modelo para la creación y uso de un repositorio digital de segmentos visuales en lengua de señas mexicanas utilizando Inteligencias Artificial”

T E S I S

PRESENTADA COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO DE:

MAESTRA EN CIENCIAS DE LA COMPUTACIÓN

Maria Luisa Galaz Palma

Director:

MC. César Enrique Rose Gómez

Hermosillo Sonora, México 22 de junio del 2022

Hermosillo, Sonora a 22 de junio DE 2022
SECCIÓN: DIV. EST. POS. E INV.
No. OFICIO: DEPI/134/22.
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS.

C. MARÍA LUISA GALAZ PALMA
P R E S E N T E

Por este conducto, y en virtud de haber concluido la revisión del trabajo de tesis que lleva por nombre **"MODELO PARA LA CREACIÓN Y USO DE UN REPOSITORIO DIGITAL DE SEGMENTOS VISUALES EN LENGUA DE SEÑAS MEXICANAS UTILIZANDO INTELIGENCIA ARTIFICIAL"**, quien fue dirigida por el M.C. César Enrique Rose Gómez, que presenta para el examen de grado de la MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN, y habiéndola encontrado satisfactoria, nos permitimos comunicarle que se autoriza la impresión del mismo a efecto de que proceda el trámite de obtención de grado.

Deseándole éxito en su vida profesional, quedo de usted.

A T E N T A M E N T E

M.C. CÉSAR ENRIQUE ROSE GÓMEZ
DIRECTOR DE TESIS

DRA. MARÍA TRINIDAD SERNA ENCINAS
SECRETARIA

M.C. RAFAEL ARMANDO GALAZ BUSTAMANTE
VOCAL

DR. GERMÁN ALONSO RUÍZ DOMÍNGUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

GARD/momv*

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE HERMOSILLO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

CARTA CESIÓN DE DERECHOS

En la ciudad de Hermosillo Sonora a el día 22 de junio del año 2022 la que suscribe C. Maria Luisa Galaz Palma, alumna de la maestría en Ciencias de la computación adscrito a la División de Estudios de Posgrado e Investigación, manifiesta que es autora intelectual del presente trabajo de Tesis titulado "Modelo para la creación y uso de un repositorio digital de segmentos visuales en lengua de señas mexicanas utilizando Inteligencias Artificial" bajo la dirección de MC. César Enrique Rose Gómez y ceden los derechos del mismo al Tecnologico Nacional de México/Instituto Tecnologico de Hermosillo, para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben de reproducir el contenido textual, graficas, tablas o datos contenidos sin el permiso expreso del autor y del director del trabajo. Este puede ser obtenido a la dirección de correo electrónico siguiente: m15331171@hermosillo.tecnm.mx . Una vez otorgado el permiso se deberá expresar el agradecimiento correspondiente y citar la fuente del mismo.

ATENTAMENTE

Maria Luisa Galaz Palma

Agradecimiento

Quiero agradecer a todas las personas que hicieron posible esta tesis y que de alguna manera estuvieron conmigo en los momentos difíciles y alegres. Principalmente a mis padres por todo su cariño, comprensión y apoyo, pero sobre todo por la paciencia que me han tenido.

A mis hermanos, por todos los consejos brindados y llenarme de alegría día tras día y por aguantarme en los momentos más estresantes.

A mis amigos, a darme los ánimos de seguir adelante y ayudarme en mis momentos más difíciles.

No puedo dejar de agradecer a mis maestros que me ayudaron en la realización de esta tesis, quienes me brindaron conocimientos y siempre me estuvieron dando de su apoyo cuando lo necesitaba.

Quiero brindar un especial agradecimiento a mi director de Tesis, el maestro Cesar Enrique Rose, el cual nunca podre dejar de agradecer, ya que, a pesar de haber sido una época difícil para él, siempre estuvo ahí a mi lado y brindándome todo el apoyo posible. Así también por enseñarme y por darme su cariño. Sin él, esta tesis no habría sido posible. Muchas gracias maestro.

Finalmente, quiero agradecer el apoyo del Consejo Nacional de Ciencias y Tecnología (CONACYT) por la beca número 004312.

Resumen

Una educación inclusiva en México ha desatado diversos métodos con el propósito de poder implementarla, y de esta forma, sin importar la capacidad de cada persona, pueda tener una educación digna. Lo anterior, a pesar de ser una tarea difícil, con la ayuda de la tecnología se han podido desarrollar diferentes propuestas para las diferentes capacidades. En el TecNM/Instituto Tecnológico de Hermosillo, se está desarrollando una plataforma para las personas con discapacidad auditiva, con el propósito de implementarla en la educación superior tecnológica. Esta plataforma tiene como objetivo ser el intérprete entre el maestro y el alumno, donde el maestro al dar la clase, la plataforma traducirá del español (en voz) a la lengua de señas mexicana (LSM, lengua utilizada por la comunidad Sorda de México).

En esta tesis tiene como objetivo realizar un modelo para la creación y uso de un repositorio digital de segmentos visuales en lengua de señas mexicanas utilizando la inteligencia artificial. Se aprovechan los videos públicos de la web, de los diferentes intérpretes de la LSM para poder crear el repositorio y transformar sus movimientos en modelo 3D de forma homogénea. Para lo anterior, se da uso de diferentes redes neuronales pre entrenadas.

Palabras claves:

Repositorio digital, Lengua de señas mexicanas, Redes neuronales, Inteligencia Artificial.

Contenido

Índice de Figuras

CAPÍTULO 1. INTRODUCCIÓN

Te imaginas vivir en una sociedad donde solamente 5 personas conocen tu idioma y te sea casi imposible comunicarte con los demás. Aún peor, que esas personas te ignoren porque sienten incomodidad al verte, ya que no creen posible establecer una conversación contigo. Esto es lo que están viviendo la gran parte de la comunidad sorda, viviendo una vida con frustración y discriminación. En México se tienen más de 2.5 millones de personas con discapacidad auditiva [1], y solo hay aproximadamente 40 intérpretes de la Lengua de Señas Mexicana certificadas en todo México [2].

La lengua de señas es la lengua natural para la comunidad sorda. A diferencia de la lengua oral, esta se compone de movimientos corporales y gestos faciales. En México se utiliza la Lengua de Señas Mexicana (LSM) para su comunidad sorda.

Al ser México un país donde predomina como idioma el español, pone el LSM como un idioma minoritario. Esto ha provocado el aislamiento de la comunidad sorda, dificultando su desarrollo personal, profesional y educativo. Por lo tanto, ha alarmado a la comunidad por la falta de cumplimiento de sus derechos, principalmente al derecho a tener una educación.

Por ende, en el TecNM/Instituto Tecnológico de Hermosillo, al notar esta problemática, aprovecharon las tecnologías que se tiene hoy en día, desarrollando una plataforma capaz de hacer interpretaciones del español (con reconocimiento de voz) al LSM en tiempo real [3]. Sin embargo, esta plataforma aún necesita mejoras para su correcto funcionamiento.

En este documento se presenta como tesis “Modelo para la creación y uso de un repositorio digital de segmentos visuales en lengua de señas mexicanas utilizando

inteligencia artificial”, con el fin de poder implementar una mejora a la plataforma anteriormente mencionada.

1.1. Antecedente

La lengua de señas es una lengua natural utilizada por la comunidad sorda. Se distingue de la lengua oral, ya que utiliza el canal de comunicación viso-gestual en lugar del oral-auditivo [4]. Como toda lengua, posee su propia sintaxis, gramática y léxico, y así mismo varía según cada región del mundo. En el caso de México, se utiliza la lengua de señas mexicanas (LSM) para su comunidad sorda. A pesar de que el LSM ha estado en México desde la antigüedad, es reconocido oficialmente como una lengua nacional y forma parte del patrimonio lingüístico desde el 10 de junio del 2005 [5].

En México predomina el español, poniendo el LSM como una lengua minoritaria, es decir, el LSM está en desventaja en un contexto cultural [6]. México cuenta con más de 2 millones de personas sordas [1]. Por desgracia, se tiene solamente 40 intérpretes de LSM certificados. Esta cifra se ha mantenido desde el 2009, a consecuencia de diferencias entre el Consejo para la Inclusión de las Personas con Discapacidad (CONADIS) y el área de la secretaria de la Educación Pública (SEP) quienes certifican a los profesionistas [2].

Por lo tanto, las personas con discapacidad auditiva presentan dificultad para comunicarse con los demás, dificultando su desarrollo profesional, humano y educativo, este último es en el que se ven mayormente afectados, ya que los discapacitados son clasificados en un mismo grupo, es decir, la misma institución, con mismos maestros y recursos para todo tipo de discapacidad. Por tanto, los maestros carecen de conocimiento de LSM, desatendiendo a los estudiantes sordos. Esto provoca complicaciones de comunicación para el estudiante con discapacidad auditiva. Gracias a la negligencia de los maestros, aprueban al

estudiante a pesar de su carencia de conocimiento, afectando su calidad educativa [2]. Lo anterior sucede hasta el nivel medio superior, pero en el nivel superior ni siquiera tienen la posibilidad de ingresar para cursar una carrera profesional.

Se han hecho investigaciones con el fin de implementar herramientas capaces de hacer una sociedad accesible, logrando que los sordos y oyentes logren tener una mejor calidad de vida. Para el desarrollo de estas herramientas se está aprovechando la inteligencia artificial. Un ejemplo de ellas es la aplicación *AVA.*

AVA se encuentra actualmente en el mercado y fue desarrollado por un equipo pequeño que se encuentra distribuido en dos ciudades: San francisco, California; y París, Francia. Esta aplicación utiliza tecnología de reconocimiento de voz basado en la inteligencia artificial (IA), generando subtítulos en tiempo real. Su desarrollo fue con el propósito de ayudar a más de 150,000 personas diarias con alguna discapacidad auditiva [7].

Existen más aplicaciones similares a AVA, pero estos tipos de aplicaciones hacen traducciones de voz a texto. Para entender el problema se debe considerar que existen tres tipos de sordos. El primero es el sordo hablante, son personas las cuales adquirieron el español como lengua materna y luego perdieron la audición. Después están los semilingües, estos no han adquirido competencias lingüísticas (señas u oral), ya que nacieron con problemas de audición impidiendo hablar en español, y por circunstancias sociales tampoco se les ha permitido obtener la lengua de señas. Y finalmente, los sordos señantes, son aquellos que pertenecen a una comunidad lingüística utilizando el LSM como su sistema primario de comunicación [6].

Por lo mencionado anteriormente, se puede entender que la realización de un traductor del español, con reconocimiento de voz en tiempo real, a la lengua de señas mexicanas es lo apropiado.

Por lo tanto, esto motivó desarrollar un proyecto en el TecNM/Instituto Tecnológico de Hermosillo para el diseño de una plataforma para la asistencia a personas con discapacidad auditiva, para facilitar su entorno de aprendizaje en un sistema de educación superior tecnológica.

1.2. Planteamiento del problema

El derecho a la educación es la principal demanda de la comunidad sorda, pues el 40% de la población sorda es analfabeta en México, según el censo del 2010 del Instituto Nacional de Estadística y Geografía (INEGI). Hay alrededor de 75 mil personas sordas que utilizan el LSM como única lengua y por falta de maestros del LSM, las personas sordas tienen que adaptarse con el sistema de educación, el cual no cumple con el derecho constitucional de una educación integral incluyente, bicultural y bilingüe [8].

Para la solución de esta problemática se desarrolló una plataforma en el TecNM/Instituto Tecnológico de Hermosillo, por el MCC Ernesto Darío Barraza Granillo [3]. Esta plataforma cuenta con la mayoría de las funcionalidades necesaria para una traducción en tiempo real del español mexicano al LSM, utilizando reconocimiento de voz, en un contexto de aprendizaje. Esto es porque cuenta con las siguientes funciones: traducción de voz a texto y traducción de texto a LSM. Asimismo, la plataforma es capaz de almacenar las sesiones de las clases, como también la posibilidad de realizar consultas en la web de palabras desconocidas. Ya que su finalidad es brindar una herramienta al estudiante con discapacidad auditiva, posibilitando su participación en sus estudios.

Sin embargo, la plataforma no se encuentra completamente funcional, ya que algunos módulos no cuentan con un desarrollo adecuado, provocando fallas en la traducción del español al LSM. Lo anterior se debe a que el LSM posee su propia sintaxis, gramática y léxico. Por lo tanto, el MCC Juan Carlos Hernández Cruz,

continuo la investigación [9], teniendo como objetivo realizar un algoritmo utilizando aprendizaje profundo, para la traducción del texto español mexicano a texto LSM. Dando como resultado, una herramienta útil para el apoyo de las traducciones realizadas en la plataforma previamente mencionada. Con esto se logró implementar un traductor capaz de obtener un texto en glosa de LSM a partir de cualquier texto en español que se le ingrese.

Una vez que se obtiene el texto en glosa de LSM, es necesario traducir esto a los movimientos corporales y, de igual forma, otro módulo que trabaje los gestos faciales, logrando de esta manera poder abarcar la estructura completa de la lengua de señas.

Lo anterior no es una tarea trivial, ya que se han creado tecnologías de forma universal, con aplicaciones capaces de ayudar a la comunidad sorda, pero hay que recordar que la lengua de señas es distinta en las diferentes zonas del mundo, inclusive a nivel nacional, es distinta en las diversas regiones del país. Así mismo, se carece de una tecnología visual para la traducción del idioma español al LSM a nivel educativo. Esta necesidad nos lleva a plantearnos las siguientes preguntas de investigación.

- ¿Cuáles son las fuentes heterogéneas en la Web de los videos o imágenes que contienen señas en LSM?
- ¿Cómo sería un repositorio de segmentos visuales en LSM?
- ¿Cuáles son los elementos de un modelo que permita realizar la creación de un repositorio digital de segmentos visuales en lengua de señas mexicana (LSM) usando inteligencia artificial?
- ¿Cuál es el mecanismo de recuperación de la secuencia de imágenes a ser visualizada a partir de la estructura de texto LSM?

De tal manera que estas preguntas nos formula una pregunta general:

- ¿Cuáles son los elementos de un modelo que permita, realizar tanto la creación de un repositorio digital de segmentos visuales en lengua de señas mexicana (LSM) usando inteligencia artificial, como la recuperación de la secuencia de imágenes para su visualización a partir de la estructura de texto LSM?

1.3. Objetivos

Para la realización del proyecto se plantean los siguientes objetivos:

1.3.1. Objetivo General

- Diseñar un modelo que incluya los elementos para recuperar palabras o sentencias en videos de LSM y transformarlos a modelos 3D en movimiento, para la creación de un repositorio de digital de segmentos visuales de LSM, para su uso en la visualización de texto LSM.

1.3.2. Objetivo Especifico

- Conocer las fuentes heterogéneas de videos LSM en la Web.
- Diseñar un algoritmo basado en IA para la recuperación de los videos LSM, así como el texto asociado.
- Diseñar un algoritmo para trasformar los videos LSM recuperados a modelos 3D en movimiento (avatar).
- Diseñar un repositorio para almacenar los segmentos visuales, así como su texto asociado.
- Diseñar un mecanismo de recuperación de la secuencia de segmentos visuales a partir de la estructura en texto del LSM.
- Generar un avatar el cual sea capaz de realizar las señas en LSM correspondientes.

- Diseñar un módulo para la visualización de la secuencia en LSM.

1.4. Justificación

Se ha implementado tecnologías en México para la educación, sin embargo, el nivel de aprendizaje permanece bajo [10]. Esto es porque a pesar de que se han implementado software, no se les ha dado la atención correcta a las distintas capacidades que tienen los estudiantes para aprender. Un ejemplo, es en la educación para un sordo. En México, se tienen aproximadamente 40 intérpretes de la lengua de señas mexicanas (LSM) certificados. Esto afecta en la educación del sordo ya que, no en todas las escuelas se cuenta con un intérprete certificado, por lo que los maestros tienen dificultades para comunicarse con el estudiante y el estudiante adquiere poca educación.

Por lo cual se necesita aprovechar las tecnologías, con el propósito de hacer una educación incluyente y de esta forma permitir con el derecho de tener una buena educación para cualquier ciudadano.

1.5. Alcances, delimitaciones y limitantes

1.5.1. Alcances

- El avatar por realizar debe de mantener la misma apariencia, sin importar el tipo de movimiento que realice respecto a las traducciones.
- Los movimientos corporales del avatar seguirán una secuencia mediante inteligencia artificial, utilizando los conocimientos obtenidos de la tesis [9].
- Se recuperará videos de la web, los cuales utilizan el LSM para basarse en los movimientos del avatar.

1.5.2. Delimitaciones

- Solamente se traducirá en lengua de señas mexicanas.
- Se basará en los movimientos corporales de inicio.
- Los movimientos se obtendrán basándose en los videos encontrados en la web.
- No se utilizarán palabras regionales según cada región de México.
- Sera para el sector de educación superior.

1.5.3. Limitaciones

- Puede que no se encuentren palabras en la web para la traducción.
- No todas las palabras se utilizan en todas las regiones de México

1.6. Metodología

Este proyecto se divide en 8 etapas.

La primera etapa se basa en la búsqueda y análisis de información relacionado a la investigación que se va a realizar. Los métodos para realizar esta búsqueda de información serán los siguientes:

- Uso de datos secundarios, con información de otros investigadores que hayan realizado investigaciones relacionadas a esta investigación.
- Con el uso de libros, con el fin de obtener los datos de conceptos no conocidos o para mejorar su conocimiento.
- Apoyándome con las tesis anteriores a está, ya que esta es la continuación de dos tesis.
- Por medio de artículos de revistas y congresos relacionados.

La segunda etapa, es la búsqueda en la web de videos del LSM el cual utilicen palabras que sean entendibles en toda la región de México.

La tercera etapa, es el análisis de tecnologías a utilizar para el desarrollo del proyecto. Para esto será necesario ver las diferentes propuestas que se han visto en las investigaciones obtenidas en la primera etapa y de esta forma tomar una decisión.

La cuarta etapa es el desarrollo de un modelo, para la realización del repositorio digital, el cual se utilizarán análisis para una correcta recuperación de datos.

La quinta etapa es la implementación del repositorio digital con segmentos visuales, para el desarrollo de un modelo 3D (Avatar).

La sexta etapa se realizarán pruebas y correcciones para el correcto funcionamiento del repositorio digital. Para esto será necesario presentar el proyecto a usuarios que utilizaran esta implementación en la plataforma y también será necesario la ayuda de intérpretes certificados para la validación del funcionamiento.

La última etapa se redactará todos los resultados obtenidos de la investigación, así mismo las sugerencias de mejoras para próximos proyectos.

CAPÍTULO 2. MARCO TEÓRICO

2.1. Lengua de señas mexicanas

La lengua de señas es una lengua natural de las personas sordas. Se diferencia de la lengua oral ya que utiliza el canal viso-gestual en lugar del audio-vocal [11] .Como toda lengua ésta varía en cada parte del mundo. La comunidad sorda de México utiliza la Lengua de Señas Mexicanas (LSM). Esta fue reconocida oficialmente como una lengua nacional y parte del patrimonio lingüístico de México el 10 de junio del 2005 [12]. Según [13] define la LSM como:

"La lengua de una comunidad de sordos, que consiste en una serie de signos gestuales articulados con las manos y acompañados de expresiones faciales, mirada intencional y movimiento corporal, dotados de función lingüística, forma parte del patrimonio lingüístico de dicha comunidad y es tan rica y compleja en gramática y vocabulario como cualquier lengua oral."

A través del tiempo los oyentes han formulado teorías sobre las personas sordas, las cuales son falsas. Entre esos mitos esta la idea de una lengua de señas universal. A pesar de que si existe un Sistema de Signos Internacional (SSI), este no es una lengua, es un sistema utilizado cuando no se comparte un mismo código lingüístico. Esto permite comunicarse y entenderse, sin importar del lugar de procedencia [14], pero contiene un léxico limitado y simplificado.

Teniendo claro que SSI es un sistema y no una lengua, es importante saber que cada comunidad lingüística tiene su propia lengua y no todos los sordos hablan una misma lengua de señas. Es decir, cada comunidad de sordos posee su propia lengua de señas, e inclusive en un mismo país se posee diferentes lenguas de señas, como es en el caso de México.

En la lengua de señas mexicanas una palabra puede representarse de manera diferente según la región del país y a pesar de que hay diccionarios oficiales de la lengua de señas como *Manos con voz* [15] de la ciudad de México y *Manual de lengua de señas mexicanas* [16] establecido en Puebla, estos pueden contener distintas traducciones para una misma palabra. Un ejemplo es la palabra piña, como se logra apreciar en la Figura 2.1.

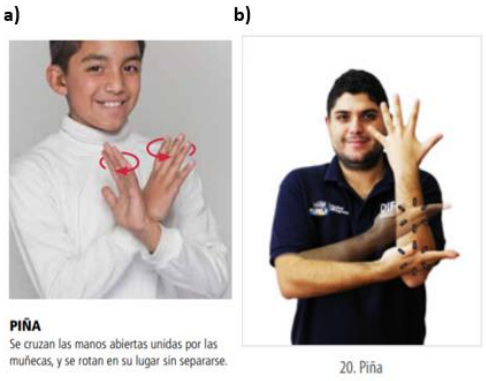

Figura 2.1 Traducción de piña en diferente diccionario. En la imagen a) se muestra la traducción de la palabra según el diccionario Manos con voz [15]. Mientras en la imagen b) se muestra la traducción según el manual de lengua de señas mexicanas [16].

Otro de los mitos que hay, es la idea de una estructura gramatical del lenguaje de señas, igual al del español. Muchos oyentes creen que la lengua de señas mexicanas es realizar signos tras signo siguiendo la estructura gramatical del español. Lo anterior no es LSM, más bien español signado, pues a pesar de ser más fácil entender el español signado para las personas oyentes hablantes del español, no es lo correcto. Esto es por la diferente forma de compresión de un sordo, en comparación de un oyente, por lo tanto, la gramática del LSM es completamente diferente a la gramática del español. La estructura gramatical de la LSM es Tiempo-Lugar-Objeto-Sujeto-Verbo, presentando variaciones dependiendo del tipo de verbo o por situaciones pragmáticas y semánticas [17].

Por el hecho de la complejidad de la lengua de señas, se han realizado diferentes tipos de investigaciones para desarrollar programas de cómputo como Sign'n [18];

aplicación móvil en desarrollo el cual realiza la interpretación del español (voz) al LSM mediante un avatar, utilizando tecnología de captura de movimientos.

Como se ha mencionado a lo largo de esta sección, la lengua de seña es un tema complejo, dando una gran área de oportunidad para realizar investigaciones y desarrollar una tecnología para aumentar la oportunidad de comunicación entre las personas sordas y oyentes.

2.2. Visión artificial

La vista humana es el sentido más avanzado que poseemos, el cual nos da la capacidad de poder procesar las imágenes ya sea para clasificar, analizar, entre otras cosas. Aun así, nuestra vista es limitada a las bandas visibles del espectro electromagnético. Por otro lado, las maquinas son capaces de percibir casi el espectro completo, desde los rayos gamas, hasta las ondas de radio, y estas pueden procesar imágenes generadas de fuentes, los cuales los humanos no asociamos con imágenes [19].

Aun no se ha definido con exactitud en que campo termina el procesamiento digital de imágenes (PDI) y comienzan otros campos como el análisis de imágenes y la visión por computadora (VC). Esto se debe a las diferentes opiniones de los autores [19]. En esta investigación se consideran tres tipos de procesos, comenzando con el PDI y finalizando con la VC. Estos tres procesos se clasifican en:

- Procesos de bajo Nivel: utilizan operadores para reducir el ruido, mejorar contraste, y filtros de enfoque. Se caracterizan porque tienen como entradas y salidas imágenes.
- Procesos de nivel medio: sus entradas son generalmente imágenes, pero tienen como salida atributos de la imagen de entrada.

- Procesos de alto nivel: implica el análisis de imágenes y realiza las funciones cognitivas asociadas con la vista.

2.2.1. Digitalización de una imagen

Según [20], una imagen es la representación óptica de uno o más objetos iluminados por una o varias fuentes de radiación. Esta puede ser definida matemáticamente como una función bidimensional, *f(x,y)*, donde *x* e *y* son coordenada espaciales, y *f* en cualquier par de coordenadas es la intensidad o nivel de gris de la imagen en esa coordenada [19]. Para manipular y almacenar una imagen dentro de una computadora, se tiene que digitalizar. Es considerada una imagen digital cuando todos los elementos de la función bidimensional son cantidades finitas y discretas [19]. Una imagen digital es considerada como un arreglo de pixeles, el cual contiene información de color en la ubicación de cada píxel [20]. Un píxel es la unidad mínima de una imagen digital y se puede apreciar en la Figura 2.2.

Figura 2.2 Vista de un píxel [21].

Al momento de digitalizar una imagen, esta se reconstruye. Para realizar una reconstrucción con la menor perdida de detalles, se debe de seguir el teorema de muestreo de Shannon. El cual establece que, para obtener una recuperación completa de la imagen, la frecuencia de muestro debe de ser dos o más veces mayores a la frecuencia contenida en el espectro de la señal original. Sí no se cumple lo anterior, el resultado será la perdida de detalle [22].

Lo anterior, da resultado a uno los parámetros más importantes de las imágenes digitales, la resolución de la imagen, el cual consiste en la cantidad de pixeles en una imagen. Entre mayor número de pixeles, mayor será el nivel de detalle. Comúnmente, la resolución se expresa en dos valores numéricos, donde el primero es la cantidad de pixeles por columna y el segundo es la cantidad de pixeles por fila [23].

2.2.2. Procesamiento digital de Imágenes

El procesamiento digital de imagen (PDI) es un conjunto de técnicas aplicadas a una imagen, mediante el uso de operadores o algoritmos, manipulando sus pixeles con el propósito de mejorar, comprender y extraer medidas de una imagen digital y de esta forma mejorar el proceso de interpretación visual de la misma, ya sea por parte de las personas o por el proceso de interpretación de datos por parte de una maquina autónoma [24] [22]. El PDI, es considerado un proceso de nivel bajo y nivel medio, ya que tiene como entrada imágenes, pero como salida puede tener una imagen, o atributos de la imagen de entrada [19].

Cabe mencionar, el PDI consiste en 4 fases generales (Figura 2.3) comenzando con la adquisición de la imagen digital, la cual puede ser adquirida de una forma directa o realizando un muestreo y cuantificación de esta. Después se realiza un preprocesamiento, permitiendo obtener la imagen con la menor cantidad posible de ruido no deseado, facilitando las siguientes fases. Como tercera etapa, se aíslan los objetos de estudio para poder analizarlos de forma individual, y a esta fase se le llama segmentación. Finalmente se extraen las características de la imagen, convirtiendo los datos en bordes o puntos de la imagen siendo legibles para el operador [25].

Figura 2.3 Fases del PDI [25].

El sistema del PDI se compone de sensores, hardware de procesamiento de imágenes especializado, computadora, software, almacenamiento en masa, dispositivos de despliegue de imágenes, dispositivos de copia dura y red [19].

2.2.3. Visión por computadora

La visión computacional (VC), también denominado visión artificial, es el estudio para el procesamiento de imágenes para el reconocimiento y localización de objetos de un entorno. Esto con el fin de desarrollar maquinas con la capacidad de comprender y percibir una imagen, de forma similar a como lo realizan los ser vivo [22]. Marr [26] definió a la visión como “un proceso que produce a partir de imágenes del mundo exterior una descripción que es útil para el observador y que no tiene información irrelevante.” Esta definición contiene tres aspectos importantes, los cuales son: la visión es un proceso computacional, la descripción depende del observador y toda información inútil se debe de descartar [22].

El objetivo de la VC es extraer características de una imagen para la interpretación y descripción por computadora. Ahora bien, existen diferentes arquitecturas y etapas para poder lograr el objetivo, esto dependerá específicamente lo que se desea realizar. De una forma general, se puede considerar el seguimiento de las

etapas que se ven en la Figura 2.4 ya que estas etapas se utilizan en la mayoría de los tiempos.

Figura 2.4 Etapas de un Sistema de visión por computadora [25].

Como se ha mencionado anteriormente, la visión por computadora y el procesamiento digital de imágenes se encuentran muy ligado, y así mismo presentan similitudes, pero presentan diferencias significativas. El procesamiento de imágenes, lo que hace es mejorar la imagen, con el propósito de que un externo le sea más sencillo comprender la imagen. Mientras la VC procesa la imagen para poder obtener información de esa misma e interpretarlos. Un ejemplo de VC se puede ver en la Figura 2.5, la cual muestra ciertas descripciones relevantes de una placa y obtiene una salida complementada con un módulo de reconocimiento de patrón para detectar las letras y los números.

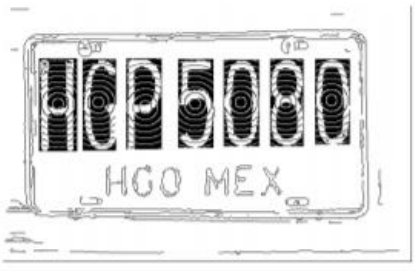

Figura 2.5 Reconocimiento de caracteres de una placa mediante visión artificial [22].

2.3. Computación grafica

Al escuchar la palabra computación grafica uno se puede imaginar la primera película de Pixar; Toy Story, por ser la primera película animada con mayor duración utilizando diseño por computadora. Pero la computación grafica ha existido desde tiempos anteriores a esa película. Desde su comienzo, esta tecnología ha sido de gran interés para los profesionales de todos los campos, desde entretenimiento como películas y video juegos, hasta el área de educación. Esta tecnología consiste en generar una imagen para realizar alteraciones, modificaciones y reformularla con el propósito de dejar una plantilla en la computadora, para ser utilizada en próximas necesidades [27].

En los comienzos de esta tecnología se encontraba limitada por los tipos de hardware y software con los que se contaban. Aun así, lograron agregarla en películas como Jurassic Park para darle vida a los dinosaurios en ciertos momentos. Con el tiempo, gracias a investigaciones y avances tecnológicos, el crecimiento de la computación grafica ha avanzado de una forma impresionante. Lo anterior se puede ver con la diferencia de la animación de la primera película de Toy Story con Toy Story 4, donde se nota mejor detalle de los materiales y los movimientos fluidos de los personajes.

En la industria de los video juegos se ha percatado en desarrollar sus personajes con el mayor parentesco de la realidad. Un claro ejemplo se puede ver en el famoso juego Tomb Raider con el personaje principal Lara (Figura 2.6). Si se compara Lara con su aparición en el primer Tomb Raider con Rise of the Tomb Raider, se puede notar como se ha logrado avanzar la tecnología dando mayor realismo es personaje y escenarios.

Figura 2.6 Comparación de Lara en Tomb Raider con Lara en Rise of the Tomb Raider [28].

Con lo anterior, podemos entender que la industria de los video juegos es la que ha tomado mayor provecho a la computación gráfica y es gracias a esta industria que se ha podido avanzar de forma rápida la mejora de esta tecnología. Donde en un principio los costos era cantidades elevadas por la dificultad de realizar animaciones utilizando supercomputadoras. Mientras ahora, se logra realizar personajes desde computadoras convencionales con la ayuda de algunos programas de cómputo de manera sencilla [29].

2.3.1. SMPL: A skinned multi-person linear model

A skinned multi-person linear model (SMPL) es un modelo realista del cuerpo humano, capaz de aprender a realizar los diferentes movimientos y posturas humanas, respetando las diferentes formas corporales. Al mismo tiempo es compatible con programas de computación gráfica, permitiendo el control de su animación, y se encuentra disponible con fines de investigación [30].

Los métodos tradicionales modelan como los vértices están relacionados de forma interna en una estructura esquelética. El modelo Linear Blend Skinning (LBS) es el más utilizado y es compatible con todos los programas para el desarrollo de video juegos y es eficiente para renderizar. La razón por ser el más utilizado, es por proveer deformaciones en la malla de forma aceptable con un buen rango de situaciones, sin la necesidad de utilizar intensidad computacional. Pero su desventaja es la perdida de volumen con movimientos pronunciados (como doblar un brazo). A este problema se han realizado diferentes investigaciones, al igual se han hecho investigaciones para obtener modelos de cuerpos con una alta calidad de realismo. Donde el método con mejor argumento está basado en la deformación de triángulos. A pesar de lo anterior, los modelos existentes carecen de realismo, no trabajan con paquetes existentes, representan un solo tipo de forma corporal, presentan incompatibilidad con gráficos piperline estándar, o son complicados de utilizar.

Por lo anterior, SMPL se formó con el objetivo de hacer el modelo lo más simple y estándar posible, para poder ser utilizado en diversas áreas y al mismo tiempo mantener el realismo de las deformaciones basadas en el aprendizaje de los datos.

Se utilizó la mezcla de formas (en inglés blendshape), con el propósito de corregir las limitaciones del trabajo estándar con la piel (en inglés skinning). Diferentes mezclas de formas para cada identidad, su pose y dinámica de tejidos blandos (en inglés soft-tissue dynamics) son combinados con una plantilla de apoyo (en inglés rest template) antes de ser transformados por un blend skinning. Para lograr hacer una animación y entrenamiento simple, se formuló el blendshape como una función lineal $B_s(\vec{\beta}): \mathbb{R}^{|\vec{\beta}|} \rightarrow \mathbb{R}^{3N}$, tomando como entrada un vector de los parámetros de la forma, $\vec{\beta}$, y la salida es un blendshape el cual modifica la identidad del sujeto.

Para capturar una dinámica de tejidos blandos, se extendió el SMPL adaptando el modelo Dyna (modelo matemático que sintetiza las deformaciones realísticas del cuerpo mediante dinámica de tejidos blandos [31]). Se tuvo como resultado el

modelo Dynamic-SMPL(DMPL), el cual se basa en vector en vez de deformación de triángulos. Esta extensión muestra como la deformación depende de la forma del cuerpo.

Como se puede ver en la Figura 2.7, el modelo SMPL descompone la forma del cuerpo entre identidades dependientes de la forma y posturas no rígidas dependientes de la forma. El modelo está definido por un template shape representado por un vector N concatenado con el vértice $\overline{\boldsymbol{T}} \in \mathbb{R}^{3N}$ en la posición cero, $\vec{\theta}^{*}$; un conjunto de blend weigths, $\omega \in \mathbb{R}^{N\,x\,K}$; una función blendshape; una función para predecir la ubicación de las uniones, $J(\vec{\beta})\colon \mathbb{R}^{|\vec{\theta}|} \rightarrow \mathbb{R}^{3K}$ como una función del parámetro de la forma, $\vec{\beta}$; y una función de la posición dependiente del blend shape, $B_p(\vec{\theta})\colon \mathbb{R}^{|\vec{\theta}|} \rightarrow \mathbb{R}^{3N}$. Finalmente se aplica una función blend Skinning , W para rotar los vértices alrededor del centro de unión estimada definida por el blend weigths [30].

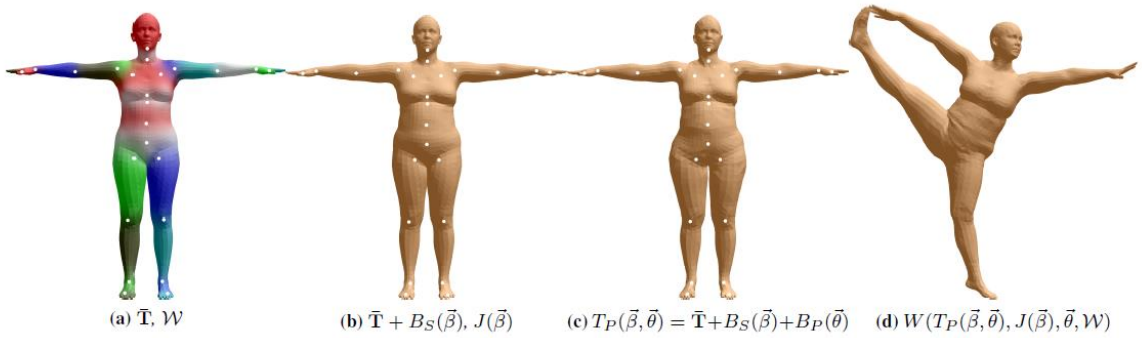

Figura 2.7 a) template mesh con el blend weights indicado por colores y las uniones mostradas en blanco. b) Identidad del sujeto con solo la contribución del blendshape. c) deformación de las mallas para la preparación de la postura, se puede notar en el área de las caderas. d) obtención de la postura del sujeto. [30]

Con lo anterior, se define el modelo con la siguiente ecuación:

$$M(\vec{\beta}, \vec{\theta}; \Phi) = W\left(T_p(\vec{\beta}, \vec{\theta}; \overline{\boldsymbol{T}}, S, P), J(\vec{\beta}; \mathcal{J}, \overline{\boldsymbol{T}}, S), \vec{\theta}, W\right)$$

Donde Φ representa el conjunto completo de los parámetros del modelo el cual es igual a $\Phi = \{\overline{\boldsymbol{T}}, W, S, \mathcal{J}, P\}$.

SMPL eXpressive (SMPLX)

A partir de SMPL se han generado varios tipos de proyectos los cuales tienen el propósito de mejorar el modelo SMPL, uno de ellos es SMPLX [32]. El modelo SMPLX implementa al modelo SMPL la posibilidad de hacer el modelo más realista, ya que ha agregado la función de hacer el modelo expresivo, es decir es capaz de modificar las expresiones faciales del modelo. Así mismo, cuenta con la implementación de modificar la configuración de la mano, dando la oportunidad de poder dar movimiento a los dedos y manos de los modelos, siendo una de las piezas claves para dar mayor realismo. Estas dos funciones agregadas en SMPLX, es una de los puntos fundamentales para poder generar una interpretación en LSM, por su oportunidad de poder modificar los movimientos de las manos y mostrar los gestos faciales.

2.4. Trabajos Relacionados

Con el paso del tiempo, las personas han detectado la necesidad de mejorar la comunicación entre oyentes y sordos, por lo tanto, se han hecho diferentes investigaciones con el fin de resolver este problema.

En esta sección se mencionarán algunos de los trabajos de investigación que cuentan con una relación con este trabajo de tesis.

2.4.1. Modelado 3D del lenguaje de señas mexicano para un sistema de voz-a lengua de señas

En el trabajo de investigación [33], se desarrolló un sistema de traducción español a lengua de señas mexicana (LSM), con la utilización de un modelo 3D (avatar) y reconocimiento de voz. Se desarrolló con el propósito de implementar una tecnología para el mejoramiento de comunicación entre una persona sorda y un oyente. Este consta de una plataforma interactiva con el usuario, donde se puede seleccionar una de las palabras del vocabulario o frases ya establecidas, mientras el avatar realiza la traducción en LSM. Así mismo, se tiene la opción de entrenar el avatar, donde se dicta una frase y el avatar aprende a realizar la interpretación.

Para la realización del avatar se hicieron capturas de los movimientos de un intérprete de LSM, utilizando el Kinect Microsoft como sensores 3D. Este dispositivo se utilizó por su bajo precio y su eficacia para detectar los movimientos. Al tener la captura del movimiento se aplicó un procesamiento de imagen, con el fin de obtener solo el movimiento de la seña eliminando los datos externos, en este caso el fondo. El procesamiento consiste en aplicar un filtro de segmentación, donde se elimina el fondo, ya que el fondo fue el mismo para todas las tomas. Al tener solamente la seña correspondiente y la percepción del cuerpo del intérprete, se genera un esqueleto para seguir los movimientos de la seña y poder mapearlos a un avatar 3D. En el caso de los movimientos de las manos y los dedos, se implementó una estructura esquelética para la mano y así realizar el mapeado correspondiente.

Para lograr generar el avatar, los movimientos obtenidos se exportan a un archivo con formato bvh (biovision hierarchical data). Esos archivos se utilizan con DAZ Studio 4, para mapear los movimientos del LSM al modelo 3D. En este caso, el avatar es un modelo neutro, sin género ni textura.

Al tener generado el avatar, se prosiguió con el reconocimiento de voz utilizando HTK. Para el entrenamiento del reconocimiento de voz, se necesitó la ayuda de diferentes personas que contaban con el mismo acento de la región. Este

reconocimiento de voz se adaptó al avatar, con el propósito de reconocer la palaba dictada por el hablante. Si la palabra se encuentra en la base de datos se prosigue con realizar la seña correspondiente por el avatar. Sin embargo, si la palabra no pertenece en la base de datos, el avatar prosigue con deletrear la palabra en señas.

Este sistema tiene un desempeño del 96.2%, con 1990 palabras. A pesar de mostrar un desempeño satisfactorio, este no cuenta con la traducción del español al LSM en tiempo real y cuenta con solo 70 palabras en total en la base de datos.

2.4.2. Interacción con avatar 3D para la creación y edición de poses corporales y faciales en la lengua de señas costarricense LESCO

PIELS [34] es una plataforma con el fin de crear y editar un avatar 3D para la interpretación de la lengua de seña costarricense (LESCO). Con esta plataforma se edita los movimientos de la mano y expresiones faciales para realizaron una señal determinada. Cuenta con la sección de mano, donde se cuenta con terminologías correspondientes a los parámetros gramaticales. Así mismo, se cuenta con la edición de los dedos, donde si se desea hacer un movimiento no predeterminado en la plataforma, se realiza de manera manual.

En cuanto a la edición de las expresiones faciales, se utiliza la “rueda de las emociones” propuesta por el psicólogo estadunidense Robert Plutchik (Figura 2.8) donde divide las emociones en 8 categorías básicas y 8 categorías avanzadas.

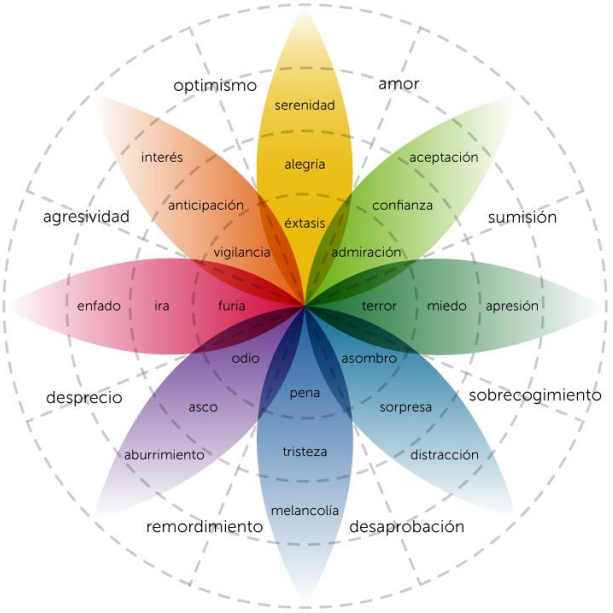

Figura 2.8 Ruleta de las emociones [35]

Para la validación de la plataforma se realizó una encuesta a 60 participantes con un rango de edad de 18 a 50 años. Donde el 46.15% presentaron dificultad para crear una seña. Esta plataforma es una herramienta útil para la creación de las señas por avatar. Sin embargo, el sistema es manual lo cual toma bastante tiempo y proceso para realizar cada seña.

2.4.3. Traductor de texto y voz a lengua de señas ecuatoriana a través de un avatar implementado para dispositivos Android

En el artículo [36] se presenta una plataforma para dispositivos Android capaz de realizar traducciones de voz o texto del español a la lengua de señas ecuatoriana (LSEC). La interpretación en LSEC se realiza con un avatar, por ser una herramienta agradable para el Sordo, siendo más sencillo de entender. Este avatar se desarrolló con el software Daz Studio 4.9 Pro, el cual cuenta con diferentes platillas para la elección de la apariencia del avatar. En este caso utilizaron la platilla Génesis 3

Male. Al mismo tiempo, con la ayuda del software DAZ studio se realizaron las animaciones, dando lugar a los movimientos correspondientes para la formación de las diferentes palabras.

En el caso del reconocimiento de voz, se utilizó GOOGLE NOW, el cual consta de 16000 redes multicapas, facilitando el reconocimiento de las palabras dichas por el orador. Lo anterior cuenta con un 5% de margen de error. Así mismo la plataforma consta de 120 palabras en su base de datos, donde se tiene una limitación de 25 caracteres a las frases a traducir.

2.4.4. Deaf Talk Using 3D Animated Sign Language A Sign Language Interpreter using Microsoft's Kinect v2

Los desarrolladores del artículo [37], tomaron en cuenta el problema de comunicación entre personas sordas y oyentes. Donde el sordo se le dificulta entender al oyente, así mismo, el oyente no entiende la lengua de señas. Por lo anterior, [37] desarrollaron una plataforma utilizando el Kinect v2 de Microsoft con dos funciones: traducción del inglés (voz) a la lengua de señas (LS), y traducción de la lengua de señas al inglés.

Para realizar esta plataforma, se modificó el Kinect, mejorando la resolución de la imagen y la detección del audio. La resolución de la imagen aumento de 640x480 pixeles a 1920 x 1080 pixeles, ayudando en la detección de los gestos. Mientras el audio se incrementó de 16 bits de canal a 62 bit de canal, mejorando la calidad del reconocimiento del audio. Así mismo, el Kinect v1 sin modificar detecta 20 huesos del esqueleto humano. Al realizar las mejoras, se detectan 26 huesos, incluyendo 2 huesos más en cada mano para comprender los movimientos de los dedos, el cual es esencial para la comprensión del LS.

En el primer módulo (ingles a LS) se utilizó la estrategia mostrada en la Figura 2.9 flujo del módulo ingles a seña . Se comienza con la persona oyente hablando

enfrente al Kinect. Este detecta la palabra o frase dicha y lo transforma a texto. Después el sistema toma el texto y realiza una búsqueda en la base de datos implementada, encontrando la seña correspondiente y después manda esa seña al módulo de visualización. Con Unity3D se mapea el gesto correspondiente al modelo 3D (avatar). Finalmente, el avatar se muestra en la pantalla realizando la seña correspondiente.

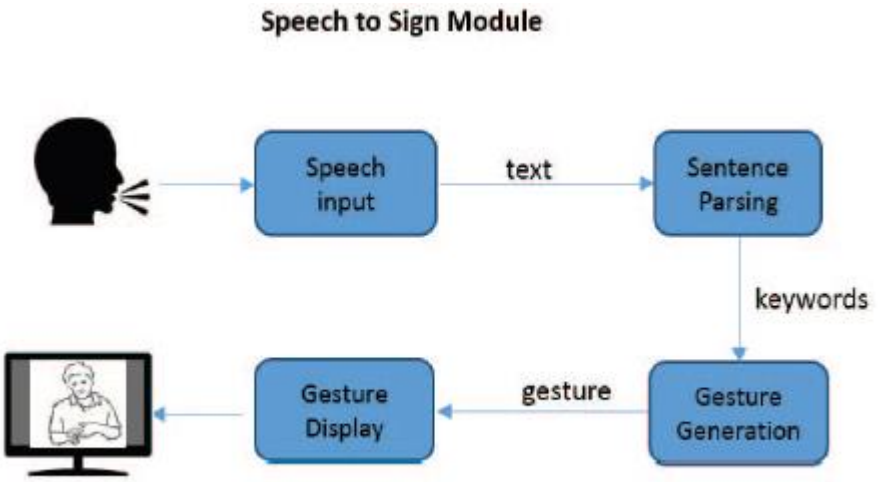

Figura 2.9 flujo del módulo ingles a seña [37]

Para el segundo módulo (LS a inglés), Kinect v2 SDK provee una herramienta útil, llamada gesture builder, el cual fue diseñado con el propósito de reconocer gestos simples. Esta herramienta les brindo la simplicidad de abstracción de los detalles, eliminando la necesidad de realizar cálculos de coordenadas, ángulos, estados y profundidad al reconocer los gestos. Así mismo, esta herramienta cuenta con entrenamiento de los gestos utilizando aprendizaje automático. Aun así, se le implemento librerías para la base de datos de las señas.

En el caso del módulo seña a voz se sigue el flujo mostrado en la Figura 2.10. Primero, la persona sorda se coloca enfrente del Kinect y realiza las señas correspondientes. El sistema detecta los gestos y busca en la base de datos los

gestos que tengan similitud con los detectados. Después se construye la frase utilizando los textos correspondientes al gesto. Y finalmente se muestra la frase formada, junto con el audio en inglés diciendo la frase.

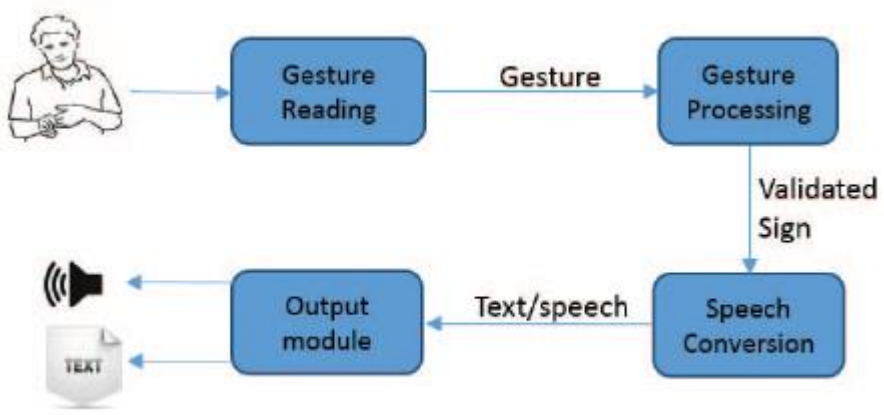

Figura 2.10 flujo del módulo seña a voz [37]

Para dar validación al proyecto, se realizaron pruebas para detectar el porcentaje de precisión y desempeño. Se utilizaron 3 personas con diferentes compleciones físicas y se realizaron 100 pruebas. Donde se obtuvo una precisión del 85% del módulo LS a voz y un 87% para voz al LS. Se presentaron errores en determinados tipos de cuerpos y también no se detectaba de forma correcta acentos no nativos.

CAPÍTULO 3. Diseño del Modelo para la Creación y Uso de un Repositorio Digital de Segmentos Visuales en Lengua de Señas Mexicanas.

3.1. Introducción

En este capítulo se da a conocer el diseño del modelar, para la creación y uso del repositorio digital de segmentos visuales en la lengua de señas mexicanas. Se presenta la arquitectura general del modelo, continuando con la descripción de cada módulo compuesto, los cuales consisten en 4 secciones: módulo de extracción de video de señas LSM; módulo de transformación 3D; módulo de recuperación de segmentos visuales; y el módulo de la generación del video de la seña LSM.

3.2. Arquitectura del Modelo

Para diseñar la arquitectura de la creación y uso de un repositorio digital de segmentos visuales en LSM, se comenzó realizando el diseño de la creación del repositorio continuando con el diseño para su utilización. Esta arquitectura se puede ver en la Figura 3.1.

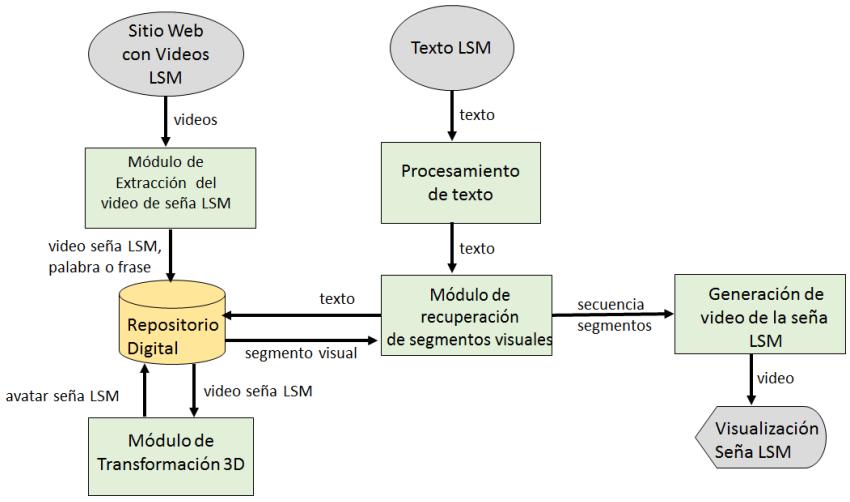

Figura 3.1 Arquitectura para la creación y uso de un repositorio digital de segmentos visuales en Lengua de Señas Mexicanas. Fuente: Elaboración propia.

Para la creación del repositorio digital de segmentos visuales en LSM, se decidió obtener las palabras de la LSM junto con su traducción al español, mediante videos encontrados en la web, proporcionado por intérpretes o personas sordas. Estos videos deben de contar con un intérprete diciendo la palabra en LSM, y la traducción al español en texto o en voz. La extracción de estos videos se realiza en el módulo de extracción del video de seña LSM. Estos videos se almacenarán en el repositorio digital vinculados con la palabra traducida al español.

Teniendo almacenado los videos, se prosigue con el módulo de transformación 3D. Este módulo consiste en transformar los movimientos de la persona a movimientos 3D, dentro de un avatar utilizando la tecnología SMPL. Al tener estos movimientos transformados serán guardados en el repositorio digital con su respectiva traducción al español.

Con lo anterior se tiene diseñado la parte de la creación del repositorio digital. Ahora, para el diseño de su utilización, hay que tomar en cuenta que se partirá desde un texto LSM donde ya se procesó desde la lengua natural del español, hasta el texto con gramática del LSM, trabajo realizado por [3] y [9].

El texto LSM se separa por palabras o frases ya establecidas en el repositorio digital y se procede al módulo de recuperación de segmentos visuales. En este módulo, se llaman los segmentos visuales correspondientes al texto dado, dando una secuencia de los movimientos, generando un video de la seña LSM. Con esto se visualizará las señas en LSM del texto correspondiente mediante un avatar.

A continuación, se explica de forma más específica cada módulo de la arquitectura establecida.

3.3. Módulo de Extracción del Video de Seña LSM

En la Web se encuentran videos de la Lengua de Señas Mexicana (LSM) con el propósito de enseñanza o divulgación. Particularmente en sitios web como YouTube y en algunas páginas Web, que contienen diccionarios de LSM, se utilizan dos maneras para dar la descripción de una seña determinada. La primera es mediante el lenguaje verbal; es decir, una persona dice la palabra a traducir y procede a realizar la seña. La segunda es mediante texto, donde se tiene una anotación con texto dentro del video, o se tiene asociados a la referencia del video la frase o palabra.

3.3.1. Extracción de secuencia de fotogramas y su correspondiente texto de videos de LSM

Existen videos de LSM que muestran la frase o palabra que describe la seña LSM; por lo tanto, es necesario realizar un procedimiento que permita obtener la secuencia de fotogramas y su correspondiente texto. En la Figura 3.2 se muestra un diagrama para realizar este procedimiento, y la correspondiente descripción de sus componentes.

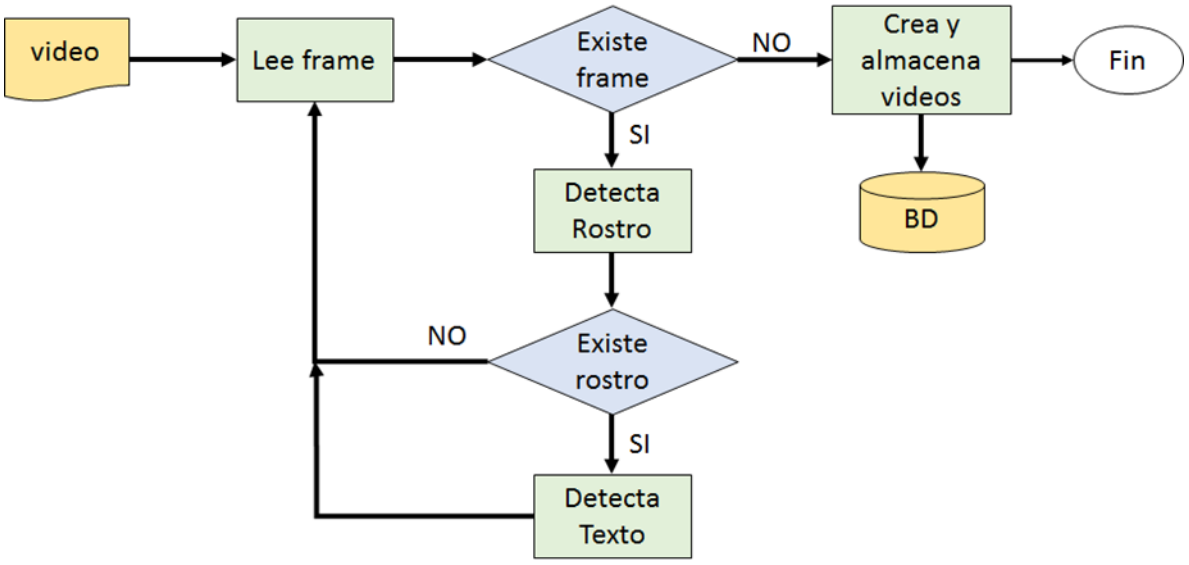

Figura 3.2 Diagrama de bloques del procedimiento de extracción del video y texto. Fuente: Elaboración propia.

Una vez que se lee el video, se procesa analizando cada fotograma (*fotograma*), de manera que cada fotograma corresponde a una imagen. En la mayoría de los videos, se tiene una secuencia introductoria como se muestra en la Figura 3.3, donde no se tiene propiamente la seña LSM, de forma que estos fotogramas se descartan y se consideran aquellos donde aparece la persona.

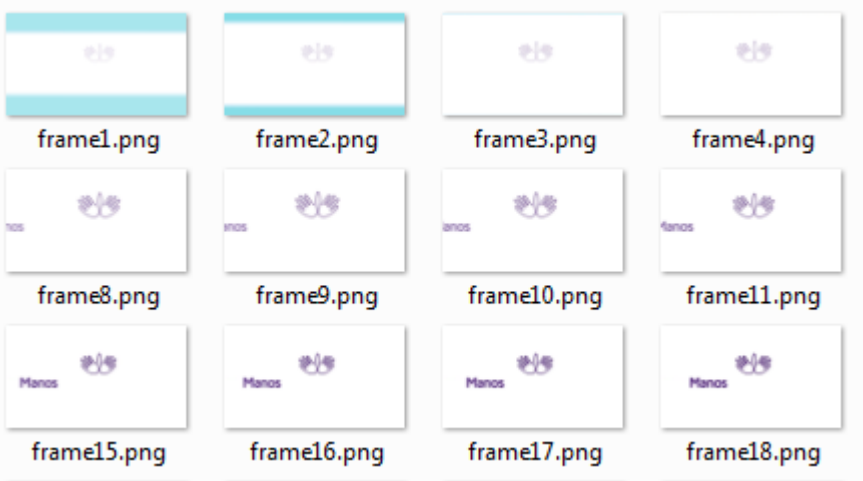

Figura 3.3 Secuencia de fotogramas que se descartan. Fuente: obtenido de videos públicos de YouTube.

En las siguientes subsecciones, se describen las funcionalidades que se aplican para el análisis de cada fotograma y poder determinar la palabra o frase, así como los fotogramas correspondientes al video de la seña LSM.

Reconocimiento y Obtención de Texto en una Imagen

La Figura 3.4 muestra imágenes contenidas en videos que contienen texto para describir la seña en LSM. Estas imágenes se obtuvieron una vez descargado el video de YouTube.

Figura 3.4 Ejemplos de imágenes con texto para describir la seña LSM [38] [39].

Como se aprecia en la Figura 3.4, se puede tener una sola palabra o una frase u oración; asimismo, el texto puede tener las palabras de manera horizontal o vertical. A continuación, se describen los pasos para obtener el texto de la imagen.

Paso 1. Cargar la imagen

Para realizar esta funcionalidad se usa la librería OpenCV, de manera que, una vez cargada la imagen, se obtiene el ancho y altura de la imagen original (Figura 3.5 lado izquierdo) y se ajusta a un tamaño de 320x320 (Figura 3.5 lado derecho), para usarla en el reconocimiento de patrones, en este caso el texto.

Figura 3.5 Cambiar el tamaño de la imagen a 320x320. Fuente: obtenido de videos públicos de YouTube

Paso 2. Reconocimiento de los patrones de texto usando el modelo EAST

En la librería OpenCV 4, se encuentra un detector de texto en escenas basado en el aprendizaje profundo llamado EAST (**E**fficient and **A**ccurate **S**cene **T**ext Detector), propuesto por Zhou et al [40].

El detector de texto EAST de OpenCV, es un modelo de aprendizaje profundo, basado en una arquitectura y un patrón de entrenamiento novedoso; una técnica bastante robusta, capaz de localizar texto incluso cuando está borroso, reflejado u oculto parcialmente, que supera significativamente a los métodos más avanzados en términos de precisión y eficiencia. EAST básicamente detecta texto en una imagen (o video) y proporciona geometría y puntajes de confianza para cada bloque de texto que detecta.

La red neuronal totalmente convolucional (FCN, por sus siglas en inglés), predice directamente palabras o líneas de texto de orientaciones arbitrarias, cuando se enfrentan a escenarios desafiantes, eliminando pasos intermedios innecesarios (por ejemplo, agregación de candidatos y partición de palabras) computacionalmente costosos, que otros detectores de texto aplican típicamente. En la Figura 3.6 se

muestra la arquitectura de la red neuronal totalmente convolucional, usada en la detección de texto en escenas.

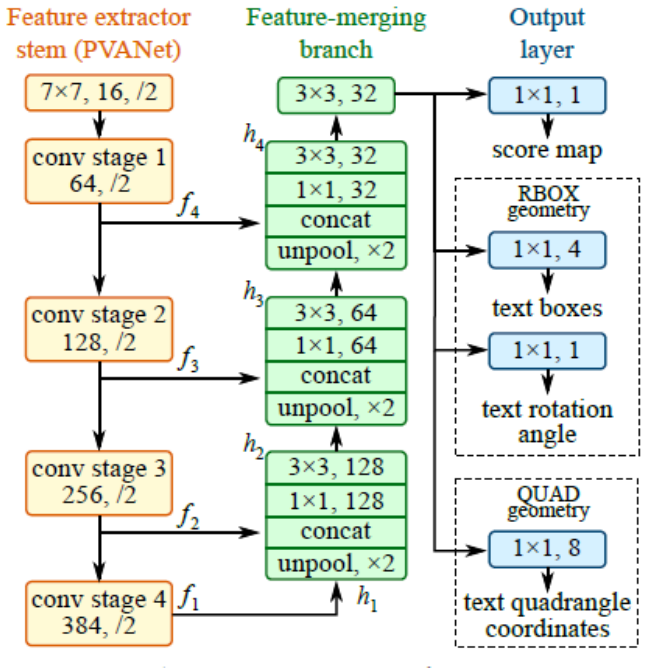

Figura 3.6 Estructura de la FCN para la detección de texto, tomada de [40]

Como se presenta en la Figura 3.6, en la capa de salida se tiene un mapa para las predicciones (score), la salida para la geometría de las cajas que encierran al texto, así como sus ángulos de rotación; adicionalmente, se tiene la geometría QUAD, agregando un término de normalización adicional diseñado para cuadriláteros de palabras, que suele ser más largo en una dirección.

Este modelo usa red pre-entrenada "frozen_east_text_detection.pb" que se puede descargar de Internet y se construye un blob de la imagen y ejecuta un paso hacia adelante en el modelo, para obtener los dos conjuntos de salida: geometry y scores, que corresponden a la geometría de las cajas de texto y las probabilidades correspondientes.

Para suprimir delimitaciones débiles y cajas superpuestas se aplica la función "Non-Maxima Suppression" [41], esta función es una parte muy importante en el proceso

de detección de objetos. Al buscar objetos en una imagen, generalmente se encuentran varios puntos como objetos, pero algunos de ellos no son realmente objetos. Supresión no máxima (NMS) consiste en seleccionar cuáles de esos máximos son realmente objetos y suprimir los que no lo son, como se muestra en la Figura 3.7.

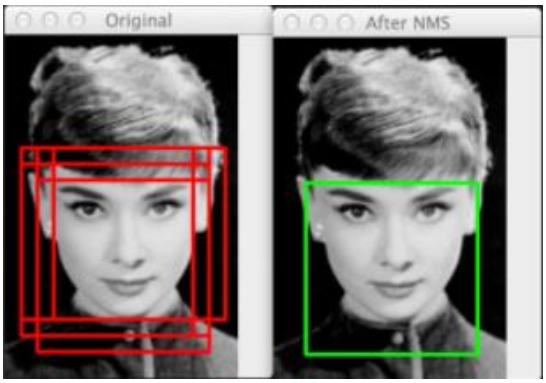

Figura 3.7 Aplicación de NMS [41]

Paso 3. Análisis para determinar el área de texto

Una vez que se tienen las cajas de texto definitivas, esto es, los patrones encontrados y que tienen la certeza necesaria, se realiza primero un análisis para encontrar los falsos positivos con respecto a la solución del problema, en este caso, es encontrar el texto referente a la seña de LSM. Esto es necesario, porque se pueden tener imágenes como la que muestra en la Figura 3.8, donde se aprecia texto adicional que no es de interés. Para comprender esto, se han marcado con un cuadro rojo estos casos y con verde el correcto.

Figura 3.8 Detección de falsos positivos [42]

También se determina si el texto se encuentra con una orientación horizontal o con una orientación vertical. De tal manera que se recalculan las coordenadas para obtener una sola caja que encierre el texto de interés. Para comprender mejor el procedimiento descrito previamente, se presentan los siguientes casos: una palabra de texto de manera horizontal y de texto de manera vertical.

Caso 1: Una sola palabra. En la Figura 3.9 se muestra la imagen original para el caso de una sola palabra.

Figura 3.9 Imagen original con una palabra para la seña [42].

En la Figura 3.10 se muestra la imagen con la detección de los textos en la imagen, se puede apreciar que ha encontrado dos textos y los ha marcado con los marcos en verde.

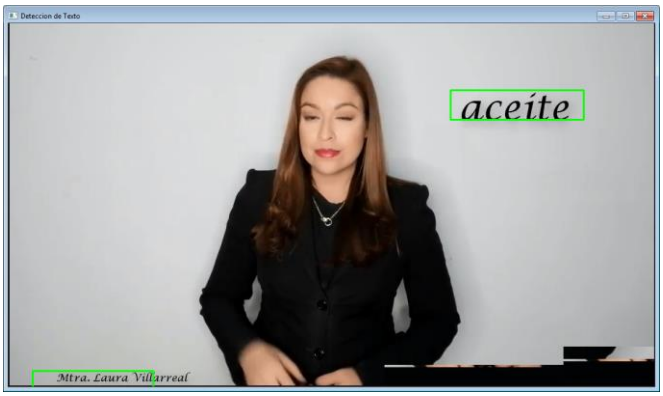

Figura 3.10 Detección del texto de interés [42]

Una vez que se realice el proceso para determinar el objeto de interés y discriminar los falsos positivos, entonces se ajusta las coordenadas para asegurar que se tenga todo el texto. Como se muestra en la Figura 3.11.

Figura 3.11 Detección del texto de interés [42].

Paso 4. Obtención de texto como cadena de caracteres

Una vez obtenida el área donde se encuentra el texto, entonces, se recorta esa área y se convierte a una imagen binaria para facilitar la obtención de la cadena de caracteres, a través del reconocimiento óptico de caracteres (OCR del inglés Optical Character Recognition).

En la Figura 3.12, se muestra la conversión de la imagen de color a binario inverso.

Figura 3.12 Transformación de la imagen del texto. Fuente: Elaboración propia.

Finalmente, para realizar la obtención de la cadena de caracteres usando el reconocimiento óptico de caracteres, se usa Tesseract que es un motor de reconocimiento óptico de caracteres para varios sistemas operativos. Es software libre, liberado bajo la licencia Apache, Versión 2.0. La cadena de caracteres es procesada para eliminar espacios, nuevas líneas y otros caracteres en los extremos de la cadena de caracteres. El resultado obtenido está en minúsculas.

Caso 2: Varias palabras de manera horizontal

En la Figura 3.13 se muestra la imagen original para el caso de varias palabras de manera horizontal.

Figura 3.13 Varias palabras de manera horizontal. [38]

La Figura 3.14 muestra la detección del texto y como se puede apreciar se pueden tener varias cajas detectadas por la red.

Figura 3.14 Detección de texto de manera horizontal. [38]

La Figura 3.15 muestra la detección del texto de interés.

Figura 3.15 Detección del texto de interés. [38]

De tal manera que una vez terminado el proceso del texto de interés, se obtiene la región que se muestra en la Figura 3.16 y finalmente se obtiene la cadena: “me das un ejemplo”.

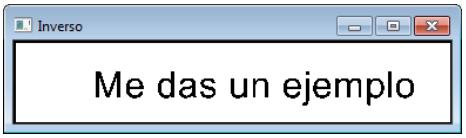

Figura 3.16 Imagen con la región de interés. Fuente: Elaboración propia.

Caso 3: Varias palabras de manera vertical

En la Figura 3.17 se muestra la imagen original para el caso de varias palabras de manera vertical, inclusive, de manera horizontal se tiene varias palabras. De tal manera, que el objetivo es encontrar la región de interés que incluya todo el texto.

Figura 3.17 Texto de manera vertical y horizontal [39].

La Figura 3.18 muestra la detección del texto y como se muestra, se pueden tener varias cajas detectadas por la red.

Figura 3.18 Detección de texto de manera vertical y horizontal [39].

La Figura 3.19 muestra la detección del texto de interés.

Figura 3.19 Detección de texto de interés [39].

De tal manera que, terminado el proceso del texto de interés, se obtiene la región que se muestra en la Figura 3.20, y finalmente se obtiene la cadena: “¿Dónde te veo?”.

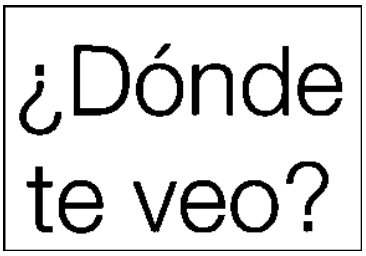

Figura 3.20 Imagen con la región de interés. Fuente: Elaboración propia.

Creación de video de la seña LSM y su almacenamiento en la base de datos

Al analizar los fotogramas (fotogramas) del video, se fue creando una tabla con las palabras que fueron encontradas y los fotogramas correspondientes al video. A continuación, se muestra un ejemplo de esto, donde se muestra que cada fila contiene la palabra o en su caso la frase, y tiene asociado el *fotograma* de inicio y el *fotograma* final, los tres siguientes valores (0, 0, 0) representan el color que se usará para colorear el cuadro donde se encontró la palabra; finalmente los últimos cuatro valores son las coordenadas del cuadro del texto.

[('familia', 677, 745, 0, 0, 0, 156, 326, 470, 395),

('papá', 738, 805, 0, 0, 0, 160, 328, 422, 402),

('padre', 798, 855, 0, 0, 0, 148, 337, 442, 402)]

En la Figura 3.21 se muestran parcialmente los *fotogramas* correspondientes a la seña “familia”. Esto es del *fotograma* 677 hasta el *fotograma* 745 se usan para crear el video de la seña.

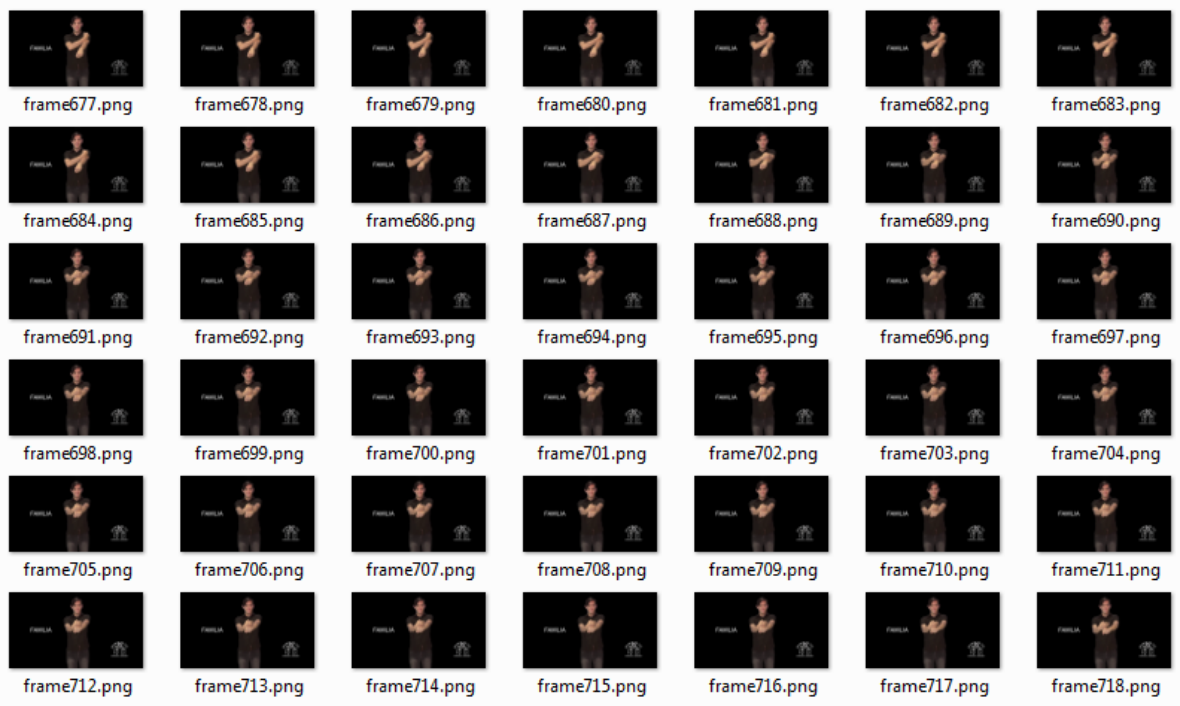

Figura 3.21 Algunos fotogramas de la seña "familia". Fuente: obtenido de videos públicos de YouTube

Una vez creado el video se almacena junto el texto en una tabla de una base de datos, la cual se usará para el proceso posterior de transformar el video a un modelo 3D.

3.3.2. Obtención de videos de señas LSM y su correspondiente texto de videos LSM de páginas Web

Existen en la Web diccionarios de la LSM que contienen una lista de palabras o frases en LSM, de tal manera que al seleccionar una de ellas, se redirecciona a una página Web, donde se encuentra el video asociado en Youtube. En la Figura 3.22 se muestra un diagrama de flujo que permite ilustrar el procedimiento.

Un ejemplo de este tipo de diccionario se encuentra en [43] y como se presenta en la Figura 3.23, se puede seleccionar la palabra o frase que se encuentra en orden alfabético.

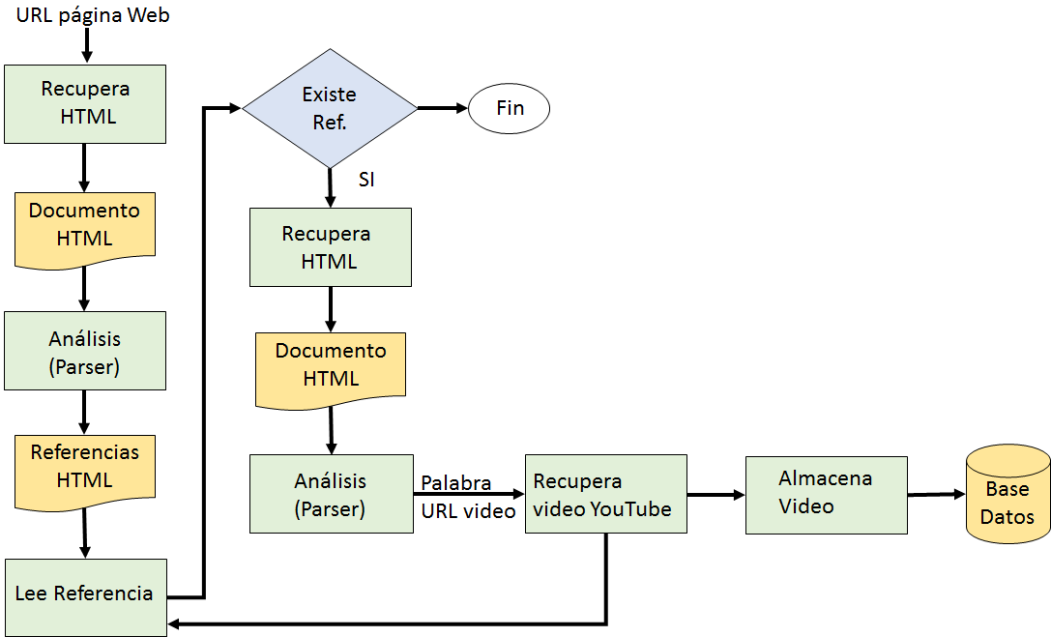

Figura 3.22 Procedimiento para recuperar el video de la seña LSM y su palabra asociada de un diccionario en la Web [43]

Figura 3.23 Diccionario wikisigns [43]

De manera como se muestra en la Figura 3.24, en el documento HTML se tiene la palabra y tiene asociada la referencia de la página Web donde se puede ejecutar el video. Por lo tanto, en este caso se tiene que realizar un análisis del documento HTML para obtener los elementos que lo componen, una vez hecho esto, se localiza el elemento donde se encuentran las referencias, como se aprecia en la Figura 3.23.

```
        <a href="/es/lsm/Abreviatura">Abreviatura</a>    </div>
<div class="views-row views-row-23 views-row-odd">

        <a href="/es/lsm/abril">abril</a>    </div>
<div class="views-row views-row-24 views-row-even">

        <a href="/es/lsm/Abrir%20la%20Puerta">Abrir la Puerta</a>    </div>
<div class="views-row views-row-25 views-row-odd">

        <a href="/es/lsm/Absorber">Absorber</a>    </div>
<div class="views-row views-row-26 views-row-even">

        <a href="/es/lsm/Abstenerse">Abstenerse</a>    </div>
<div class="views-row views-row-27 views-row-odd">

        <a href="/es/lsm/absurdo">absurdo</a>    </div>
<div class="views-row views-row-28 views-row-even">

        <a href="/es/lsm/abuela">abuela</a>    </div>
<div class="views-row views-row-29 views-row-odd">

        <a href="/es/lsm/abuelo">abuelo</a>    </div>
<div class="views-row views-row-30 views-row-even">

        <a href="/es/lsm/Acabar">Acabar</a>    </div>
<div class="views-row views-row-31 views-row-odd">
```

Figura 3.24 Referencia en el documento HTML [43].

Estas referencias son utilizadas para obtener el documento HTML, donde se encuentra la liga para ejecutar el video seleccionado; por lo tanto, se realiza un proceso similar al descrito anteriormente y se obtiene la referencia al video en Youtube, como se presenta en el siguiente código. Una vez que se tiene la referencia se descarga el video y se almacena en la base de datos junto con la palabra.

<ifotograma class="media-youtube-player" width="438" height="266" title="IMG 9423"
src**="https://www.youtube.com/embed/0DY16S_PCDw**?wmode=opaque&controls=&modestbranding=1&rel=0&showinfo=0&theme=light&color=white&autohide=1" name="IMG 9423" fotogramaborder="0" allowfullscreen>Video of IMG 9423</ifotograma>

3.3.3. Obtención de videos de señas LSM y su correspondiente texto del diccionario DIELSEME

Como se establece en [44], “El Diccionario Español – Lengua de Señas Mexicana (DIELSEME) es un diccionario bilingüe que responde a la necesidad básica de enseñar la lengua de señas mexicana (LSM) con referencia al español escrito, a personas Sordas, así como a una diversidad de usuarios potenciales: padres oyentes que tienen hijos sordos, maestros de todos los niveles educativos que atienden alumnos sordos, intérpretes en formación, escuelas de intérpretes y la sociedad en general”.

Este diccionario se encuentra publicado tanto en papel como en versión digital. Para el objetivo de este proyecto se usó la versión digital que se encuentra bajo la denominación DIELSEME 1, se leyeron los videos que se encuentran en formato swf y se convirtieron al formato mp4, de tal manera que fueron almacenados en la

base de datos junto con la palabra asociada al video. Lo anterior se muestra en la Figura 3.25.

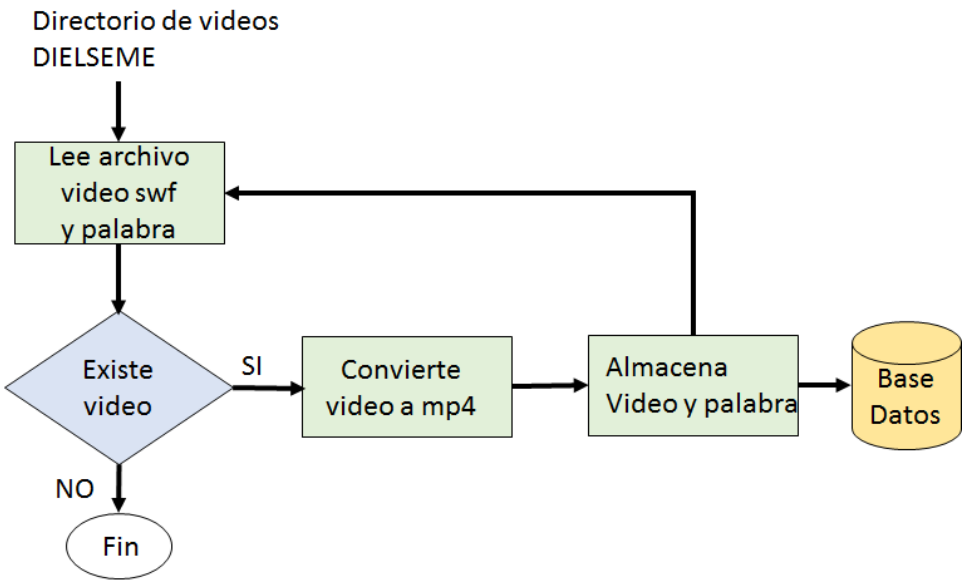

Figura 3.25 Obtención de video y palabra del diccionario DIELSEME. Fuente: Elaboración propia.

3.4. Módulo de Transformación en 3D

Detectar la postura de una persona dentro de una imagen o video 2D y mapearlo a un plano 3D, es una tarea compleja, sobre la que se han realizado múltiples investigaciones, exponiendo diversos métodos. Uno de los métodos más utilizados es extraer las características 2D del cuerpo, donde se detectan los puntos de unión principales (keypoints) para determinar la postura de la persona. Asimismo, se han utilizado métodos matemáticos y redes neuronales entrenadas, para mapear los puntos de unión en un plano 3D [32] .

El método que se utiliza en este trabajo es la extracción de características 2D, con el propósito de incorporar los movimientos de las personas (intérpretes LSM) en el modelo 3D (Avatar).

Para lograr el objetivo de este módulo se diseñó el algoritmo mostrado en la Figura 3.26. Como se puede ver, se inicia con la recuperación de un determinado video desde el repositorio digital. Se extraen las características 2D de los movimientos del interprete, y se incorporan al modelo 3D, dando como resultado un archivo obj. El anterior archivo se transforma en un archivo png para posteriormente unir todas las posturas del avatar y transformarlo a un video en formato mp4. El video creado se almacena dentro del repositorio digital.

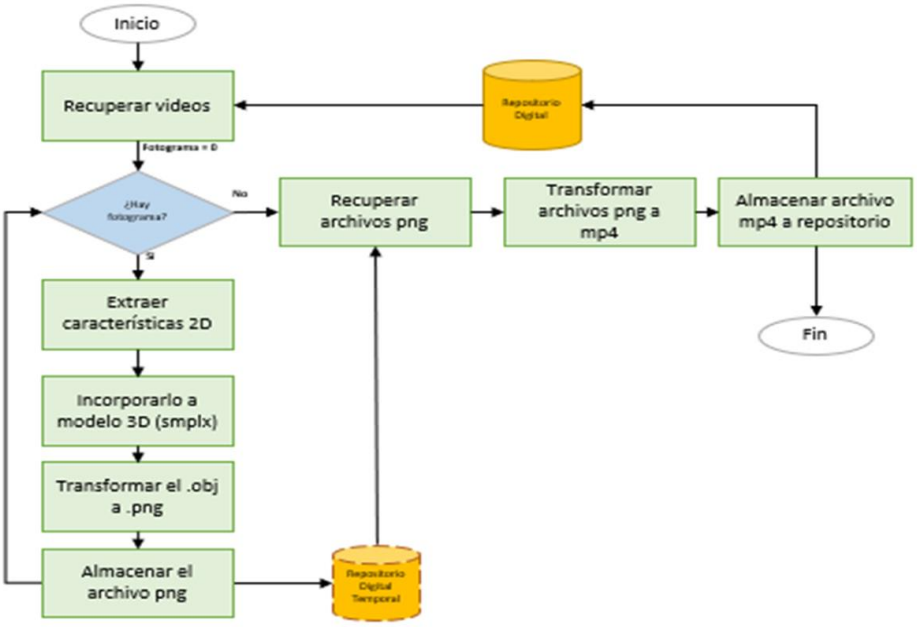

Figura 3.26 Algoritmo para la transformación 3D de la seña LSM. Fuente: Elaboración propia.

3.4.1. Obtención de las características 2D

Para la detección de los movimientos de la seña realizada por los intérpretes en cada video, es necesario leer cada uno de los *fotogramas*, para detectar la posición del cuerpo, rostro y manos del interprete. Para realizar esta tarea se necesita utilizar

un sistema con inteligencia artificial utilizando visión artificial y aprendizaje profundo. Por lo tanto, se utilizó OpenPose [45] [46] [47] [48], el cual cuenta con una red neuronal pre entrenada capaz de detectar la posición del cuerpo humano, como también los gestos faciales y la posición de las manos.

OpenPose cuenta con dos modelos diferentes para detectar el cuerpo humano: Coco (Figura 3.27) y Body_25 (Figura 3.28).

En la Figura 3.27, se puede apreciar el modelo Coco, este modelo cuenta con 18 puntos de unión (keypoints) para la detección del cuerpo humano, excluyendo los valores de las piernas y punto central de la cadera. Este último es un dato necesario para obtener mayor precisión de las coordenadas de las características 2D y de esa forma transformarlo a 3D.

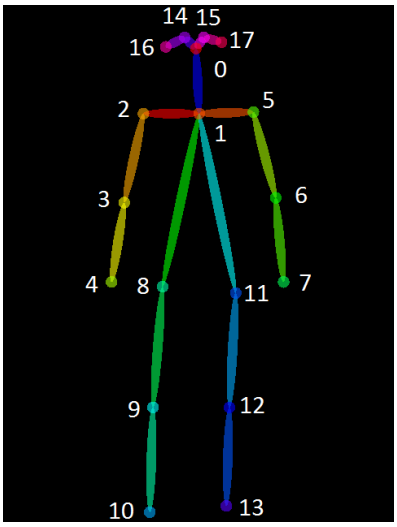

Figura 3.27 Modelo Coco de OpenPose [49]

El modelo Body_25, como se ve en la Figura 3.28 cuenta con 25 keypoints, para detectar el cuerpo humano. Este modelo cuenta con el keypoint 8 el cual da el valor central de la cadera, siendo el modelo necesario para utilizar.

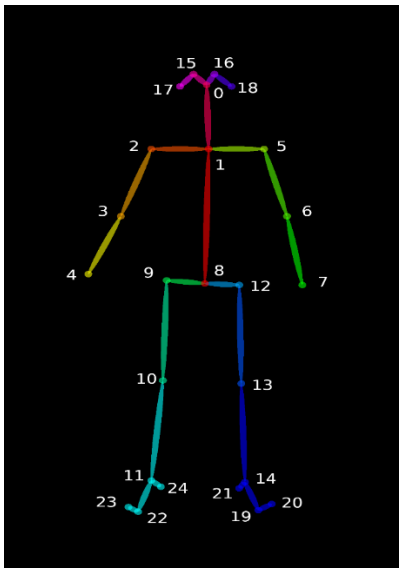

Figura 3.28 Modelo Body_25 de OpenPose [49]

Así mismo, los modelos para obtener las expresiones faciales y la posición de las manos son diferentes, y se pueden apreciar en la Figura 3.29. El modelo del rostro cuenta con 70 keypoints, mientras el modelo de la mano con 21.

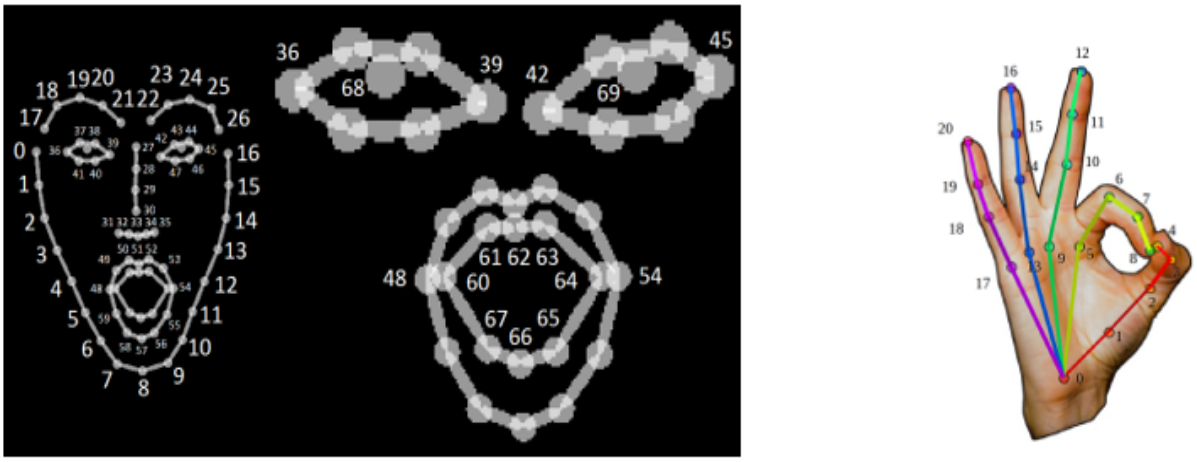

Figura 3.29 a. Modelo de detección rostro. b. Modelo de detección mano. [49]

La ventaja de OpenPose, es que estos tres modelos están relacionados entre sí, permitiendo utilizarlos para obtener una detección total. Lo anterior, permite obtener

los 3 elementos necesarios para la LSM: movimiento corporal; movimientos de la mano y dedos; y las expresiones faciales.

Al procesar los videos con OpenPose, obtenemos las coordenadas de los keypoints y estos los almacenamos en formato json dentro de una carpeta llamada *"keypoints"*. Así mismo se guardan los fotogramas resultantes del proceso (un ejemplo se puede ver en la Figura 3.30) en una carpeta llamada *"images"*.

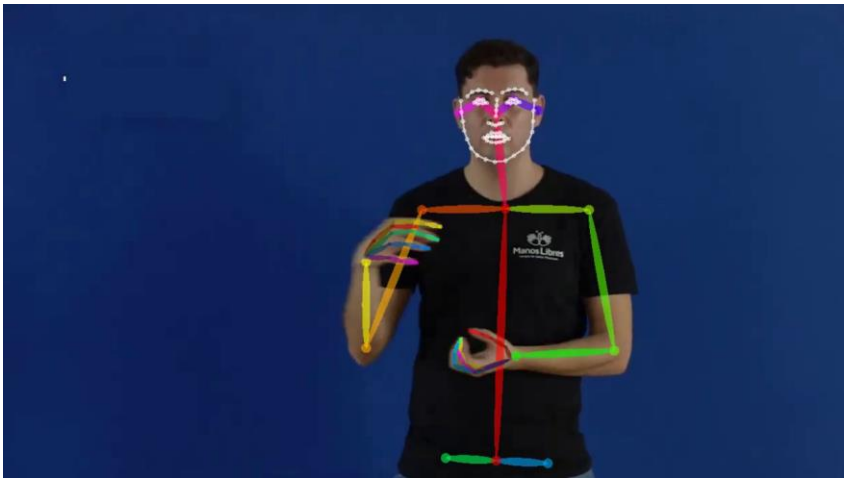

Figura 3.30 Procesamiento de un fotograma de la palabra Tarea en LSM, al procesarlo con el modelo de OpenPose.

Fuente: Elaboración propia utilizado modelo OpenPose.

Las dos carpetas mencionadas anteriormente, se almacenan dentro de una carpeta llamada "Data", ya que serán los datos de entrada para poder incorporarlos al modelo 3D.

3.4.2. Incorporarlo a modelo 3D

Al tener los keypoints de los videos almacenados, se prosigue incorporar las coordenadas a un modelo 3D. Para realizar este paso, primero se tienen que procesar esas coordenadas 2D a coordenadas 3D, esto se realiza mediante cálculos matemáticos y entrenamiento neuronal, ya que no se tiene un punto de referencia para obtener la profundidad de los videos. Para esto se utiliza el algoritmo

de smplify-x [32], el cual utiliza el modelo 3D smplx y adapto su posición corporal, facial y de las manos, basándose en una imagen, como se aprecia en la Figura 3.31.

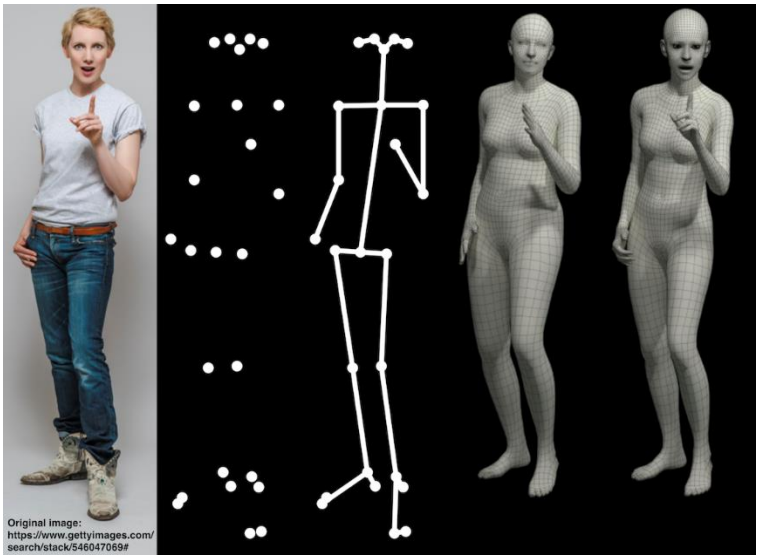

Figura 3.31 Modelo 3D Smplity-X [32].

Para el uso de este algoritmo se debe tener la imagen y los keypoints procesados por OpenPose, y estos datos deben de estar almacenados dentro de una carpeta Data, el cual es la carpeta de entrada para poder generar el avatar.

El resultado es la generación de una maya del avatar en formato obj, con su respectiva pose. Este archivo .obj lo convertimos a una imagen en formato png, con el propósito de obtener un conjunto de imágenes, de las diferentes poses del avatar y transformarlo en un video mp4.

3.5. Módulo de Recuperación de Segmentos Visuales

Teniendo el repositorio digital de segmentos visuales de la LSM creado, ya puede ser utilizado para realizar la conversión desde el texto LSM, obtenido mediante el trabajo hecho por [9], a la seña LSM.

Existen tres diferentes situaciones:

1. El segmento visual de la frase dada en texto LSM se encuentre ya almacenada en el repositorio digital Figura 3.32.
2. El texto se divide en palabras para obtener el segmento visual (Figura 3.33).
3. El segmento de una determinada palabra no se encuentra en el repositorio por lo que se divide en letras y se procede a deletrearse en LSM Figura 3.34.

En la primera situación, se sigue el diagrama de flujo mostrado en la Figura 3.32, donde se consulta la frase completa en el repositorio digital, si se encuentra en la base de datos se recupera el segmento LSM y se prosigue al módulo de generación del video de la seña LSM.

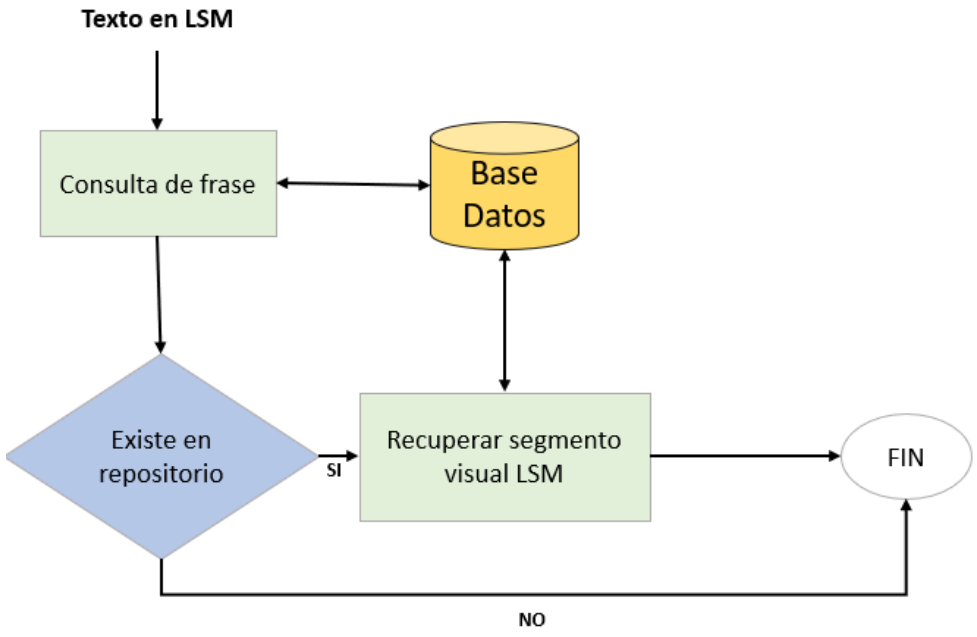

Figura 3.32 Recuperación del segmento visual LSM de la frase. Fuente: Elaboración propia

Para la situación donde la frase no se encuentra en el repositorio digital, se prosigue con seguir el diagrama de flujo mostrado en la Figura 3.33. El texto en LSM se separa en palabras y se almacena en una matriz llamada texto [] de tipo string. Se consulta las palabras en la base de datos, según su orden dentro de la matriz. Si se encuentra la palabra en el repositorio, se recupera el segmento visual y se almacena de forma temporal dentro de un objeto. Esto se repetirá hasta recuperar todos los segmentos visuales correspondientes, los cuales se encuentran en orden según la búsqueda realizada. Al terminar el ciclo se tiene como resultado las secuencias de los segmentos visuales del texto y se continúa con el módulo de generación del video.

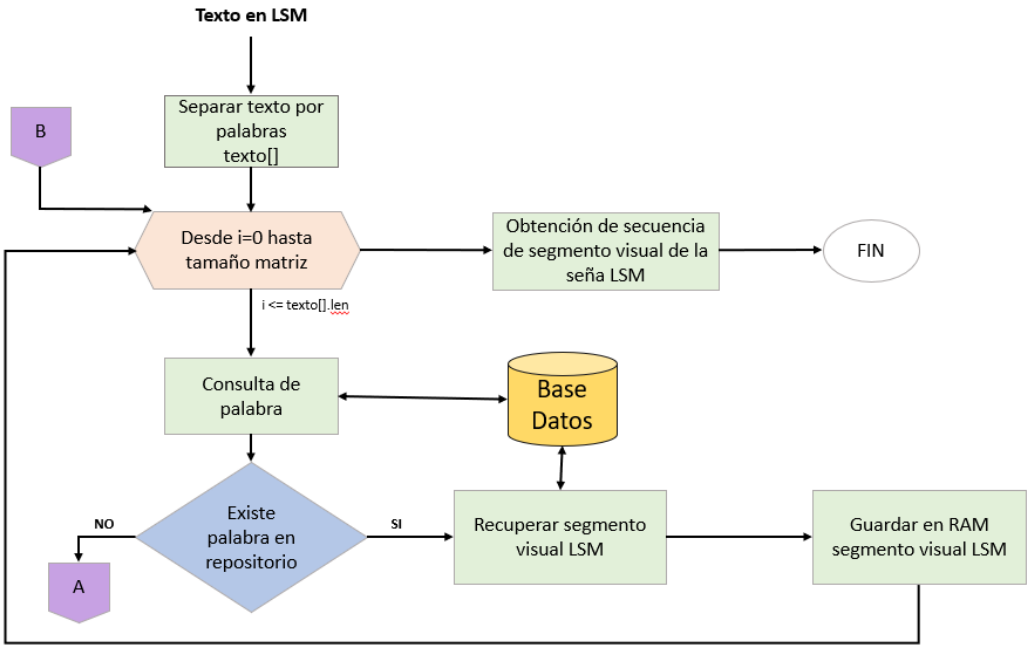

Figura 3.33 Algoritmo para la recuperación de los segmentos visuales en LSM de un determinado texto. Fuente: Elaboración propia

Para el caso de no encontrar una palabra dentro del repositorio digital, se tendrá que deletrear, siguiendo el diagrama de la Figura 3.34. Al tener la secuencia de segmentos visuales de la palabra se regresa al diagrama de la figura 3.34 y se continúa con la siguiente palabra.

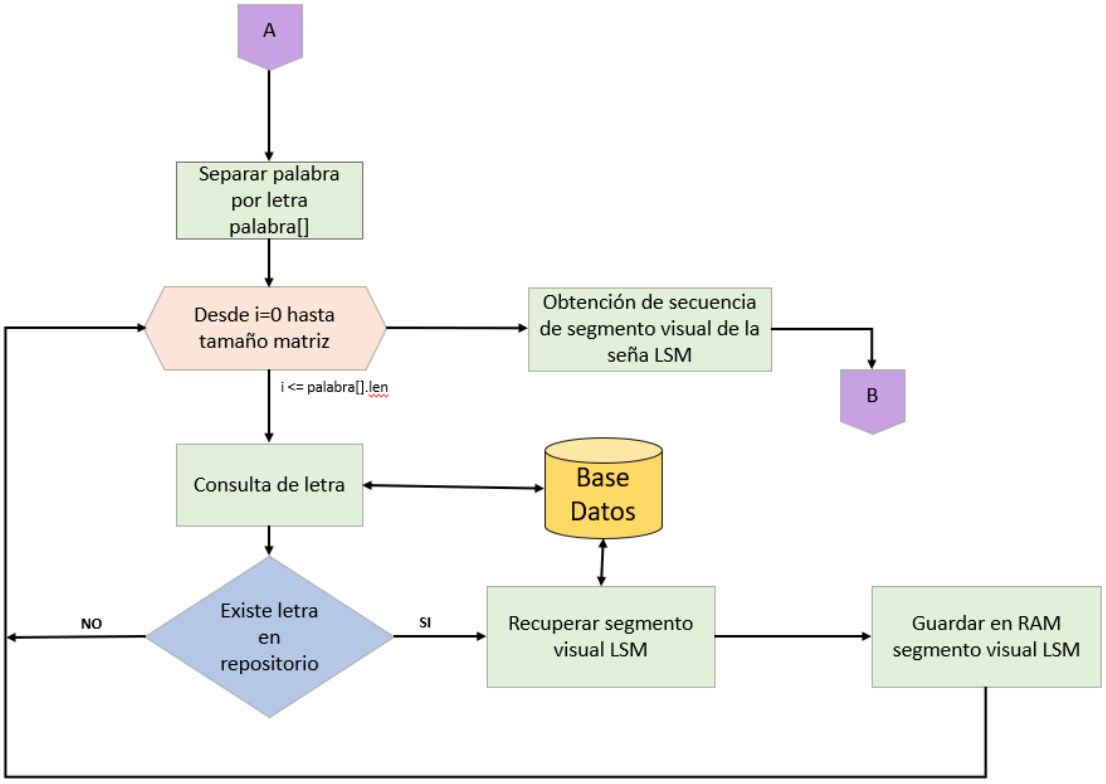

Figura 3.34 Algoritmo para deletrear las palabras. Fuente: Elaboración propia

3.6. Generación de Video de la Seña LSM

Al tener la secuencia de los segmentos visuales de las señas LSM del texto correspondiente, se genera el video de la frase en LSM. Para la realización del video se sigue el algoritmo mostrado en la Figura 3.35. Se importan los videos correspondientes y se unen según su secuencia dentro de un solo video tipo mp4.

Este video se importa a la plataforma para la educación generada por [3].

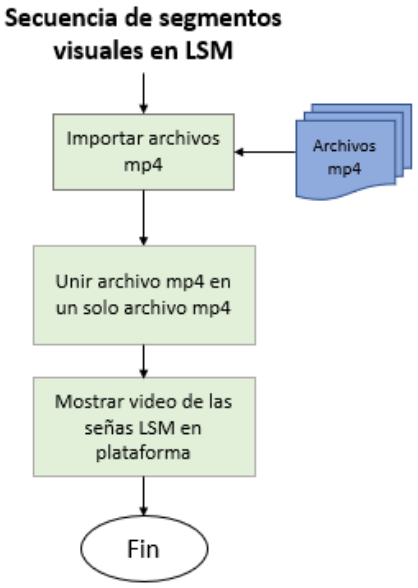

Figura 3.35 Diagrama de flujo para generar el video de las señas en LSM

CAPÍTULO 4. Implementación

4.1. Implementación de la base de datos

La base de datos se implementó en el manejador de bases de datos PostgreSQL, la base de datos sólo contiene una tabla, cuyos campos (columnas) se pueden apreciar en la Figura 4.1. El primer campo es la “palabra” correspondiente al video original de la seña LSM que se almacena en el campo “originalsenia”, finalmente se tiene el campo “avatarsenia” que almacena el video del avatar que se ha obtenido a partir del video original de la seña LSM.

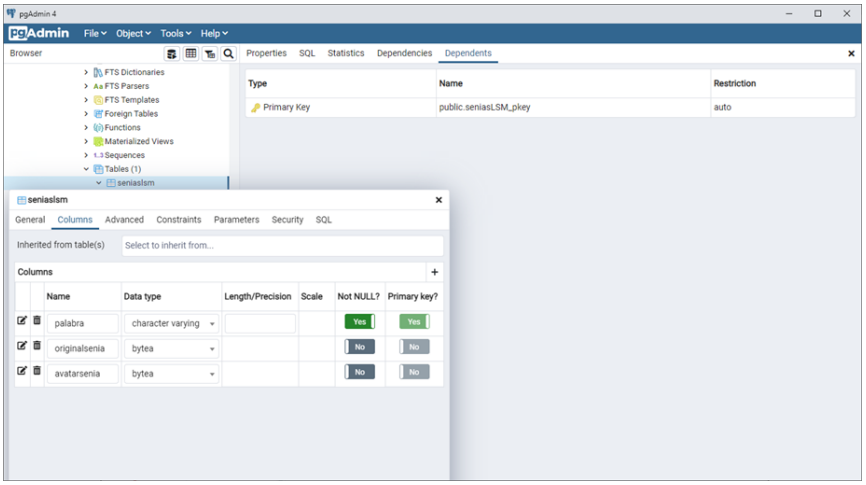

Figura 4.1 Tabla seniaslsm

En la Figura 4.2 se puede apreciar la funcionalidad para insertar en la base de datos la palabra y el archivo que contiene el video de la seña LSM correspondiente a esa palabra.

```
def almacenaBD(palabra,archivo):
    video = open(archivo, 'rb').read()

    conexion1 = psycopg2.connect(host="localhost", database="LSM", user="postgres", password="postgres", port="5432")
    cursor1=conexion1.cursor()
    try:
        cursor1.execute("insert into seniaslsm(palabra, originalsenia) values(%s,%s)", (palabra, psycopg2.Binary(video)))
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    conexion1.commit()
    conexion1.close()
```

Figura 4.2 Inserción en la tabla seniaslsm

En el caso de almacenar los demás datos correspondientes dentro de la base de datos, se sigue la misma estructura vista en la Figura 4.2, modificando la consulta a realizar y variables, la cual se realiza dentro del comando:

cursor1.execute("insert into seniaslsm (palabra, originalsenia) values(%s,%s)", (palabra, psycopg2.Binary(video)))

4.2. Implementación del módulo de extracción del video de seña LSM

La implementación de este módulo se ha desarrollado en el lenguaje Python. Para el uso de Python en un entorno gráfico amigable se instaló Anaconda Navigator quien permite el uso de entornos gráficos virtuales con las características requeridas para el desarrollador. Al instalar Anaconda Navigator se preparó el entorno virtual desde Jupyter Notebook, que nos permite manejar las librerías necesarias.

Una vez generado el entorno de desarrollo en Jupyter Notebook, se puede ingresar al mismo desde la ventana de Anaconda, en donde nos abre el entorno de desarrollo creado desde el navegador predeterminado, direccionando al servidor web local de nuestro equipo (http://localhost:8888) lo cual se muestra en la Figura 4.3.

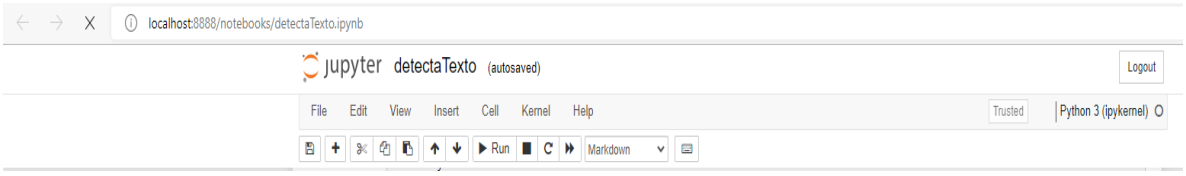

Figura 4.3 Entorno de Desarrollo Jupyter

4.2.1. Procesamiento de imágenes

Para el procesamiento de imágenes se usó la librería OpenCV, la cual contiene diversas funcionalidades para el preprocesamiento de la imagen, tales como el ajuste del tamaño de la imagen, así como el reconocimiento de rostros para considerar sólo aquellos fotogramas que contengan una persona. En la Figura 4.4 se puede apreciar en la parte superior el código para el reconocimiento de rostros y la parte inferior su resultado.

```python
# -*- coding: latin-1 -*-
"""
    Código 1.1 - Detección facial en una imagen estática
    Este programa encuentra todos los rostros de una fotografía.

    Escrito por Glare y Transductor
    www.robologs.net
"""
def deteccionRostro(imagen):

    #Cargamos nuestro classificador de Haar:
    cascada_rostro = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')
    # Si utilizas otro clasificador o lo tienes guardado en un directorio diferente al de este script python,
    # tendrás que cambiar 'haarcascade_frontalface_alt.xml' por el path a tu fichero xml.


    #Cargamos la imagen y la convertimos a grises:
    #img = cv2.imread('imagen_input.jpg')
    img_gris = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
    # Nota: La imagen de ejemplo que hemos utilizado para el tutorial ya está en blanco y negro,
    # por lo que no sería necesario convertirla. Lo he hecho igualmente por si más adelante queréis
    # probar con una imagen en color.


    #Buscamos los rostros:
    coordenadas_rostros = cascada_rostro.detectMultiScale(img_gris, 1.3, 5)
    # Nota 1: La función detectMultiScale() requiere una imagen en escala de grises. Esta es la razón
    # por la que hemos hecho la conversión de BGR a Grayscale.
    # Nota 2: '1.3' y '5' son parámetros estándar para esta función. El primero es el factor de escala ('scaleFactor'): la
    # función intentará encontrar rostros escalando la imagen varias veces, y este factor indica en cuánto se reduce la imagen
    # cada vez. El segundo parámetro se llama 'minNeighbours' e indica la calidad de las detecciones: un valor elevado
    # resulta en menos detecciones pero con más fiabilidad.

    if (len(coordenadas_rostros) >= 1):
        #Ahora recorremos el array 'coordenadas_rostros' y dibujamos los rectángulos sobre la imagen original:
        for (x,y,ancho, alto) in coordenadas_rostros:
            cv2.rectangle(imagen, (x,y), (x+ancho, y+alto), (0,0,255) , 3)


        #Abrimos una ventana con el resultado:
        #cv2.imshow('Output', imagen)
        #print("\nMostrando resultado. Pulsa cualquier tecla para salir.\n")
        #cv2.waitKey(0)
        cv2.destroyAllWindows()
        return True
    else:
        return False
```

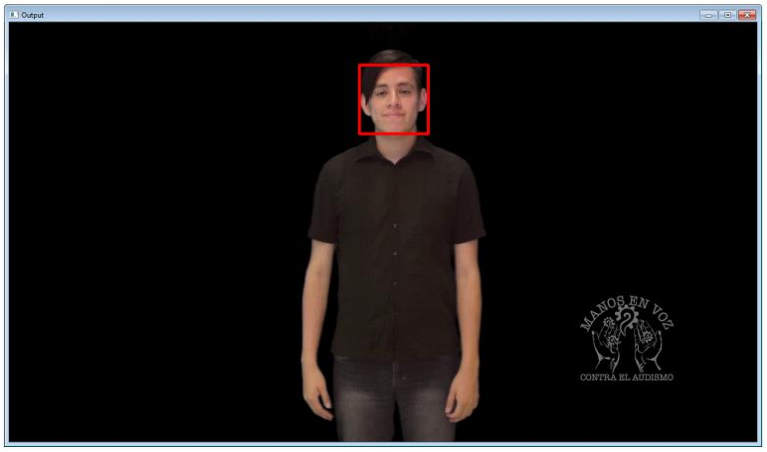

Figura 4.4 Código para el reconocimiento de rostro, con su resultado

4.2.2. Reconocimiento de los patrones de texto usando al modelo EAST

En la librería OpenCV 4, se encuentra un detector de texto en escenas basado en el aprendizaje profundo llamado EAST (**E**fficient and **A**ccurate **S**cene **T**ext Detector) que ya fue descrito en el capítulo CAPÍTULO 3. Esto permite la detección de texto

en una imagen. La implementación de esta funcionalidad se muestra en la Figura 4.5.

```
# define los dos nombres de la capa de salida para el modelo detector EAST (red neuronal profunda) que se va usar
# el primero es las probabilidades de salida
# el segundo puede ser usado para derivar las coordenadas de la caja de texto
layerNames = [
    "feature_fusion/Conv_7/Sigmoid",
    "feature_fusion/concat_3"]
```

```
# Carga el detector de texto pre-entrenado EAST
print("[INFO] cargando detector de texto EAST ...")
net = cv2.dnn.readNet("frozen_east_text_detection.pb")
# construya un blob de la imagen y ejecuta un paso hacia adelante en el modelo
# para obtener los dos conjuntos de salida
blob = cv2.dnn.blobFromImage(image, 1.0, (W, H),
    (123.68, 116.78, 103.94), swapRB=True, crop=False)
start = time.time()
net.setInput(blob)
(scores, geometry) = net.forward(layerNames)
end = time.time()
#print(scores)
#print(geometry)
# show timing information on text prediction
print("[INFO] la detección de texto uso {:.6f} seconds".format(end - start))
```

```
[INFO] cargando detector de texto EAST ...
[INFO] la detección de texto uso 0.265200 seconds
```

Figura 4.5 Reconocimiento de texto en una imagen.

4.2.3. Obtención del texto en una imagen

La obtención de texto dentro de una imagen es un proceso complejo, donde se integran diversas funcionalidades, entre las cuales se encuentran: obtener la certeza de detección del texto, verificar la existencia de falsos positivos, la determinación de la orientación del texto, esto es, si esta de manera vertical o manera horizontal, obtener las coordenadas de las áreas donde se encuentra el texto, entre otras. En la Figura 4.6 se muestra parcialmente el código correspondiente a estas funcionalidades. En este caso para la extracción de las probabilidades para calcular la certeza de que exista texto en la imagen, así como de los datos geométricos para calcular las coordenadas del área (caja) donde se encuentra el texto. Al final del código se aprecia la supresión de cajas superpuestas y delimitaciones débiles.

```
    for y in range(0, numRows):
        # extraer los scores (probabilidades), seguido por los datos geometricos
        # usados para derivar las coordenadas potenciales de las cajas que encienrran el texto
        scoresData = scores[0, 0, y]
        xData0 = geometry[0, 0, y]
        xData1 = geometry[0, 1, y]
        xData2 = geometry[0, 2, y]
        xData3 = geometry[0, 3, y]
        anglesData = geometry[0, 4, y]
        # ciclo sobre el numero de columnas
        for x in range(0, numCols):
            # si el score no tiene suficiente probabilidad (>= 0.5), lo ignora
            if scoresData[x] < 0.5:
                continue
            # calcula el valor del factor de ajuste(offset) como mapa de caracteristicas resultantes
            # y será 4x más pequeño que la imagen de entrada
            (offsetX, offsetY) = (x * 4.0, y * 4.0)
            # extrae el angulo de rotación para la predicción y
            # calcula el seno y coseno del angulo
            angle = anglesData[x]
            cos = np.cos(angle)
            sin = np.sin(angle)
            # usa el volumen geometrico para derivar el ancho y la altura de la caja que encierra el texto
            h = xData0[x] + xData2[x]
            w = xData1[x] + xData3[x]
            # calcula la coordenadas de inicio y fin (x, y) de la caja que encierra el texto
            endX = int(offsetX + (cos * xData1[x]) + (sin * xData2[x]))
            endY = int(offsetY - (sin * xData1[x]) + (cos * xData2[x]))
            startX = int(endX - w)
            startY = int(endY - h)
            # añade las coordenadas y los scores de probabilidad correspondientes a sus respectivas listas
            rects.append((startX, startY, endX, endY))
            confidences.append(scoresData[x])
    # aplica la función "non-maxima suppression" para suprimir delimitaciones débiles y cajas superpuestas
    #print(rects)
    #print(confidences)
    centro = [160,190,161,191]
    distancia = []
    boxes = non_max_suppression(np.array(rects), probs=confidences)
```

Figura 4.6 Determinación de certeza y obtención de las coordenadas geométricas.

Una vez que se ha determinado el área del texto, se obtiene una imagen de esa área, como se ilustra en la Figura 4.7, para ello se procesa la imagen con funcionalidades como el recorte, la transformación de la imagen a color a gris y su transformación a una imagen binaria.

```
crop_img = orig[startY:endY, startX:endX]

if (len(crop_img) > 0 ):
    img_gris = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)

    cv2.imwrite('SalidaH.png',img_gris)

    val = np.reshape(img_gris,-1)
    media = np.mean(val)

    if (media > 160):
        texto = 1
    else:
        texto = 0
    if (texto == 0):
        # texto en blanco
        (thresh, im_bw) = cv2.threshold(img_gris, 225, 255, cv2.THRESH_TOZERO)

        inverso = abs(255 - im_bw)
        cv2.imwrite('Salida.png',inverso)

    else:
        (thresh, im_bw) = cv2.threshold(img_gris, 128, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)

        cv2.imwrite('Salida.png',im_bw)
```

Figura 4.7 Procesamiento de la imagen que contiene el texto

Finalmente, se realiza la obtención de una cadena de caracteres que corresponde a la seña LSM a través del reconocimiento óptico de caracteres (OCR del inglés Optical Character Recognition). En la Figura 4.8 se muestra el código para realizar esta tarea, para ello se utiliza Tesseract que es un motor de reconocimiento óptico de caracteres para varios sistemas operativos. Es software libre, liberado bajo la licencia Apache, Versión 2.0. La cadena de caracteres es procesada para eliminar espacios, nuevas líneas y otros caracteres en los extremos de la cadena de caracteres. El resultado obtenido está en minúsculas.

```
        # Uso de OCR
        from PIL import Image

        img = Image.open("Salida.png")
        pytesseract.pytesseract.tesseract_cmd = "tesseract.exe"
        result = pytesseract.image_to_string(img, lang="spa")
        #print(result)
        result = result.replace("|"," ")
        result = result.replace("["," ")
        result = result.replace("-"," ")
        result = result.replace("!"," ")
        result = result.replace("¡"," ")
        result = result.replace("/"," ")
        result = result.replace("$"," ")
        #result = result.replace("¿"," ")
        result = result.strip()
        result = result.replace("\n", " ")
        result = result.lower()

        if (len(result) > 0):
            return result+'$'+str(b)+'$'+str(g)+'$'+str(r)+'$'+str(startX)+'$'+str(startY)+'$'+str(endX)+'$'+str(endY)
        else:
            return "no hay texto"
    else:
        return "no hay texto"
```

Figura 4.8 Obtención de la cadena de caracteres

4.2.4. Descarga de videos de señas LSM desde YouTube

Existen en la Web algunos videos de palabras o frases en LSM, por ejemplo, en YouTube. Asimismo, existen en la Web algunos diccionarios que contienen una lista de palabras o frases, de tal manera que al seleccionar una de ellas se redirecciona a una página Web donde se encuentra el video asociado y el cual se encuentra en YouTube. Algunas funcionalidades para recuperar estos videos de una página Web en particular, se pueden observar en la Figura 4.9 y la Figura 4.10. En la Figura 4.9 se muestra la funcionalidad para obtener las palabras o frases del diccionario, y así mismo sus respectivas ligas.

```
URL = 'https://www.wikisigns.org'
URLI = 'https://www.wikisigns.org/es/lsm/diccionario?page=1'
page = requests.get(URLI)

soup = BeautifulSoup(page.content, 'html.parser')

results = soup.find(id='container')

ligas = results.find_all('a')

for liga in ligas:
  atributos = liga.attrs
  referencia = atributos.get('href')

  cadena = liga.string # con esto se obtiene la cadena de la etiqueta

  page2 = requests.get(URL+'/'+referencia)
  soup2 = BeautifulSoup(page2.content, 'html.parser')
  results2 = soup2.find(id='container')
  ligas2 = results2.find_all('iframe')
  referencia = str(ligas2)

  elementos = referencia.split('title')
  try:
      direccion = elementos[0].split('src')
      youtube=direccion[1].replace('="','"')
      youtube = youtube.replace('"',' ')
      youtube = youtube.strip(' ')
      youtubeD = youtube.split('?')
      print(youtubeD[0])
      download_video(youtubeD[0],cadena)
  except:
      print('video no valido')
```

Figura 4.9 Funcionalidad para obtener la palabra o frase y la liga del video de la seña LSM

En el caso de la Figura 4.10 se muestra la descarga de los videos y su almacenamiento dentro de la base de datos LSM.

```
def download_video(video,filename):
    video = video.strip()
    palabra = filename
    filename = 'd:/Documentos/ITH/Tesis/videos/'+filename+'.mp4'
    formato='mp4'
    ytdl_opts = {'outtmpl': filename,'format':formato}
    with youtube_dl.YoutubeDL(ytdl_opts) as ytdl:
        ytdl.download([video])
        almacenaBD(palabra.lower(),filename)

def almacenaBD(palabra,archivo):
    video = open(archivo, 'rb').read()

    conexion1 = psycopg2.connect(host="localhost", database="LSM", user="postgres", password="****", port="5432")
    cursor1=conexion1.cursor()
    try:
        cursor1.execute("insert into senia(palabras, seniaoriginal) values(%s,%s)", (palabra, psycopg2.Binary(video)))
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    conexion1.commit()
    conexion1.close()
```

Figura 4.10 Funcionalidad para descargar y almacenar el video de la seña LSM

4.3. Implementación del módulo de Transformación en 3D

Para realizar la implementación del módulo de Transformación en 3D se utilizó el sistema operativo Ubuntu 18.04, Cuda 10, CUDNN 7.3. y Python 3.6. Así mismo, se dio uso de Google Colab Pro para utilizar OpenPose, ya que se necesita una memoria GPU mayor de 6G para el procesamiento de los videos. Así mismo se dio uso de entornos virtuales venv, con el propósito de tener un área de trabajo deseable sin presentar incompatibilidad entre librerías, para cada proceso que realizar.

Antes de comenzar con esta implementación se extrajeron los datos almacenados en la base de datos LSM dentro de una carpeta con el nombre videos para una fácil manipulación e implementarlo al modelo 3D, utilizando el siguiente comando:

cursor1.execute(""" SELECT palabras, seniaoriginal FROM senia""")

4.3.1. *Obtención de características 2D*

Para la obtención de los keypoints, se utilizaron 3 modelos pre entrenados de *machine learning* utilizando visión computacional para la detectar los movimientos de los intérpretes dentro de los videos en LSM. Los modelos utilizados pertenecen a OpenPose [45] [46] [47] [48], con el modelo body_25, rostro y manos. Ya que este proceso consume una gran cantidad de memoria GPU se utilizó Google Colab Pro, permitiendo generar un entorno virtual apto para correr el programa. Fue necesario obtener la versión de paga de Google Colab pro, por la necesidad de mayor tiempo de ejecución, memoria de almacenamiento y memoria GPU.

Para la obtener los Keypoints de los videos, se utilizó OpenPose demo dentro de colab. Se comenzó instalando las dependencias y OpenPose demo, utilizando el código de la Figura 4.11.

```
import os
from os.path import exists, join, basename, splitext

git_repo_url = 'https://github.com/CMU-Perceptual-Computing-Lab/openpose.git'
project_name = splitext(basename(git_repo_url))[0]
if not exists(project_name):
  # see: https://github.com/CMU-Perceptual-Computing-Lab/openpose/issues/949
  # install new CMake becaue of CUDA10
  !wget -q https://cmake.org/files/v3.13/cmake-3.13.0-Linux-x86_64.tar.gz
  !tar xfz cmake-3.13.0-Linux-x86_64.tar.gz --strip-components=1 -C /usr/local
  # clone openpose
  !git clone -q --depth 1 $git_repo_url
  !sed -i 's/execute_process(COMMAND git checkout master WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}\/3rdparty\/caffe)/execute_process(COMMAND git checkout f019d0dfe86f49d1140961f
  # install system dependencies
  !apt-get -qq install -y libatlas-base-dev libprotobuf-dev libleveldb-dev libsnappy-dev libhdf5-serial-dev protobuf-compiler libgflags-dev libgoogle-glog-dev liblmdb-dev op

  # build openpose
  !cd openpose && rm -rf build || true && mkdir build && cd build && cmake .. && make -j`nproc`
```

Figura 4.11 Instalación de OpenPose Demo.

Teniendo instalado OpenPose se procede a la importación de los videos de la base de datos dentro de Google drive y se vincula Google drive con Google Colab. Se obtiene la dirección de los videos y estos se procesan con OpenPose demo.

Al ser Google Colab un entorno virtual, este se desconecta y se pierde la información generada, por lo tanto, los resultados generados son guardados dentro de Google Drive con el siguiente formato:

Palabra:

- → Images
  - → Palabra_0000000000.png
  - → Palabra_0000000001.png
  - → Palabra_0000000002.png

…

- → Keypoint
  - → Palabra_0000000000_keypoints.json
  - → Palabra_0000000001_ keypoints.json
  - → Palabra_0000000002_ keypoints.json

…

Al generar una cantidad de datos inmensa, ya que se tienen que procesar 2455 videos y estos datos se tienen que exportar para almacenarlos dentro de la computadora y después poder generar el avatar, se guardaron dentro de subcarpetas (68 palabras en LSM por carpetas), siguiendo el siguiente formato:

Output:

- → Ouput_1/
  - → Palabra_1/
  - → Palabra_2/
  - → Palabra_3/

…

  - → Palabra_68/

- → Ouput_2/
  - → Palabra_69/
  - → Palabra_70/
  - → Palabra_71/

…

  - → Palabra_136/

…

  - → Output_37/

Teniendo un total de 37 carpetas. De esta forma cada una de las subcarpetas (Output_n) se exportan a la computadora, sin perder ningún dato.

El proceso realizado para obtener los videos, procesarlos y guardar los resultados en sus respectivas carpetas se puede apreciar en la Figura 4.12.

```
path_dir_original_videos = '/content/drive/MyDrive/tesis_openpose/videos_pare'
number_dir = 1 #68 archivos por directorio
cont_video = 0 #contar del 0 al 612
path_dir_output=create_new_dir(number_dir=number_dir)
list_videos = list_direct(path_dir_original_videos)
for video in list_videos:
  print(video)
  #si se exceden de los 68 videos
  if (cont_video >= 68):
      cont_video=0
      number_dir=number_dir+1
      path_dir_output=create_new_dir(number_dir=number_dir)

  name_video=re.sub(".mp4","",video)
  path_dir=path_dir_output+name_video
  path_dir_output_key=path_dir+'/keypoints/'
  path_dir_video_key=path_dir+'/videos/'
  try:
    os.makedirs(path_dir_output_key)
    os.makedirs(path_dir_video_key)
  except FileExistsError:
    # directory already exists
    pass
  videos_a =path_dir_original_videos+'/'+video
  do_keypoint(videos_a=videos_a, path_dir_output_key=path_dir_output_key, path_dir_video_key=path_dir_video_key)
  extract_frames(path_dir,path_dir_video_key,name_video)
  print('video '+str(cont_video)+' del directorio '+str(number_dir)+' listo!')
  cont_video=cont_video+1
```

Figura 4.12 Proceso para procesar los videos en LSM y almacenarlo en sus respectivas Carpetas.

Así mismo, la función para poder generar los *keypoints* con OpenPose, es el que se muestra en Figura 4.13.

```
def do_keypoint(videos_a, path_dir_output_key, path_dir_video_key):
  print(videos_a)
  #file to make the video mp4
  path_dir_video_key_mp4 = path_dir_video_key +'output.mp4'
  !rm openpose.avi
  !cd openpose && ./build/examples/openpose/openpose.bin --video $videos_a --hand --face --keypoint_scale 1 --write_json $path_dir_output_key --display 0 --write_video ../openpose.avi
  # convert the result into MP4
  !ffmpeg -y -loglevel info -i openpose.avi $path_dir_video_key_mp4
```

Figura 4.13 Generar los keypoints de los videos LSM

4.3.2. Incorporación al modelo 3D

Teniendo los keypoints definidos de cada fotograma de cada video, se prosigue con el modelado 3D. Para esto se utiliza el modelo SMPLX [32], el cual es un modelo 3D real del cuerpo humano, capaz de modificar las expresiones faciales, los movimientos de las manos y dedos, dando movimientos parecidos a la realidad. Para generar el modelo se utilizó el algoritmo smplify-x [32], el cual genera el modelo smplx mediante una imagen 3D. Para dar uso de este algoritmo, es necesario utilizar el sistema operativo Linux, en este caso se utilizó Ubuntu 18.04.

Asimismo, es necesario el uso de diferentes dependencias: Pytorch, es una librería de aprendizaje profundo para aplicaciones que utilizan visión artificial; Vpose [32], es un sistema entrenado para obtener una variación de posturas humanas, se utiliza con cinemática inversa para resolver diferentes tareas como obtener la pose de un humano dentro de una imagen [50]; y finalmente Homogenus, el cual es utilizado para obtener el género de un personaje dentro de una imagen, este no es utilizado por este trabajo de investigación, puesto que el avatar deseado no debe de cambiar su forma física, sin embargo es necesario su instalación para poder correr smplify-x.

Teniendo el área de trabajo preparado para poder utilizar smplify-x, se prosigue con generar el modelo 3D. Para esto, es necesario almacenar los resultados obtenidos de Openpose dentro de una carpeta con el nombre Data. La función para poder generar el modelo da cada fotograma se puede ver en la Figura 4.14.

```
def generate_smplifyx(input_path, output_path):
    try:
        pipe=subprocess.call("python3 smplifyx/main.py --config cfg_files/fit_smplx.yaml --data_folder "+ input_path +" --output_folder " +
                             output_path+ " --visualize="+ "False" +" --model_folder "+"MODEL_FOLDER"+" --vposer_ckpt "+ "VPOSER_FOLDER" +
                             " --part_segm_fn "+ "smplx_parts_segm.pkl"+"--genere MALE",shell=True)
        print("Ejecución con exito "+input_path)
    except:
        print("NO SE PUDO REALIZAR LA TRANSFORMACIÓN DE LOS DATOS: "+ input_path)


if __name__ == '__main__':
    root_input = './Data'
    list_name_file=file_list(root_input)
    for file_name in list_name_file:
        input_path=root_input+'/'+file_name
        output_path=create_dir(file_name=file_name)
        generate_smplifyx(input_path=input_path, output_path=output_path)
```

Figura 4.14 Generar modelo 3D smplx utilizando smplify-x.

En la Figura 4.15, se puede apreciar el resultado del algoritmo. En la imagen superior se muestra un fotograma de la palabra “Tarea”, mientras en la imagen inferior se encuentra el modelo 3D generado por el smplify-x, basándose en la imagen superior.

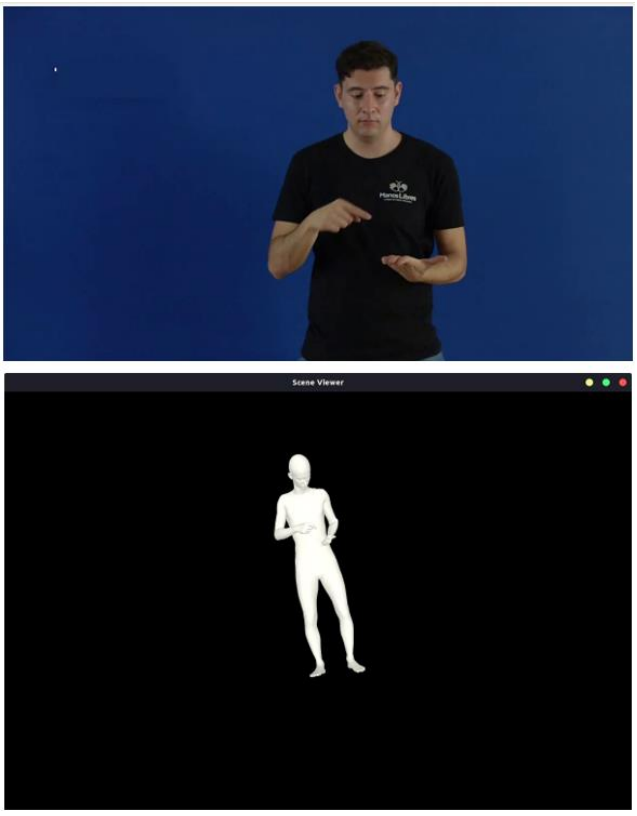

Figura 4.15 Modelo 3D utilizando smplify-x

El resultado obtenido es un archivo obj por lo tanto, se necesita convertirlo a imagen tipo png. Para eso se utilizó el siguiente código obj2png, el cual se encuentra en [51], el cual permite transformar un archivo obj a png. Para correr el programa, se utiliza el siguiente código:

```
python3 obj2png.py -i fotograma_palabra.obj -a -95 -e 100
```

El resultado obtenido de convertir el modelo 3D a png, se muestra en la Figura 4.16.

Figura 4.16 Modelo 3D transformado a png

Al generarse el conjunto de obj de un video en LSM a png, estas se unen para formar el video del avatar diciendo la palabra en LSM en formato mp4. La función utilizada para generar el video esta ilustrada en la Figura 4.17.

```
def generate_video():
    image_folder = '/home/luisa/Documents/Tesis/Codigo/AreaTransformacion3D/SMPLIFY-X/V3/object2mp4/obj2png/src/imagenes'
    video_name = 'mygeneratedvideo.avi'
    os.chdir('/home/luisa/Documents/Tesis/Codigo/AreaTransformacion3D/SMPLIFY-X/V3/object2mp4/obj2png/src/imagenes')

    images = [img for img in os.listdir(image_folder)
              if img.endswith(".jpg") or
                 img.endswith(".jpeg") or
                 img.endswith("png")]


    print(images)

    frame = cv2.imread(os.path.join(image_folder, images[0]))


    height, width, layers = frame.shape

    video = cv2.VideoWriter(video_name, 0, 1,(width, height))


    for image in images:
        video.write(cv2.imread(os.path.join(image_folder, image)))


    cv2.destroyAllWindows()
    video.release()

generate_video()
```

Figura 4.17 Función para generar el video de la palabra en LSM

Finalmente, el video generado se almacena dentro de la base de datos de PostgreSQL, teniendo de esta forma el repositorio digital de segmentos visuales definido.

4.4. Implementación del módulo de Recuperación de Segmentos Visuales

Al tener almacenado en el repositorio digital el segmento visual de las palabras o frases en LSM, se prosigue al módulo de Recuperación de segmentos visuales. Para este módulo el propósito es lograr obtener la secuencia de los segmentos visuales respetando la gramática de la lengua de señas mexicanas. Para este módulo se dio uso de Python 3.9.7 e IDLE como editor de Texto.

Obtenemos la frase con la gramática de la lengua de señas mexicanas y se realiza la consulta. Comenzamos revisando si la frase se encuentra dentro del repositorio, realizamos la consulta como se muestra en la Figura 4.18. Si la consulta devuelve

un valor, el segmento visual será almacenado dentro de una carpeta, para después ser utilizado.

```
# Este programa recupera de la base de datos el video de la palabra asociada
# El resultado se almacena como un archivo mp4
#palabra='zapato'
conexion1 = psycopg2.connect(host="localhost", database="LSM", user="postgres", password="****", port="5432")
cursor1=conexion1.cursor()

cursor1.execute(""" SELECT avatarsenia
                    FROM public.senialsm
                    WHERE palabra like 'hola como estas';
                    """)
```

Figura 4.18 código para obtener una frase

En el caso de que se devuelva un valor nulo se proseguirá con descomponer la cadena de caracteres, a una lista de palabras utilizando el comando palabras=frase.split(), donde *palabras* es la variable matriz que contiene las palabras dentro de la frase, y frase es la variable de la frase a traducir en lsm. Después se prosigue a realizar la consulta de cada palabra como se observa en la Figura 4.18, y se almacena el segmento visual obtenido dentro de la carpeta definida. El archivo obtenido se modifica su nombre por el número de orden en el que debe de visualizarse.

```
for index in palabras:
    cursor1.execute("SELECT avatarsenia FROM public.senialsm WHERE palabra like '%s", index )
```

Figura 4.19 Consultar en la base de datos una palabra

Sin embargo, la palabra no se encuentra dentro del repositorio digital, se prosigue a descomponer la palabra a letras y dando a consecuencia a deletrear la palabra, repitiendo el procedimiento anterior.

4.5. Implementación para generar el Video de la Seña LSM.

Teniendo los segmentos correspondientes para poder visualización la frase/palabra deseada en LSM, se prosigue a unir los videos de una forma consecutivas siguiendo la secuencia correspondiente. Para esto, se utiliza Python y la librería moviepy, el cual permite realizar edición a videos como unir, rotar, cortar, etc. En la se encuentra el código utilizado para generar el video con la unión de los demás segmentos visuales.

```
from moviepy.editor import *
def unir_videos(videos, path):
    shape = np.size(videos)
    a=np.empty(shape, dtype=str)
    cont=0
    for video in videos:
        clip=VideoFileClip(video)
        a[cont]=clip
        cont+=1
    final_clip = concatenate_videoclips(a)
    final_clip.write_videofile(str(path) + "resultado.mp4")
```

Figura 4.20 Función para generar el video de la seña en LSM

CAPÍTULO 5. Pruebas y análisis de resultados

5.1. Extracción de secuencia de fotogramas y su correspondiente texto de videos de LSM

Los videos LSM obtenidos de diversas fuentes tales como YouTube, diccionarios en la Web y el diccionario DIELSEME nos ha permitido crear un corpus de 2455 videos de señas LSM que se encuentran almacenados en el repositorio. Estos videos de señas LSM corresponden a palabras o frases en español. Para la obtención de estos videos LSM se usaron varios videos de prueba para la extracción de la secuencia de fotogramas y su correspondiente texto y validar la funcionalidad. En la Figura 5.1 se muestra un primer video el cual esté compuesto de varias señas. Cómo se puede apreciar, se han marcado los videos extraídos como extracción exitosa (palomita verde) o fallidas (cruz roja), permitiendo ajustar el algoritmo.

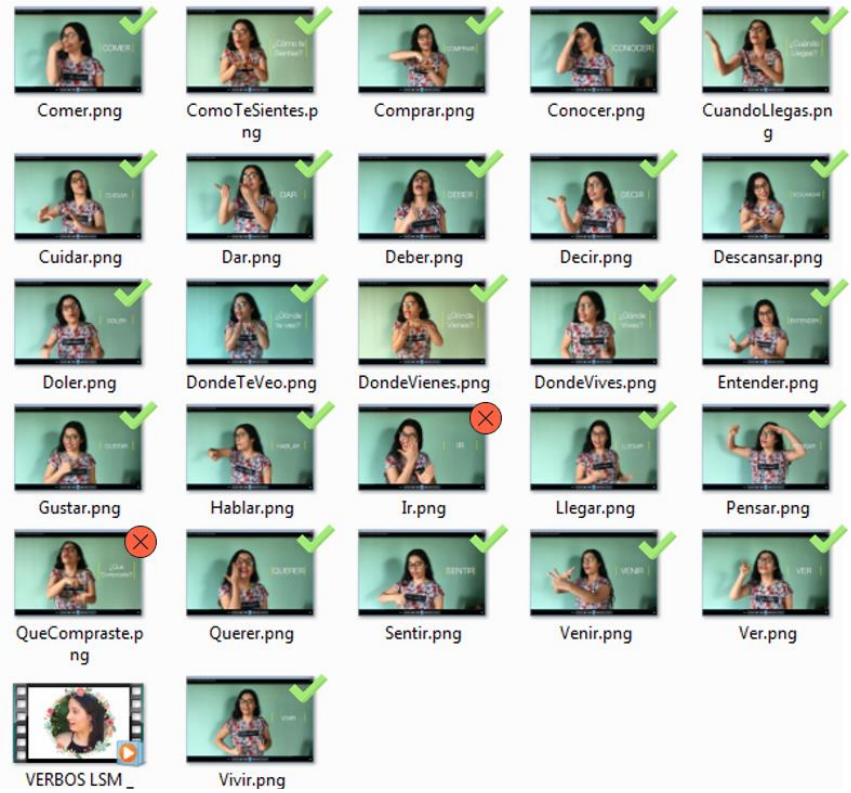

Figura 5.1 Resultado de video para la validación del algoritmo de extracción.

Como se puede apreciar en la Figura 5.1, los archivos mostrados corresponden a un fotograma del video, los nombres de los archivos indican la palabra o frase correspondiente a la seña LSM. Por ejemplo, podemos ver que el video correspondiente al verbo "*Ir"* o la frase "*Que compraste"* son incorrectos, en algunos casos se detectó que la falla era porque se tiene el brazo encima donde está la palabra y se interrumpía la secuencia de fotogramas. En otros videos se encontró que el contraste en los fotogramas no era el adecuado, lo cual interrumpe la generación de la secuencia de fotogramas para conformar el video de la seña LSM.

En la Figura 5.2 se muestran los videos extraídos del archivo denominado "*ConceptosEscuela*", se extrajeron 64 videos LSM, los nombres de archivo indican la palabra o frase correspondiente a la seña LSM.

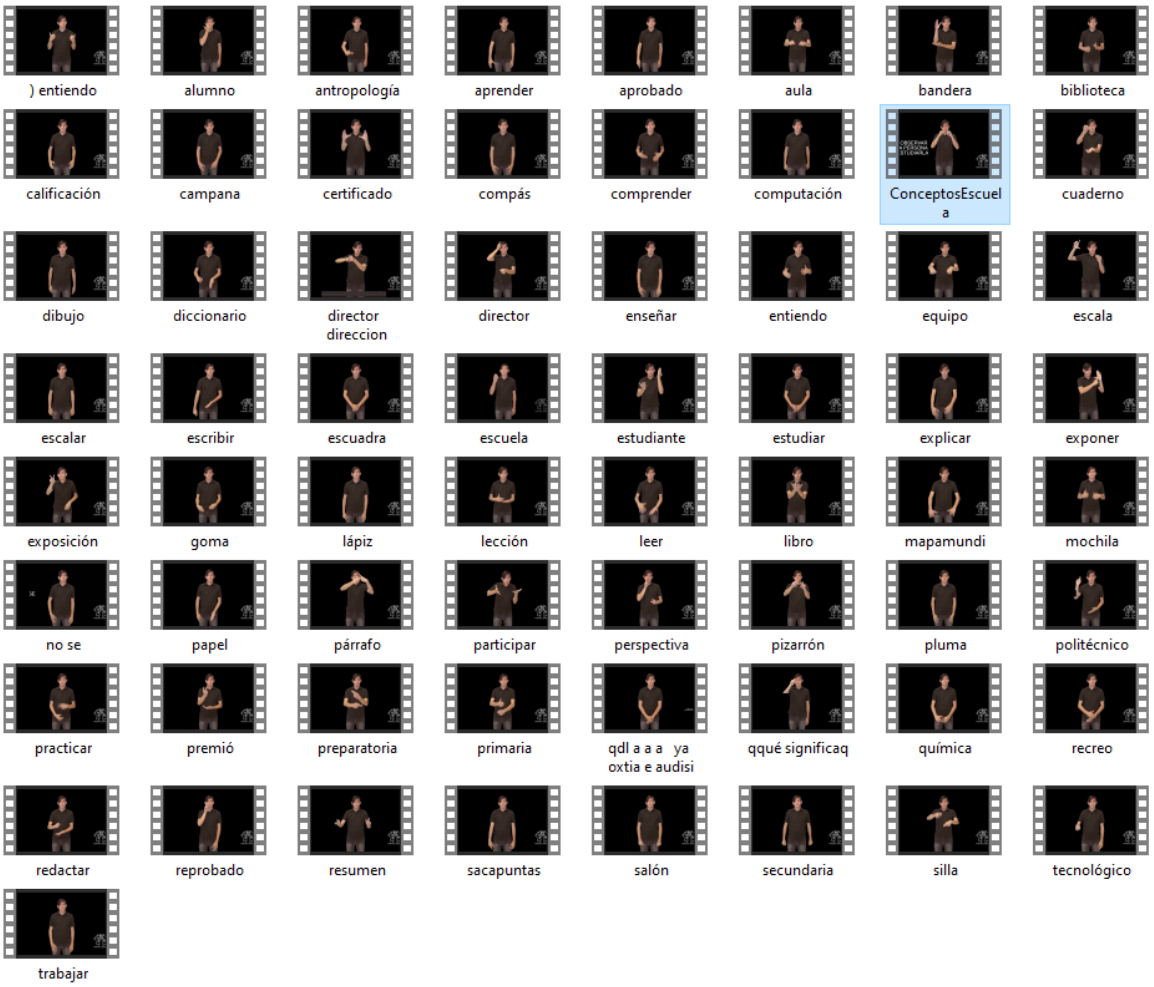

Figura 5.2 Videos de señas LSM extraídos del video conceptoEscuela

Como se puede apreciar en la Figura 5.2, existen tres videos que tienen incorrecta la palabra o frase, el que tiene como nombre del archivo "*qdl a a a ya oxtia e audisi*" no fue almacenado en la base de datos ya que la cantidad de fotogramas es muy pequeño, lo que indica que no corresponde a un seña LSM.

Es importante señalar que cuando se procesaba un nuevo video, el cual tenía mejores propiedades, en el sentido de contraste, oclusión del texto, entre otras, se reemplazaba el video LSM en la base de datos, lo cual nos permite asegurar que en su mayoría son correctos de acuerdo con una verificación aleatoria. Sin embargo,

también es importante indicar que se requiere de una revisión del corpus de videos LSM para asegurar un conjunto de videos de calidad.

5.2. Generación del modelo 3D con respecto a videos almacenados en el repositorio digital.

Para generar los modelos 3D respecto a los videos almacenados en el repositorio, se comienza pasando por el proceso de OpenPose, el cual se logró realizar el mapeo de los 2455 videos almacenados en la base de datos, en un tiempo estimado de 20 horas en total, donde se almacenaron los *fotogramas* de cada uno de los videos procesados en png, los *keypoints* de los videos en json y el video procesado en mp4.

Los videos de las señas LSM, se graban a partir del torso hacia arriba, lo cual las partes inferiores como las piernas, no se logran mapear, como se aprecia en la Figura 5.3.

Figura 5.3 Mapeo del video apagar. Fuente: elaboración propia con video público de la web y uso de Openpose.

Así mismo, estos videos al ser obtenidos por la web no cuentan con un escenario normalizado, provocando fallas en la lectura de algunos *fotogramas* de los videos

para detectar correctamente la posición de algún miembro del cuerpo. Un ejemplo de lo anterior se puede apreciar en la Figura 5.4, el cual es el fotograma número 17 del video de la palabra apagar y es la continuación del fotograma mostrado en la Figura 5.3. Como se puede apreciar se muestra una falla para detectar la posición de los dedos de la mano derecha de la persona, a pesar de haberlo detectado correctamente en el fotograma anterior (Figura 5.3). Los puntos al no ser detectados se almacenan dentro del archivo json con valor 0.

Figura 5.4 Falla en la dirección de la mano derecha del video apagar. Fuente: elaboración propia con video público de la web y uso de OpenPose.

Para la conversión de los fotogramas procesados a los modelos 3D, se utiliza smplify-x, este proceso demora alrededor de 45 minutos para un video con 33 fotogramas y con una computadora de tarjeta gráfica de GTX 1060 6GB, el cual es un proceso el cual consume una cantidad alta de tiempo. Así mismo, se consideran videos con un mayor número de fotogramas superando los 200 fotogramas. Debido a la previamente descrito, se utilizó una cantidad de 20 videos para poder realizar las pruebas de este trabajo. Los videos seleccionados fueron videos aleatorios, algunos de los resultados se pueden apreciar en la Figura 5.5, donde en la columna de la izquierda se aprecian los fotogramas sin procesar del video, en la columna central se encuentran los fotogramas procesados por OpenPose y en la columna de la derecha se encuentran los modelos 3D generados con smplify-x.

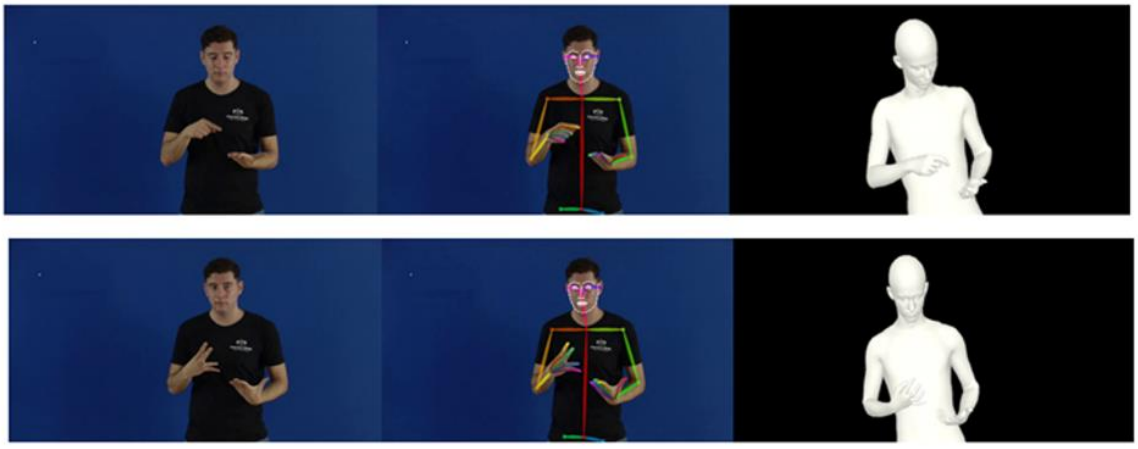

Figura 5.5 Generación de modelo 3D a partir de un video. Fuente: elaboración propia utilizando video público de la web, openpose y smplify-x

En algunos fotogramas, como se mencionó anteriormente, por la falta de normalización del escenario de los videos da como consecuencia lecturas fallidas de los keypoints correctamente, provocando una generación fallida del modelo 3D. Una de las fallas presentadas se puede apreciar en la Figura 5.6, donde se generan dos modelos 3D a partir de un solo fotograma, esto se debe a que se generó otro modelo body-25 dentro de una misma imagen, esto se debe porque openpose interpretó la imagen como si hubiera más de una sola persona.

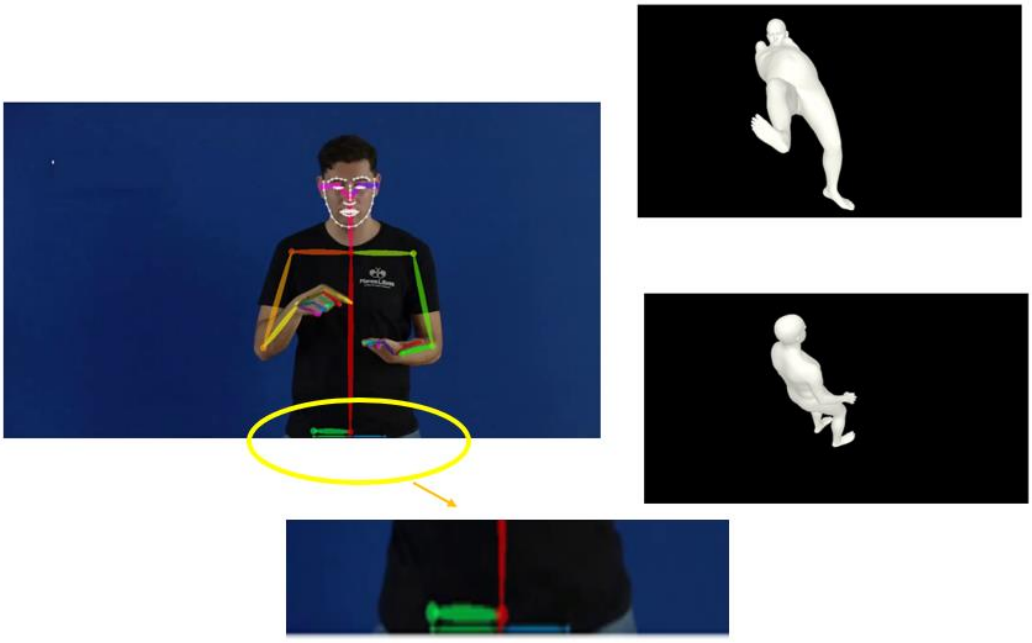

Figura 5.6 Generación doble de Keypoints. Fuente: elaboración propia utilizando videos públicos de la web, openpose y smplify-x

La solución para la situación anterior es combinar los valores de los keypoints generados por los dos modelos de openpose, sirviendo como complementos a datos faltantes, teniendo un solo archivo json. En este caso en el modelo 1 le hacía falta el keypoint 12 (ver Figura 3.28), el cual el modelo 2 tenía y se pudo generar el modelo correctamente como se puede apreciar en la primera fila de imágenes en la Figura 5.5.

Otra situación dada, el cual provocaba fallos al generar el modelo 3D, es la falta de los keypoints 8, 9 y 12. Esta falta de lectura de estos keypoints, se debe a que las grabaciones se realizan del torso hacia arriba y en ocasiones el final del torso no se logra apreciar. En la parte superior izquierda de la Figura 5.7 se tiene un fotograma en la cual falta el keypoint 9 y 12 dando como consecuencia al modelo 3D de la parte superior derecha. Mientras en la parte inferior izquierda hace falta los keypoints 8,9 y 12 dando como consecuencia el modelo 3D de la parte inferior derecha. Estos dos modelos 3D resultantes, son modelos fallidos, los cuales nos

dan a entender que es necesario tener los keypoints 8,9 y 12 para una generación correcta del modelo 3D.

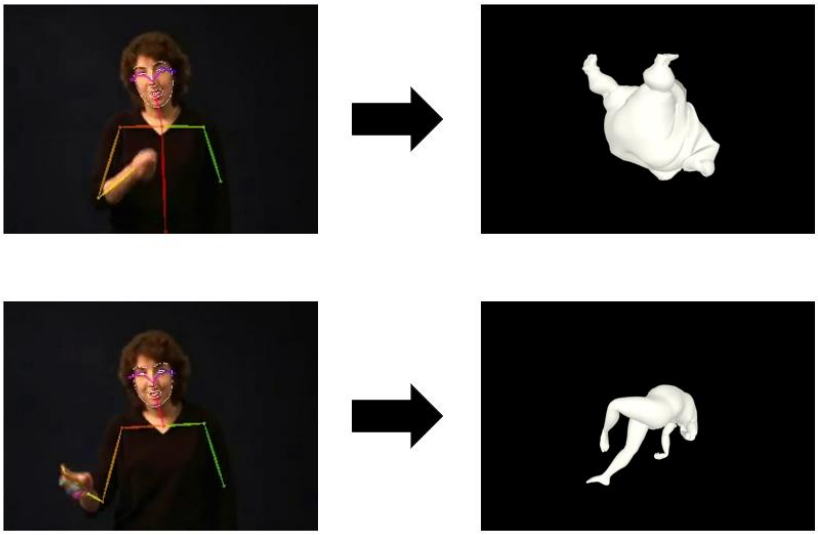

Figura 5.7 Falla de detección de la parte inferior del torso. Fuente: elaboración propia utilizando videos públicos de la web, openpose y smplify-x

Para solucionar los problemas como los encontrados en la Figura 5.7, donde faltan los valores de los keypoints 8,9 y 12, se utilizaron fórmulas para obtener estos valores. Basándonos en las relaciones de proporciones de un cuerpo perfecto [52] que existen entre los diferentes miembros, utilizado por artistas al momento de crear obras del cuerpo humano [53]. Para esto, las medidas para obtener las proporciones del cuerpo se realizan con el tamaño de la cabeza. Se tiene que el ancho entre el hombro derecho e izquierdo es proporcional a 3 cabezas, y el ancho de la pelvis (keypoint 9 al 12) es de 2 cabezas. Asimismo, la pelvis (keypoint 8) se encuentra a $4\frac{1}{2}$ cabezas. Utilizando las condiciones mencionadas, se modifica el archivo json y se logra obtener los resultados mostrados en la Figura 5.8, mostrando una mejoría del modelo 3D comparándolos a los resultados obtenidos en la Figura 5.7.

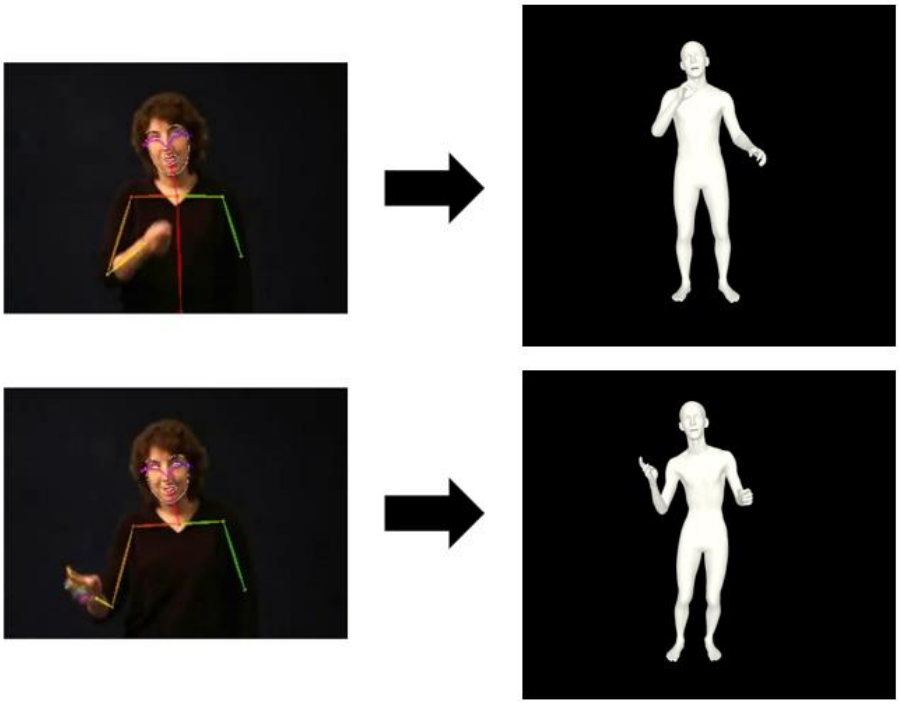

Figura 5.8 Solución de keypoints faltantes. Fuente: elaboración propia utilizando video público de la web, openpose y smplify-x

Teniendo los modelos 3d generados, continuamos con transformar los resultados a imágenes y de esta forma crear el video en formato mp4 de cada una de las señas LSM con el modelo 3D.

Para poder transformar el modelo que se encuentra en formato obj, se necesita definir la orientación en la que queremos que el modelo 3D se encuentre y de esta forma tener la imagen plasmada como deseamos. En este caso se les dio un azimut de -95 y una elevación de 100. En la Figura 5.9 se muestra algunos de los videos resultantes de las señas LSM, donde se logra ver que la orientación asignada es la correcta, a excepción de los videos marcados con “x” los cuales se necesitan afinar su orientación.

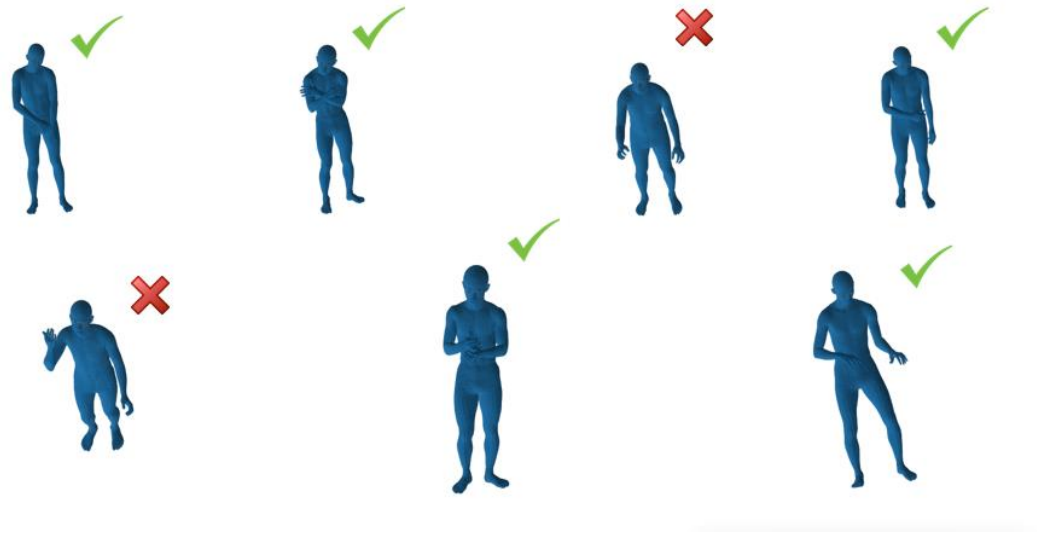

Figura 5.9 Videos de las señas LSM. Fuente: Elaboración propia.

5.3. Recuperación de los segmentos visuales y generación del video LSM

Para finalizar, se continuó generando diferentes frases del español a la lengua de Señas mexicanas, con el propósito de probar como se recuperaban los segmentos visuales y se generaba el video final de la seña LSM.

La primera frase utilizada fue “El alumno comprende la computación”, la cual es una frase que no se encuentra en el repositorio digital por lo tanto se tienen que recuperar los segmentos visuales de cada una de las palabras. Como resultado, se generó el video cuyas imágenes son mostradas en la Figura 5.10. En el primer fotograma mostrado en el lado izquierdo de la figura, se muestra el avatar realizando la seña de “computación”, seguido del fotograma del avatar realizando la seña de “alumno” y finalmente se muestra otro fotograma del avatar realizando la seña de “comprender”.

Figura 5.10 Imágenes del video de la seña LSM de la frase "El alumno comprende la computación". Fuente: elaboración propia

Lo anterior muestra, que se está recuperando los segmentos visuales correspondientes, siguiendo el orden de la gramática de la LSM en donde primero se tiene el objeto (computación), seguido del sujeto(estudiante) y finalmente se tiene el verbo (comprender).

Otra frase que se utilizó para realizar las pruebas fue “La anciana abraza a su hija”, teniendo como resultado un video con imágenes como la que se muestra en la Figura 5.11, el cual también sigue la gramática de la LSM.

Figura 5.11 Imagen del video de la frase "La anciana abraza a su hija". Fuente: Elaboración propia

La siguiente frase, es una frase ya establecida en el repositorio digital, ésta se realizó con el propósito de probar si se lograba obtener los segmentos de frases ya almacenados. La frase utilizada fue “Estoy aprendiendo LSM” y el resultado obtenido se muestra en la Figura 5.12.

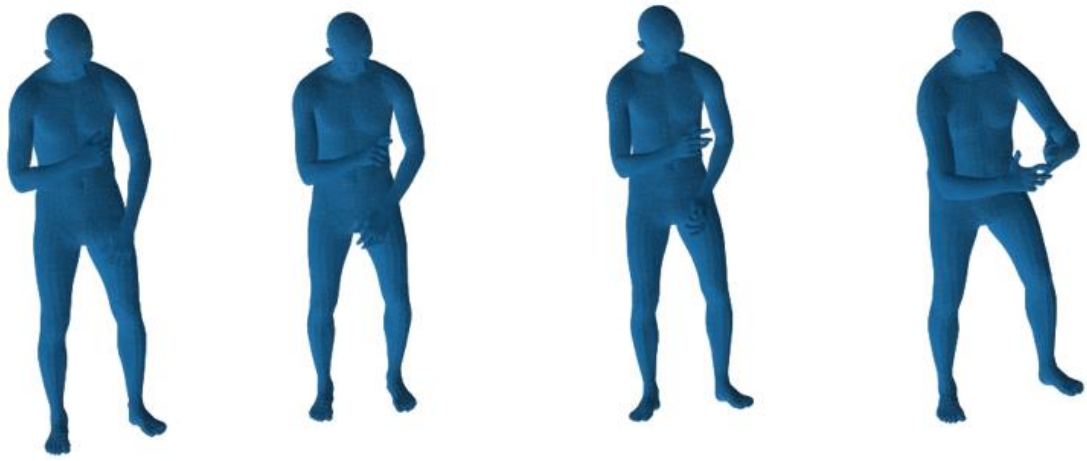

Figura 5.12 Fotogramas del video en LSM de la frase "Estoy aprendiendo LSM". Fuente: elaboración propia

Finalmente, la última frase utilizada para ver cómo se estaban recuperando los segmentos visuales y se generaba el video de la seña LSM, es una frase más larga, la cual es “Vamos a trabajar durante una hora”, teniendo como resultado el video que tiene las imágenes mostradas en la Figura 5.13. En esta última frase se tiene la palabra durante, el cual no se encuentra dentro del repositorio digital, prosiguiendo con el deletreo de la palabra.

Figura 5.13 Fotogramas del video en LSM de la frase "Vamos a trabajar durante una hora". Fuente: elaboración propia

A pesar de tener la recuperación de los segmentos visuales de una forma correcta y se unen estos segmentos dentro de un solo video para realizar las frases del español a la seña en LSM, se logra ver en el video el cambio de una seña a otra, lo cual será necesario para trabajos próximos hacer que se tenga un cambio entre señas más fluido.

En la Figura 5.14 se muestra una tabla con la cantidad de videos utilizados para formar las frases pruebas. En la primera columna se tienen las frases u oraciones formadas; en la segunda columna la cantidad de videos para formar la determinada frase u oración; en la tercera columna se tienen el nombre de los videos extraídos del repositorio digital; en la última columna se muestra si la extracción del orden de los videos fue la correcta. Como se puede ver las 3 primeras frases tienen un resultado aceptable donde se mantiene el orden de las palabras según la gramática de las señas y se logra entender la oración en seña LSM. En cambio, la ultima oración se considera fallido, ya que la frase se encuentra en tiempo futuro y se necesita agregar la seña para indicar este tiempo, así mismo, el momento de deletrear la palabra durante, el avatar no logra hacer correctamente los movimientos de los dedos, siendo complicado entender cual es la palabra que está deletreando.

Frase u Oración	Cantidad de videos	Videos usados	Correcto/Incorrecto
El alumno comprende la computación	3	• Computación • Alumno • Comprender	
La anciana abraza a su hija	4	• Suya • Hija • Anciana • Abrazar	
Estoy aprendiendo LSM	1	• Estoy aprendiendo LSM	
Vamos a trabajar durante una hora	12	• Nosotros • Ir • Durante(deletreado) • Trabajo • Uno • hora	

Figura 5.14 Tabla con la cantidad de videos utilizados para formar una determinada frase

Con lo mostrado anteriormente, podemos notar que el sistema utilizado para la recuperación de los segmentos visuales dentro del repositorio digital está funcionando de forma correcta, sin embargo, se necesita realizar más pruebas con mayor número de palabras y variantes.

CAPÍTULO 6. Conclusiones

Esta tesis tuvo como objetivo crear un modelo para la creación y uso de un repositorio digital de segmentos visuales en lengua de señas mexicanas (LSM) con la aplicación de la inteligencia artificial. Esto con el propósito de aportar una solución en la educación de las personas con discapacidad auditiva y de esta forma dar apoyo en el desarrollo de una educación inclusiva.

Para lograr este objetivo, se aprovechó los videos web dados por diferentes personas que tienen el conocimiento de la LSM, que tienen el propósito de inculcar en la sociedad la LSM y de esta forma incluir en la sociedad no sorda a personas sordas, sin presentar limitaciones por la comunicación. Estos videos los podemos encontrar en diferentes fuentes en la web, como en diccionarios webs o videos en YouTube. Cabe rescatar, que hay videos de la LSM que contienen la traducción de frases del español a la lengua de señas que no siguen una estructura gramatical de la LSM, por lo tanto, es importante saber que los videos obtenidos sean de fuentes confiables o de intérpretes que cuenten con la certificación de la LSM.

Asimismo, los videos recuperados de la web al ser de fuentes heterogéneas cuentan con diferente formato, es decir hay videos que tienen en su título la palabra o frase a interpretar en LSM y otros videos donde la palabra o frase a interpretar se encuentra dentro del video y dentro de un mismo video hay varias palabras a interpretar. La solución de lo anterior fue el uso de librerías las cuales cuentan con aprendizaje profundo capaz de detectar rostro y texto, obteniendo los videos de la LSM con la palabra o frase a interpretar. Sin embargo, con las pruebas realizadas se demostró que se presentan algunas fallas para ciertos videos como una falta en la detección del texto o un recorte no deseado de los videos, puesto que no se cuentan con un formato normalizado para realizar las lecturas. Se logró obtener 2455 palabras o frases en LSM para el repositorio digital.

Se logró transformar los videos obtenidos en la web a modelo 3D con la utilización de OpenPose y Smplify-x. Este proceso toma un tiempo alto para su ejecución y su transformación a modelos 3D, por lo tanto, no fue posible generar el modelo 3D de todos los videos obtenidos de la web. Así mismo, aun se presentan fallas para la obtención de los *keypoints* y desarrollar el modelo 3D de forma más precisa, esto se da ya que los videos obtenidos son videos 2D y no se tiene una referencia para detectar la profundidad. A pesar de lo anterior, se logra realizar pruebas con diferentes palabras y frases, y de esta forma se logró implementar en el código una fórmula para corregir algunas deformaciones del modelo 3D, dando un modelo más realista. Asimismo, el modelo generado fue un modelo el cual no cambiaba su aspecto físico a pesar de ser generado por diferentes videos.

Teniendo la generación de los modelos 3D, se logró definir el repositorio digital de segmentos visuales en LSM y para la recuperación de los segmentos se utilizaron diferentes algoritmos para obtenerlos. Estos algoritmos varían dependiendo de las diversas situaciones posibles, como la frase a interpretar ya se encuentra almacenada en el repositorio, si se tiene que obtener un video por cada palabra de la frase a interpretar o si es necesario deletrear una cierta palabra no encontrada dentro del repositorio. Aun hace falta la implementación del código desarrollado por [9], pero los resultados obtenidos mediante las pruebas realizadas demuestran que se podrán implementar de forma sencilla.

Finalmente, se logró la generación del video de la seña LSM con el avatar, este video puede ser implementado dentro de la plataforma desarrollada por [3], el cual será implementada en un futuro. Aun se necesita agregar una textura humana al modelo 3D, ya que éste se mantiene de un color azul, y no lo hace muy agradable visualmente.

Para un trabajo a futuro, se propone la necesidad de utilizar una super computadora, ya que el procesamiento de transformar los videos 2D a modelo 3D es de una capacidad muy alta siendo necesario su uso. Asimismo, se necesita buscar otro método para obtener una mejor detección de los *keypoints*, de esta forma obtener

resultados más precisos y evitar deformaciones del modelo 3D. También, se debe buscar una solución para generar una continuidad entre cada seña evitando la visión de cortes entre ellas. Y finalmente, resulta interesante la posibilidad de implementar una textura humana al modelo 3D haciendo más agradable la visión.

Con esta investigación, se logró abrir la posibilidad de desarrollar soluciones más automatizadas para generar un avatar capaz de realizar la interpretación entre el español a la lengua de señas mexicanas, aprovechando los videos 2D obtenidos por la web a modelo 3D, dando solución a investigaciones anteriores, las cuales utilizan métodos extensos y complicados, para el diseño del modelo 3D y sus movimientos de la seña en LSM.

Bibliografía

[1] M. Mariano, «Enfrentan personas con discapacidad auditiva retos en escenario actual,» Tecnologico de Monterrey, MONTERREY, 2020.

[2] Redacción SIPSE, «En México se hacen ciegos ante los sordos,» *SIPSE,* 1 Octubre 2016.

[3] E. D. Barraza Granillo, Plataforma para la asistencia de personas con discapacidad auditiva en el contexto de la educación [Tesis de Maestria], Hermosillo: Instituto Tecnologico de Hermosillo, 2018.

[4] L. A. Tovar, «La importancia del estudio de las lenguas de señas,» *ResearchGate,* nº 28, pp. 42-61, 2001.

[5] Consejo Nacional para el Desarrollo y la Inclusión de las Personas con Discapacidad, «Día Nacional de la Lengua de Señas,» *gob,* 2017.

[6] L. Escobar y Dellamary, «La lengua de señas mexicana, ¿una lengua en riesgo? Contacto bimodal y documentación sociolingüística.,» *Estudios de Lingüística Aplicada,* vol. año 33, nº 62, pp. 125-152, 2015.

[7] AVA, &AVA, 2020. [En línea]. Available: https://es.ava.me/. [Último acceso: 03 Enero 2021].

[8] E. Arenas, «Por una educación de verdad incluyente,» *El occidental,* 28 Septiembre 2019.

[9] J. C. Hernández Cruz, Traducción de Texto en Español a Texto LSM Usando Aprendizaje Profundo [Tesis de Maestria], Hermosillo: Instituto Tecnologico de Hermosillo, 2019.

[10] I. Chávez, «Tecnología en la escuela, ¿Para qué?,» *IMCO,* 24 Febrero 2020.

[11] L. Toval, «La importancia del estudio de las lenguas de señas,» *Cultura Sorda,* 2001.

[12] Consejo Nacional para el Desarrollo y la Inclusión de las Personas con Discapacidad, «Día Nacional de la Lengua de Señas Mexicana (LSM),» 09 Junio 2017. [En línea]. Available: https://www.gob.mx/conadis/articulos/dia-nacional-de-la-lengua-de-senas-mexicana-lsm?idiom=es. [Último acceso: 21 Marzo 2021].

[13] Diario Oficial de la Federación, *Ley General para la Inclusión de las Personas con Discapacidad,* Secretaria de gobernación, 2011.

[14] S. Siré, «Sordera: construyendo verdades y derribando pensares sociales,» *Cultura Sorda,* 2017.

[15] M. E. Serafín De Fleischmann y R. González Pérez, Manos con voz, Ciudad de México: Consejo Nacional para prevenir la discriminación , 2011.

[16] Sistema municipal DIF, Manual de lengua de señas mexicanas, puebla: DIF, 2018.

[17] S. B. Nava Oyorzabal, M. E. Tlapala Escobar y R. R. Meza Ramírez, «Biblingüismo, una forma de potenciar el aprendizaje en estudiantes sordos del estado de morelos,» *Conisen,* 2019.

[18] Chiapas Paralelo, «Sign'n, la aplicación que traduce el lenguaje de señas,» *Chiapas paralelo,* 16 Abril 2018.

[19] J. R. Mejía Vilet, Apuntes de Procesamiento Digital de Imágenes, San Luis Potosí: Facultad de Ingeniería UASLP, 2005.

[20] O. S. Ochoa Guevara, Artist, Implementación de un sistema de adquisición de imágenes multiespectrales, para integración en un vehículo aereo no tripulado. [Art]. Universidad de las fuerzas armadas, 2015.

[21] Meyda, «Píxeles y Resolución de la Imagen Digital,» Medya Escuela de fotografía, 09 julio 2014. [En línea]. Available: https://cursodefoto-madrid.com/pixeles-y-resolucion-de-la-imagen-digital/. [Último acceso: 21 04 2021].

[22] G. Gómez y L. E. Sucar, Visión Computacional, Puebla: Instituto Nacional de Astrofísica, Opticá y Electrónica, 2008.

[23] N. Aguirre Dobernack, Implementación de un sistema de detección de señales de trafico mediante Visión Artificial Basado en FPGA [Proyecto final de carrera], Sevilla: Universidad de Sevilla. , 2013.

[24] E. Portiansky, Multidimencional de Imagénes Digitales, La Plata: Universidad Nacional de La Plata, 2013.

[25] A. Cantero y D. Alcides, «Visión por computador: indentificación, clasificación y seguimiento de objetos.,» de *XXII Jornadas Jóvenes Investigadores*, Valparaíso, 2014.

[26] D. Marrr, *Vision,* San Francisco: Freeman, 1982.

[27] ESIC BUSINESS & MARKETING SCHOOL, «Modelado 3D: qué es, cómo funciona y la vida más allá de Pixar,» Febrero 2018.

[28] My info, «Comparison of Lara Croft from the first Tomb Raider to the Lara from the most recent Rise of the Tomb Raider game.,» Pinteres, [En línea]. Available: https://www.pinterest.com.mx/pin/412009065891132667/. [Último acceso: 18 Mayo 2021].

[29] R. Parent, Computer Aniamation Algorithms and Tecniques, San Diego: Morgan Kaufmann Publishers, 2002.

[30] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll y M. Black, «SMPL: A skinned Multi-Person Linear Model,» *ACM Trans. Graphics (Proc. SIGGRAPH Asia,* vol. 34, nº 6, pp. 248:1--248:16, 2015.

[31] Pons-Moll, G. Romero, J. Mahmood, N. Black y M. J, «Dyna: A Model of Dynamic Human Shape in Motio,» *ACM Transactions on Graphics, (Proc. SIGGRAPH),* vol. 34, nº 4, pp. 120:1--120:14, 2015.

[32] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas y M. J. Black, «Expressive Body Capture: {3D} Hands, Face, and Body from a Single Image,» *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),* pp. 10975-10985, 2019.

[33] S. O. Caballero Morales y F. Trujillo Romero, «3D Modeling of the Mexican Sign Language for a Speech-to-Sign Language System,» *Computación y sistemas,* vol. 17, nº 4, pp. 593-608, 2013.

[34] J. Chaves Sanchez, C. Ramírez Trejos y M. Chacón Rivas, «Interacción con avatar 3D para la creación y edición de poses corporales y faciales en la lengua de señas costarricense LESCO,» de *Avances sobre reflexiones, aplicaciones y tecnologías inclusivas*, Ciudad de México, conaic, 2019, pp. 34-40.

[35] Asura, «Inteligencia Emocional de Robert Plutchik,» Velador de palabras, [En línea]. Available: https://veladordepalabras.blogspot.com/2019/09/inteligencia-emocional-de-robert.html. [Último acceso: 25 Mayo 2021].

[36] M. Carguacundo y P. Constante, «Traductor de texto y voz a la lengua de señas ecuatoriana a través de un avatar implementado para dispositivo Android,» *Infociencia,* vol. 12, nº 1, pp. 20-25, 2018.

[37] M. Ahmed, M. Idrees, Z. u. Abideen, R. Mumtaz y S. Khalique, «Deaf talk using 3D animated sign language A sign language intepreter using Microsofy´s Kinect v2,» *SAI Computing Conference 2016,* pp. 330-335, 2016.

[38] L. IAPPPS, 25 palabras y frases en LSM para principiantes. Aprende Lengua de Señas Mexicana, Youtube [mp4], 13 febrero 2020.

[39] M. López, *VERBOS LSM | APRENDE LSM EN 5 MIN.,* Youtube [MP4], 4 abril 2019.

[40] A. Rosebrock, «OpenCV Text Detection (EAST text detector),» pyimagesearch, 20 Agosto 2018. [En línea]. Available: https://www.pyimagesearch.com/2018/08/20/opencv-text-detection-east-text-detector/. [Último acceso: 21 Noviembre 2021].

[41] A. Rosebrock, «Non-Maximum Suppresion for object detection in python,» pyimagesearch, 17 Noviembre 2014. [En línea]. Available: https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/. [Último acceso: 21 Noviembre 2021].

[42] E. b. d. laura, *Alimentos en LSM 2021,* Youtube [MP4], 16 Marzo 2021.

[43] Wikisigns, «Lengua de señas mexicanas,» Sign Language Dictionaries of the World, [En línea]. Available: https://www.wikisigns.org/es/lsm/diccionario. [Último acceso: 21 Noviembre 2021].

[44] M. T. Calvo Hernández, Diccionario Español -lengua de señas mexicana (DIELSEME).

[45] Z. Cao, G. Hifalgo Marinez, T. Simon, S. Wei y Y. A. Sheikh, «OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2019.

[46] T. Simon, H. Joo, I. Matthews y Y. Sheikh, «Hand Keypoint Detection in Single Images using Multiview Bootstrapping,» de *CVPR*, 2017.

[47] Z. Cao, T. Simon, S.-E. Wei y Y. Sheikh, «Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,» de *CVPR*, 2017.

[48] S.-E. Wei, V. Ramakrishna, T. Kanade y Y. Sheikh, «Convolutional pose machines,» de *CVPR*, 2016.

[49] T. Simon, S.-E. Wei y Y. Sheikh, «The first real-time multi-person system to jointly detect human body, hand, facial, and foot keypoints,» Doxygen 1.9.1, [En línea]. Available: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_02_output.html. [Último acceso: 08 Junio 2022].

[50] nghorbani, «VPoser: Variational Human Pose Prior for Body Inverse Kinematics,» github, 04 Mayo 2022. [En línea]. Available: https://github.com/nghorbani/human_body_prior. [Último acceso: 09 Junio 2022].

[51] pclausen, «obj2png,» github, 04 Julio 2021. [En línea]. Available: https://github.com/pclausen/obj2png/tree/main/src. [Último acceso: 06 Junio 2022].

[52] L. Da Vinci, Artist, *El Hombre de Vitruvio.* [Art]. 1490.

[53] J. Medlej, «Fundamentos de anatomía: Proporciones del cuerpo humano,» evatotuts+, 19 Junio 2021. [En línea]. Available: https://design.tutsplus.com/es/articles/human-anatomy-fundamentals-basic-body-proportions--vector-18254. [Último acceso: 18 06 2022].

[54] A. Solis Muñiz, R. Quiróz Clemente y M. E. Culebro Mandujo , «El software educativo en el aprendizaje del lenguaje de señas mexicano aplicado en el DIF del estado de Chiapas,» de *somece2015*, Tuxtla Gutiérrez, 2015.

[55] Google LLC, «mediapipe,» 2020. [En línea]. Available: https://mediapipe.dev/. [Último acceso: 2021 diciembre 30].

[56] «mediapipe,» [En línea]. Available: https://mediapipe.dev/. [Último acceso: 2021 diciembre 30].

[57] T. Roosendaal, «Blender,» Blender Foundation, [En línea]. Available: https://www.blender.org/. [Último acceso: 5 Enero 2022].

[58] ICHI.PRO, «Cómo crear su propia aplicación de traducción de lenguaje de señas extendiendo SigNN,» [En línea]. Available: https://ichi.pro/es/como-crear-su-propia-aplicacion-de-traduccion-de-lenguaje-de-senas-extendiendo-signn-25475798465453. [Último acceso: 10 Enero 2022].

[59] G. a. C. V. a. G. N. a. B. T. a. O. A. A. A. a. T. D. a. B. M. J. Pavlakos, «SMPL eXpressive,» Max-Planck-Gesellschaft, 2020. [En línea]. Available: https://smpl-x.is.tue.mpg.de/download.php. [Último acceso: 31 05 2022].

[60] zonkosoft, «BlendyPose,» Github, 19 Junio 2021. [En línea]. Available: https://github.com/zonkosoft/BlendyPose/blob/b3cc7e31324a777723c87fae9815f06d6b2b6bc0/blendy_pose.py. [Último acceso: 02 Junio 2022].