DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

TEMA: DESARROLLO DE UNA BASE DE DATOS DEL LENGUAJE DE SIGNOS ESPAÑOL ECUATORIANO A TRAVÉS DEL SENSOR KINECT V2 DE XBOX ONE

AUTORES: HU, ZHANPENG
TERÁN MEDINA, LUIS ADRIÁN

DIRECTOR: Ing. LARCO BRAVO, JULIO CÉSAR

SANGOLQUÍ

2017-2018

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICACIÓN

Certifico que el trabajo de titulación "**DESARROLLO DE UNA BASE DE DATOS DEL LENGUAJE DE SIGNOS ESPAÑOL ECUATORIANO A TRAVÉS DEL SENSOR KINECT V2 DE XBOX ONE**" realizado por los señores **HU ZHANPENG** y **TERÁN MEDINA LUIS ADRIÁN,** ha sido revisado en su totalidad y analizado por software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE, por lo tanto me permito acreditarlos públicamente y autorizar a los señores **HU ZHANPENG** y **LUIS ADRIÁN TERÁN MEDINA** para que lo sustenten públicamente.

Sangolquí, 13 de diciembre del 2017

Ing. Julio Larco
Director del proyecto

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

AUTORÍA DE RESPONSABILIDAD

Nosotros, **HU ZHANPENG** con cédula de identidad N°1716053853 y **LUIS ADRIÁN TERÁN MEDINA** con cédula de identidad N° 1721542007, declaramos que este trabajo de titulación "**DESARROLLO DE UNA BASE DE DATOS DEL LENGUAJE DE SIGNOS ESPAÑOL ECUATORIANO A TRAVÉS DEL SENSOR KINECT V2 DE XBOX ONE**" ha sido desarrollada considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente Declaramos que este trabajo es de nuestra autoría, en virtud de ello nos declaramos responsables del contenido, veracidad y alcance de la investigación mencionada.

Sangolquí, 13 de diciembre del 2017

Luis Terán
1721542007

Zhan Peng Hu
1716053853

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN

Nosotros, **HU ZHANPENG** y **LUIS ADRIÁN TERÁN MEDINA**, autorizamos a la Universidad de las Fuerzas Armadas-ESPE publicar en la biblioteca virtual de la institución el presente trabajo de titulación "**DESARROLLO DE UNA BASE DE DATOS DEL LENGUAJE DE SIGNOS ESPAÑOL ECUATORIANO A TRAVÉS DEL SENSOR KINECT V2 DE XBOX ONE**" cuyo contenido, ideas y criterios son de nuestra autoría y responsabilidad.

Sangolquí, 13 de diciembre del 2017

Luis Terán
1721542007

Zhan Peng Hu
1716053853

DEDICATORIA

Quiero dedicar este trabajo principalmente a mi familia, amigos y personas que siempre estuvieron en todo momento. A mi padre Jaime Terán quien con su sabiduría y apoyo aportó para las decisiones que tomé en la vida. A mis dos hermanos Jaime y Paúl que a pesar que el segundo ya no está con nosotros siempre estuvo presente y lo estará a lo largo de mi vida. A mi madre Elizabeth quién se preocupó de mi alimentación adecuada fuera de casa procurando que mi salud sea la más óptima para mi desempeño. A Lissette Acurio por acompañarme en mis desveles sin decaer ni un instante, incluyendo dificultades, tristezas y alegrías. A mi futuro hijo quien me motivó a alcanzar la meta lo más rápido posible y sea motivo de alegría en mi desarrollo personal y profesional. A todos gracias….

Luis Terán

Este trabajo quiero dedicarlo a mis seres queridos como mi familia y amigos porque siempre me ayudaron a superar las dificultades que se presentaron a lo largo de mi desarrollo. A mi padre Hu Guo Min quien supo darme la paciencia y entenderme para tomar decisiones importantes guiándome en mi vida. A mi madre Huang Mu Quin que con su amor, apoyo y dedicación siempre me ha ayudado a impulsarme adelante. A mi hermano Zhan Chang que siempre ha estado a mi lado en las buenas y en las malas. Muchas gracias a todos.

Zhan Peng Hu

AGRADECIMIENTO

Agradecemos a nuestros padres quienes con su paciencia y cariño han sabido ser soporte y guía, siempre brindándonos su apoyo incondicional.

A Dios por darnos sabiduría, paciencia, salud, vivienda y proveernos del pan de cada día los cuales son necesarios para el bienestar y desarrollo profesional.

A los amigos que estuvieron siempre dispuestos a brindarnos una mano y aportar con conocimiento a la concepción de este trabajo y a nuestros amigos con los cuales compartimos grandes experiencias en la etapa universitaria.

Al Instituto Nacional de Audición y Lenguaje por ofrecer el tiempo necesario para la captura de muestras de nuestro proyecto. Se desea hacer una mención especial al Técnico Iván Unda quien nos colaboró con el tiempo de sus clases y trabajando horas extra para la corrección de la base de datos.

A la Universidad de las Fuerzas Armadas-ESPE por recibirnos y brindarnos los conocimientos necesarios durante el transcurso de la carrera. Al grupo de trabajo del CIRAD por aportarnos su experiencia para el desarrollo del presente trabajo. Al ingeniero Julio Larco por aportar con sus conocimientos, ayuda y paciencia durante el transcurso del proyecto guiándonos a la culminación de este objetivo con su constante orientación.

Muchas gracias a todos.

Luis Terán y Zhan Peng Hu

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En la actualidad el uso de la tecnología busca maneras de mejorar el estilo de vida de las personas con discapacidad auditiva, quienes se sienten excluidas de la sociedad debido a la discriminación por no poder comunicarse con las personas que no conocen el lenguaje de signos. Una manera de evitar este problema es contar con aplicaciones de traducción de lenguaje de signos a lenguaje escrito o voz. Para lo cual es necesario tener una base de datos que cuente con información obtenida de personas profesionales en el tema de donde se pueda extraer muestras discriminativas y representativas de las señas efectuadas al momento de emplear signos. Este trabajo presenta la creación de una base de datos del lenguaje dactilológico español ecuatoriano utilizando el dispositivo Kinect v2.0 de Xbox One con la meta de brindar una fuente para un futuro reconocimiento del lenguaje de signos. Para su creación se capturaron muestras de video de color, video de profundidad y datos de detección, con la ayuda de personal calificado del Instituto Nacional de Audición y Lenguaje. La base de datos creada cuenta con muestras de un conjunto de 5 tipos de palabras de uso común del lenguaje dactilológico ecuatoriano, como son: adjetivos, alimentos, saludos, juguetes & cosas y colores. Además está compuesta por letras estáticas, sin cambio o movimientos adicionales, y letras dinámicas. Utilizando algoritmos de aprendizaje de máquina se realizó la verificación de la base de datos mediante pruebas de reconocimiento de signos a partir de los datos almacenados.

PALABRAS CLAVE:

- ***SKELETON TRACKING***
- ***COLORIMG***
- ***DEPTHIMG***
- ***CLASSIFICATION LEARNER***
- **MÁQUINAS DE SOPORTE VECTORIAL**

ABSTRACT

Nowadays, the use of technology seeks ways to improve the lifestyle of people with hearing disabilities, who feel excluded from society due to discrimination because they cannot communicate with people who do not know sign language. One way to avoid this problem is to have applications for translation from sign language to written language or voice. Cause of that, it is necessary to have a database that has information obtained from professional people in the subject from which discriminatory and representative signals can be extracted of the signs made at the time of applying signs. This work presents the creation of a data base of the Ecuadorian Spanish dactylographic language using the Kinect v2.0 device of Xbox One with the goal of providing a source for a future recognition of sign language. For its creation, samples of color video, depth video and detection data were captured, with the help of qualified personnel from the Instituto Nacional de Audición y Lenguaje. The database created has samples of a set of 5 types of words commonly used in the Ecuadorian sign language, as a child: adjectives, food, greetings, toys and things and colors. In addition, it is composed of static letters that are constant and do not change, and dynamic letters. Using machine learning algorithms, the database was verified by sign recognition tests from the stored data.

KEYWORDS:

- **SKELETON TRACKING**
- **COLORIMG**
- **DEPTHIMG**
- **CLASSIFICATION LEARNER**
- **SUPPORT VECTOR MACHINES**

CAPÍTULO I
DESCRIPCIÓN GENERAL DEL PROYECTO

1.1 Introducción

En la actualidad se busca solventar dificultades en la comunicación de manera que las personas tengan la aptitud de desenvolverse e interactuar con la sociedad, una de las problemáticas es la falta de comunicación con las personas con discapacidad auditiva, para resolver esta complicación se creó el lenguaje dactilológico o lenguaje de signos para interactuar entre las personas con dichos problemas y aquellas que conocen este lenguaje y no cuentan de discapacidad, sin embargo el uso de este lenguaje no es común para las personas que no tienen esta discapacidad, por ello siempre se presentan inconvenientes con su interacción creándose barreras al momento de desenvolverse en su entorno. El hecho de disponer de aplicaciones que interpreten este lenguaje sería de utilidad dado que facilitaría la comunicación. Para el desarrollo de estas aplicaciones se requieren de una base de datos de señas del lenguaje de signos de tal manera que sirvan para el desarrollo de una aplicación que interprete la lengua para las personas que no lo conocen.

Para el desarrollo de una base de datos se hará uso del dispositivo Kinect v2.0 de Xbox One ya que cuenta con la capacidad de rastreo del estado de la mano, y otras versiones de Kinect no disponen de dicha capacidad, con el fin de crear una base de datos de palabras y abecedario del lenguaje de signos español de las cuales poseen letras con movimiento en su ejecución como las letras “j” y “z”, que otros trabajos (Morocho, 2016) no usaron ya que en el Ecuador se han desarrollado intérpretes basándose en imágenes estáticas.

El proyecto requiere del cumplimiento de varias tareas como es la obtención de datos, procesamiento requerido para el almacenamiento de cada signo y palabra para posteriormente proceder con la elaboración de la base de datos del lenguaje dactilológico español.

1.2 Planteamiento del Problema

En los últimos tiempos, el uso de la tecnología busca maneras de mejorar el estilo de vida de las personas, entre estas tecnologías algunas están enfocadas para personas con discapacidad auditiva, las que pueden sentirse excluidas de la sociedad debido a la discriminación por no poder comunicarse con las personas que no conocen el lenguaje de signos. Una manera de evitar este problema es el uso del lenguaje de signos para que puedan interactuar en su entorno. Generalmente, las personas no conocen o dominan dicho lenguaje al no tener las mismas capacidades que ellos influyendo en gran manera hacia aquellos que si lo poseen.

Para desarrollo y análisis de aplicaciones de traducción de lenguaje de signos a lenguaje escrito usando Kinect, es necesario tener una base de datos que posea varios tipos de información captadas por el dispositivo sobre el cual trabajar, como en (Zhang, Xu, & Sun, 2015), la base de datos detallaran videos del movimiento de los signos donde se podrá extraer muestras discriminativas y representativas de las señas efectuadas para lograr un resultado efectivo y preciso al momento de emplear algoritmos para traducir de forma fiable el lenguaje dactilológico a lenguaje escrito, sin embargo, para resolver este problema es necesario contar con una base de datos, de las cuales es importante que sea realizado por personas profesionales del tema para que sea confiable.

Por esta razón, el desarrollo de una base de datos del lenguaje de dactilológico español es requerido y permitirá contribuir en la creación de aplicaciones que sirvan para traducir del lenguaje de signos a lenguaje escrito.

1.3 Justificación e Importancia

La innovación tecnológica es una de las áreas de mayor interés en los últimos años, proporcionando mayores facilidades en la vida de las personas, en especial las personas con discapacidad, facilitando su interactividad con la sociedad.

En el Ecuador existen diversos tipos de discapacidades, alrededor del 12.86% del total de personas con discapacidades tienen discapacidad auditiva y 1.31% de lenguaje, según datos estadísticos del Consejo Nacional para la Igualdad de Discapacidades

(CONADIS C. N., 2017). Para estas personas existe un problema social al no existir facilidad de comunicación con quienes no poseen dichas discapacidades. Las personas con discapacidad auditiva, en especial los niños, sufren problemas de integración entre las personas con problemas auditivos y aquellas que no lo poseen por lo que es necesario un medio de comunicación como es el lenguaje de signos.

El reconocimiento de voz y movimientos proporciona una inmensidad de soluciones a esta problemática. Apoyado en el lenguaje de signos, se han realizado proyectos para proveer una mayor integración social.

En el país, solo se han realizado proyectos de traducción simultánea de lenguaje de signos a voz usando Kinect v1.0 mediante una interfaz natural de usuario para personas con discapacidad auditiva. Uno de estos es un prototipo que facilita la comunicación e interacción con personas que desconocen el lenguaje de signos, rastreando solamente la posición de la mano, pero no puede interpretar de manera dinámica los movimientos de la mano, los cuales son fundamentales para la formulación de palabras, y letras que requieren movimiento como por ejemplo son la “j” y “z”. (Morocho, 2016)

También se desarrolló un traductor de lenguaje de señas simultánea con 30 signos en una interfaz gráfica de usuario, mediante algoritmos de predicción de movimiento manipulando la técnica de *Extreme Programming* (XP), basados en ciclos de desarrollo breves con un solo usuario como encargado para el análisis de signos, sin la posibilidad de aumentar el conjunto de palabras para formulación de oraciones. (Criollo, Wilfrido, & Cargua, 2015)

Existe un proyecto para traducción simultánea el cual ofrece una solución para el problema de comunicación, el mismo sugiere un modelo para reconocimiento de lenguaje de signos utilizando una máquina de soporte vectorial en sus siglas en inglés (SVM) apoyándose de una base de datos del lenguaje de signos americano en sus siglas en inglés (ASL). (Zhang, Xu, & Sun, 2015) (Sun, Zhang, Bao, Xu , & Mei, 2013)

Para desarrollar una aplicación que realice traducción simultánea, el primer paso es contar con una base de datos del lenguaje de signos español ecuatoriano (LSEC) adecuada, que posea suficientes muestras de palabras y letras para análisis lo cual no existe en el país, debido que no son públicos o no fueron realizados por profesionales que conocen sobre el tema. Este proyecto pretende solventar este problema creando una base de datos pública de uso libre para el lenguaje dactilológico español, cuyas muestras sean obtenidas con personal calificado que garantizará que los signos realizados son los correctos.

Además la realización de este trabajo apoyará a proyectos que formen parte del Instituto Nacional de Audición y Lenguaje (INAL), quienes requieren sistemas de apoyo para el entendimiento de lenguaje de signos que contenga palabras comunes de colores, alimentos, útiles escolares y adjetivos, pueden ayudar al entendimiento para personas que desconocen el lenguaje.

1.4 Alcance

El proyecto plantea proporcionar una base de datos captados por Kinect V2 de Xbox One que pueda ser empleada para el desarrollo o análisis de una aplicación que interprete el lenguaje dactilológico español y la traduzca a lenguaje escrito.

Mediante el uso de Kinect v2.0 el cual brinda mayor resolución de video de 1920×1080 se busca una traducción más fiable con mayor porcentaje de aciertos en comparación a aplicaciones realizadas con Kinect v1.0 que cuenta con 1280×960.

Se proyecta que la base de datos esté constituida por las letras del alfabeto español de la letra “a” a la “z”, incluyendo las letras “j” y “z” que cuentan con movimiento en su ejecución en el lenguaje de signos.

La base de datos que vamos a crear incluirá la formulación de palabras de uso común, como colores, alimentos, útiles escolares y adjetivos, que serán determinadas por el Instituto Nacional de Audición y Lenguaje, los cuales son quienes conocen de esta

problemática y están capacitadas para darnos la información adecuada para formar este grupo de palabras.

Las tareas a realizarse en este proyecto son las siguientes:

- Determinar las palabras de uso común que formarán la base de datos tomadas por personas con discapacidades auditivas.
- Determinar una metodología para la captura de datos.
- Elaboración de una programa en Matlab para la captura de datos.
- Procesamiento de la información captada del lenguaje de signos tanto de letras como palabras.
- Elaboración de la base de datos de las letras del alfabeto español.
- Elaboración de la base de datos para las palabras del dialecto español.

1.5 Objetivos

1.5.1 General

Crear una base de datos del lenguaje dactilológico Español mediante el uso de la Kinect v2 de Xbox One.

1.5.2 Específicos

- Describir el lenguaje dactilológico español para personas con discapacidades de comunicación.
- Determinar las palabras de uso común del lenguaje dactilológico español.
- Crear una base de datos de las letras del abecedario español del lenguaje dactilológico.
- Crear una base de datos de palabras como adjetivos, colores, saludos y alimentos del lenguaje dactilológico
- Analizar las muestras obtenidas mediante pruebas de la base de datos.

1.6 Estructura del Documento

El presente documento consta de seis capítulos. El primer capítulo contiene los detalles relevantes del proyecto como son una breve introducción, el planteamiento del

problema, justificación e importancia, alcance, objetivos generales y específicos del proyecto.

En el capítulo dos “Marco teórico”, detalla la documentación teórica del proyecto, el cual se compone de información sobre los dispositivos disponibles de *Kinect* y MATLAB con sus respectivas características, y fundamentos teóricos acerca de aprendizaje de máquina que será empleado para la verificación de la base de datos.

En el capítulo tres “Creación de base de datos”, se explica la metodología empleada para su elaboración, a su vez, se indica los pasos empleados para la adquisición de muestras, codificación y estructura en las que se guardarán las mismas, y, por último se presenta el diseño de un programa que simplifique y facilite al usuario el proceso de administración de la base de datos.

En el capítulo cuatro “Verificación de la Base de Datos”, se presenta una opción de aprendizaje de máquina con la cual es posible realizar un reconocimiento de signos a partir de la base de datos creada, esta se compone de los pasos a seguir para procesar las muestras de la base, los pasos para emplear *Classification Learner* (aplicación del *toolbox* de MATLAB) y el método empleado para la creación del algoritmo.

En el capítulo cinco “Análisis de resultados”, se presentará las diferentes matrices de confusión y porcentajes de precisión obtenidos a partir de los algoritmos de aprendizaje de máquina que fueron realizados para distintas pruebas con máquinas de vector de soporte y bajo diferentes parámetros. Se presentará varios modelos indicando con cual se obtuvo mejores resultados.

En el capítulo seis “Conclusiones, Recomendaciones y Trabajos Futuros”, se presentan las conclusiones y recomendaciones obtenidas durante la adquisición de muestras, elaboración de este proyecto y la validación de la base de datos, y presenta propuestas para trabajos futuros.

CAPÍTULO II
MARCO TEÓRICO

Este capítulo contiene conceptos generales acerca del lenguaje dactilológico español, sus características, además se incluye información del dispositivo Kinect y el lenguaje de cálculo técnico MATLAB, sus correspondientes funcionamientos, características y aplicaciones. Se explica en este capítulo también sobre los *toolbox* de MATLAB como *Image Acquisition Tool* para capturas de imágenes o video, y *Classification Learner* que permite entrenamiento de máquina para algoritmos de aprendizaje mediante máquinas de vectores de soporte, se indican también los elementos necesarios para la elaboración de una base de datos. Adicionalmente, se incluye detalles sobre el lenguaje de dactilológico español.

2.1 Lenguaje dactilológico español

El lenguaje dactilológico es una representación del abecedario en el espacio que es utilizada como un sistema de comunicación por las personas con capacidades especiales y las que conviven en relación con ellas.

El idioma que se usa por una persona con discapacidad auditiva no emita sonidos, en cambio, se expresa con señas con diferentes matices, además que varían dependiendo de modismos locales, por lo cual este lenguaje no es universal y se encuentra limitado para cada país, de tal manera que aquellos que lo emplean, están bajo un reglamento y código lingüístico establecidos dependiendo su lengua oral.

2.1.1 Lenguaje de signos ecuatoriano (LSEC)

El lenguaje de signos es un lenguaje visual que emplea un sistema de movimientos manuales, faciales y corporales como medio de comunicación. El lenguaje de signos no es un lenguaje universal, y distintos lenguajes de signos se utilizan en

diferentes países, de los cuales se empleará el lenguaje de signos ecuatoriana (LSEC) para la creación de la base de datos del presente proyecto.

El lenguaje de signos ecuatoriana (LSEC), es el lenguaje de signos empleado por las personas con problemas auditivos y de lenguaje en Ecuador, que a través del Art. 70 de la Ley Orgánica de Discapacidades del Ecuador, es reconocida como lengua propia y medio de comunicación de las personas con discapacidad auditiva.

Ser capaz de comunicarse e interactuar con la sociedad es clave para el desarrollo cognitivo, social, emocional y lingüístico de cualquier persona. (CONADIS C. y., 2013)

2.1.2 Alfabeto dactilológico

El alfabeto dactilológico es el alfabeto utilizado por la gente con discapacidad auditiva, que consiste en la representación manual de las letras del alfabeto de la lengua oral.

La representación es ejecutada en el espacio cercano a la cara a la altura de la barbilla, a su vez, se emplea la mano dominante en el caso de los diestros la derecha y zurdos la izquierda.

La dactilología se considera como un instrumento del lenguaje de signos que sirve para enlazar la lengua de signos y la lengua oral, por ello suele ser empleada para referenciar nombres propios, representar e identificar palabras de la lengua oral cuando se presentan por primera vez, se desconoce o no tienen signo.

Cuando existe una seña para una palabra, la dactilología se deja de emplear, pero para una persona que aprende el lenguaje de signos, es importante desarrollar destrezas receptivas y expresivas, para adquirir, producir y comprender el alfabeto dactilológico. (Vilela, 2005)

A continuación se muestra el alfabeto dactilológico: (FENASEC, 2012)

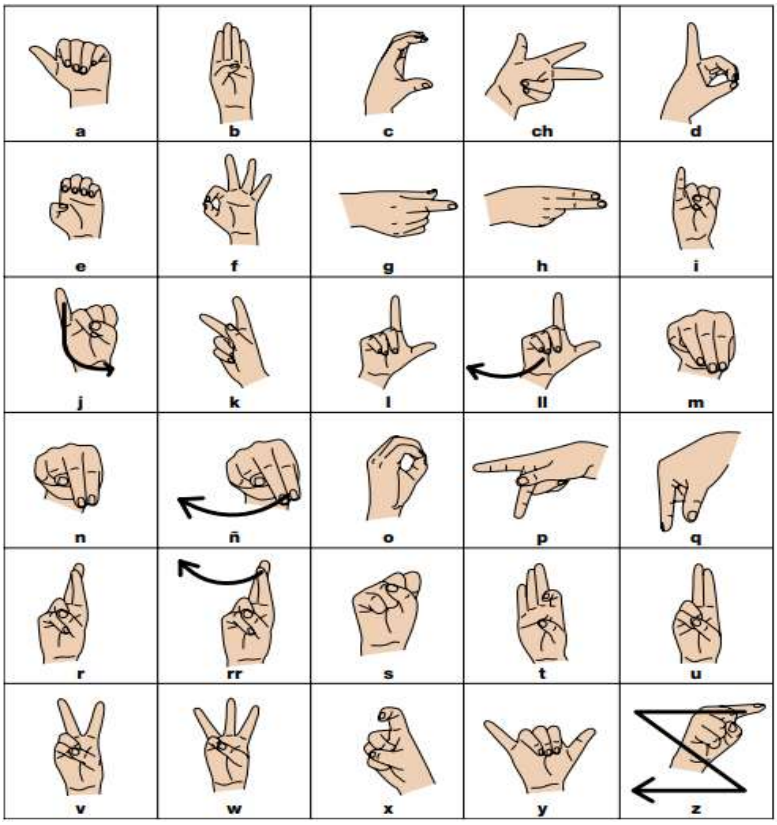


Figura 1 Alfabeto dactilológico

Fuente: (FENASEC, 2012)

2.1.3 Clasificación del lenguaje de signos

El lenguaje dactilológico universal principalmente se clasifica en diferentes clases que se detallan a continuación: (Villacis, 2014)

- Sistema de uso manual: Es el conjunto de formas de expresión lingüística a través del uso de una o dos manos.
- Sistema de uso gestual y expresivo: Es el conjunto de formas de expresión con el uso del rostro.
- Sistema de uso corporal: Es el conjunto de formas de expresión a través del uso de partes del cuerpo excepto las manos y partes del rostro como los pies, los hombros, el tronco, y el cuello.

- Sistema de empleo de objetos: Es el conjunto de formas de expresión a través del uso de ciertos objetos o herramientas para obtener un vocabulario mucho exacto, amplio y simplificado.

En el ecuador existe un diccionario de lengua de señas ecuatoriana "Gabriel Román", la cual consiste de 5000 palabras con imágenes y videos explicativos, el cual puede servir para conocer el significado de ciertas palabras comunes, fue desarrollado por el Consejo Nacional de Igualdad de Discapacidades (CONADIS) en formato web, junto con el apoyo de la Federación Nacional de Sordos del Ecuador (FENASEC), y la Universidad Tecnológica Indoamérica – UTI. (FENASEC & UTI, 2016)

2.2 Dispositivo Kinect v2

El dispositivo Kinect es un módulo físico que cuenta con una tecnología de detección sensible, una cámara a color integrada, un emisor de infrarrojos y una matriz de micrófonos; gracias a ello, se puede detectar tanto la ubicación, los movimientos y voz de los usuarios, es posible conectar el dispositivo a un PC o tableta, permitiendo el desarrollo de soluciones personalizadas y aplicaciones que hagan uso de sus sensores.

2.2.1 Sistema del dispositivo Kinect

Para hacer uso de este dispositivo es necesario revisar que el computador cumpla con los requisitos de hardware antes de seguir los pasos de configuración del sensor; de ese modo es posible desarrollar aplicaciones con Windows y Kinect, es necesario actualizar el software de la PC para la detección precisa y seguimientos de voz, cuerpo o faciales. En la figura 2 se muestra imágenes captadas por el sensor de profundidad de Kinect.

Figura 2 Sensor de Profundidad de Kinect

Fuente: (Microsoft, 2014)

2.2.2 Diferencias entre Kinect v2 y Kinect v1

El desarrollo de Kinect v2 mejora considerablemente ciertos aspectos de su antecesor la Kinect v1, considerando aplicaciones previamente realizadas por los mismos se mencionan las siguientes: (Marcal, 2013)

- **Mayor campo de visión. 70º en horizontal (antes 57º) y 60º en vertical (antes 43º).**

Esto permite la detección de mayor número de personas en un mismo campo de visión, hasta 6 pueden ser detectados simultáneamente. En diferencia a Kinect v1, Kinect v2 no tiene motor de inclinación por lo que permite una mayor precisión en la detección.

- **Mayor resolución. 1920 x 1080 Full High Definition (HD) (antes 640 x 480).**

Permite detectar con más precisión todo el entorno. Permite una orientación del cuerpo incluida las manos, que facilita la diferenciación de los dedos en el empleo de *skeleton tracking* o reconocimiento del esqueleto humano, obteniendo mejores

resultados al momento de la detección del movimiento en los dedos o la palma de las manos.

Incluye un *face tracking* o reconocimiento de rostro con mayor detalle con lo que es posible la captura de gestos faciales, con una mayor calidad en la imagen total.

- **Mejora el rango de profundidad del sensor.**

El rango de actuación pasa a ser de 0,5 a 4,5 metros, con lo que es posible una mejor percepción del entorno en 3D, para cualquier tipo de aplicación interactiva.

- **USB 3.0**

Permite aumentar la velocidad de la comunicación con el ordenador reduciendo la latencia del sensor. Pasa de 90ms a 60ms.

- **Mejora de la captación de sonidos.**

Existe una optimización y mejora respecto al reconocimiento de la voz y sonidos. Al eliminar el ruido del ambiente permite captar instrucciones vocales con un fallo mínimo.

- **Captación de movimiento a oscuras.**

Kinect v2 es capaz de reconocer y captar los movimientos a obscuras, mejorando así el contraste de la imagen en un ambiente no tan claro.

- **Kinect v2 permite calcular/analizar la fuerza de músculos y medición del ritmo cardíaco.**

Existe una funcionalidad adicional que permite destacar y remarcar estas dos como experiencias de interacción diferentes a las habituales.

2.2.3 Componentes del dispositivo Kinect v2

En la figura 3 se presentan los componentes del dispositivo Kinect v2:

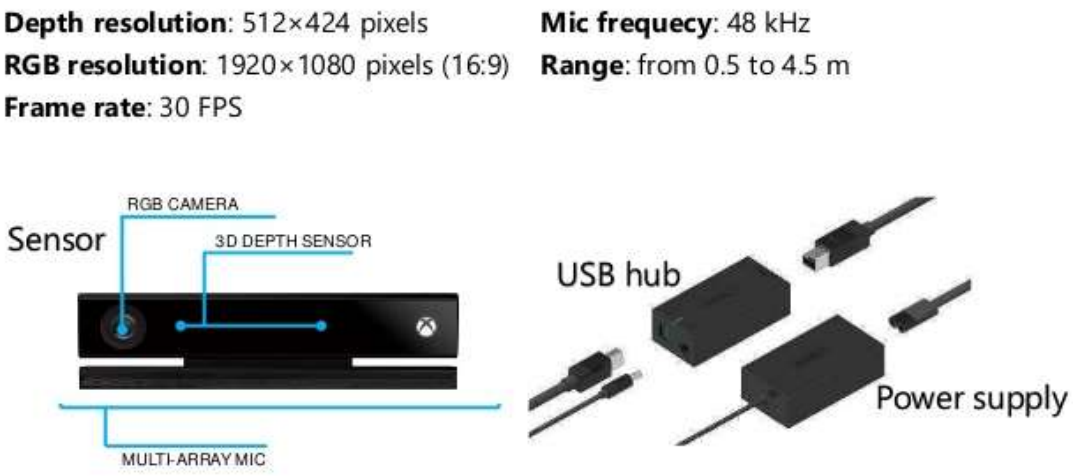


Figura 3 Componentes del Sensor

Fuente: (Oliveto, 2015)

Para la conexión se requiere de una fuente de poder mostrada en la figura 3, la cual permite la comunicación entre el sensor y la computadora.

2.2.4 Arquitectura de dispositivo Kinect (Oliveto, 2015)

Las características de la arquitectura del dispositivo Kinect se detallan a continuación:

- El sensor es un recurso y varias aplicaciones pueden acceder simultáneamente.
- El sensor da un conjunto de funcionalidades que serán conocidas como fuentes.
- Por cada fuente es posible iniciar lectores
- Cada lector provee eventos para adquirir referencias a los cuadros del dispositivo
- Por cada cuadro es posible obtener dato acerca de cada fuente específica. (color, imagen, datos de cuerpo, etc.) (Oliveto, 2015)

Se muestra en la figura 4 se muestra el diagrama de bloques de la arquitectura del dispositivo Kinect:

Figura 4 Diagrama de bloques de la arquitectura del dispositivo Kinect

A continuación se describen cada uno de los bloques que conforman el diagrama de bloques:

2.2.5 Sensor del dispositivo Kinect

- **Uso del Sensor**

Para el uso del dispositivo se realizan los siguientes pasos:

- Obtener una instancia del dispositivo Kinect
- Abrir el sensor
- Usar el sensor
- Cerrar el sensor

- **En caso de que el sensor sea desconectado**

- El dispositivo Kinect instancia una validación remanente.
- No se enviará/recibirá más *frames*.
- La propiedad de la disponibilidad del sensor se vuelve falsa.

- **Fuentes**

Las fuentes son tipos de datos extraíbles a partir de los sensores de Kinect, se muestra a continuación los diferentes tipos de fuentes por cada una de sus funcionalidades:

- Fuente de Color (*Color source*).
- Fuente de Profundidad (*Depth source*).
- Fuente infrarroja (*Infrared source*).
- Fuente de índice de cuerpo (*Body Index source*).
- Fuente de cuerpo para esqueleto, reconocimiento de mano, apoyo, etc. (*Body source skeleton, hand tracking, lean*)
- Fuente de Audio (*Audio Source*).

- **Lector**

Provee el acceso a los *frames*, (eventos, y solicitudes), múltiples lecturas pueden ser creadas por cada una de las fuentes y el lector puede ser pausado.

- **Referencia de Frame**

- Acceso al actual *frame* en base al método de adquisición de *frame*.
- El *frame* contiene el metadata, que son los datos de reconocimiento obtenidos por el *skeleton tracking* (i.e, para color: formato, ancho, alto).
- Debe ser manejado rápidamente y luego removido (si un *frame* no es removido el siguiente *frame* no puede ser captado).

- **Acceso al Frame**

Provee el acceso a los datos como un buffer directo y una copia local obtenida del *frame*. Es posible utilizar el conjunto *MultiSourceFrameReade* el cual permite seleccionar un conjunto de *frames* de multiples fuentes en un mismo evento, los frames son entregados a una baja tasa de fps (*frames* por segundo) de las fuentes seleccionadas.

Para que el dispositivo funcione correctamente en la computadora que controlará el dispositivo Kinect es necesario instalar complementos llamados SDK, que es un conjunto de herramientas de desarrollo de software que permite emular softwares desarrollados para otras plataformas, actualmente existen 2 versiones de SDK, se detallan a continuación las diferencias entre estas. Para el proyecto se empleará el SDK 2.0.

2.2.6 Incorporación de SDK 2.0 respecto de SDK 1.8

Las diferencias entre versiones de SDK se detallan a continuación:

- Es el soporte para la nueva versión.
- Alta gama de visión horizontal y vertical en color.
- Ofrece *High Definition* (HD) en las imágenes a color.

- Visión infrarroja a 30 a una tasa de fps.
- Mayor fidelidad en profundidad respecto a la anterior versión.
- Micrófonos con mayor sensibilidad para un reconocimiento preciso.
- Posee 25 articulaciones para un máximo de 6 personas.
- *Tracking* completo de mano, se diferencia pulgar, mano, gestos de mano abierta y cerrada.
- Rango de operación en gama alta.
- Es posible utilizar varias aplicaciones a la vez con el sensor. (Ruben, 2015)

2.2.7 Forma de Captura

A continuación se detallan las siguientes etapas que sigue el dispositivo Kinect para la captura de datos:

- Un foco de luz ilumina a la persona reflejando la luz al sensor.
- El chip del sensor calcula la distancia desde el tiempo de salida y llegada de la luz, por pixel.
- Un software de imagen basado en profundidad capta e identifica los objetos en tiempo real.
- El dispositivo final procesa la señal. (Ruben, 2015)

2.2.8 Fuentes de Datos de Kinect v2

- **Color**

Para la captura de imágenes a color se tiene las siguientes características:

- Arreglo de color 1920x1080 a una tasa de 30 o 15 fps, basado a las condiciones de luminosidad.
- Formato de imagen puede ser: RGBA, BGRA, YUY2(formato crudo).
- Los datos de frame pueden ser: de uso en formato crudo o convertido a otros formatos (con un costo computacional).
- El Buffer es un arreglo de bytes.
- El número de bytes por pixel depende del formato crudo (4 bytes por pixel).

En la figura 5 se muestra una imagen capturada con las fuentes de color del dispositivo Kinect:

Figura 5 Imagen a Color

Fuente: (Oliveto, 2015)

- **Fuente Infrarroja**

Para la captura de imágenes en infrarrojo se tiene las siguientes características:

- 512x424 pixeles a 30 fps.
- Mismo sensor físico de la fuente de profundidad.
- Dos fuentes: *Infrared*- único *frame* infrarrojo y *LongExposureInfrared*- sobrelapamiento de 3 *frames* (mejor relación señal/ruido pero imágenes más borrosas).
- Cada pixel está compuesto por 2 bytes (16-bit) y representa el valor de intensidad IR.
- Ambiente de luz removida: el SDK posee solo la reflexión de la luz infraroja, proyectada por el dispositivo.

En la figura 6 se muestra una imagen capturada con fuente infrarroja del dispositivo Kinect:

Figura 6 Imagen Infrarrojo

Fuente: (Oliveto, 2015)

- **Fuente de Profundidad**

Para la captura de imágenes en profundidad se tiene las siguientes características:

- 512x424 pixeles a 30 fps.
- Rango: 0.5- 4.5 metros (Puede extenderse hasta 8m).
- Cada pixel está compuesto por 2 bytes (16-bits) y contiene la distancia en milímetros desde el sensor al punto focal del plano.

En la figura 7 se muestra una imagen capturada con fuente de profundidad del dispositivo Kinect:

Figura 7 Imagen de Profundidad

Fuente: (Oliveto, 2015)

- **Fuente de Índice de Cuerpo**

Las fuentes de índice de cuerpo tienen las siguientes características:

- 512x424 pixeles a 30 fps.
- Cada pixel esta compuesto por 1 byte

En la figura 8 se detallan los datos de Pixel, y se referencian como:

(a) 0-5: Índice del cuerpo correspondiente, como *tracked* por la fuente de cuerpo
(b) >5: El cuerpo no se encuentra reconocido en ese pixel.

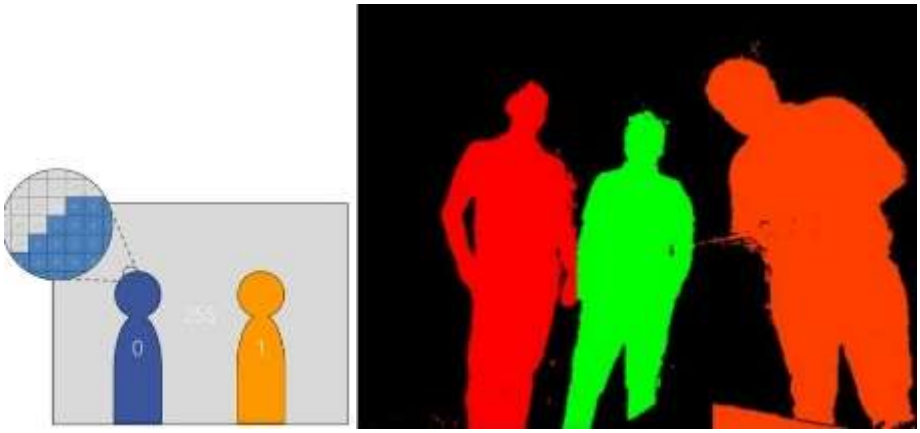


Figura 8 Índice de cuerpo

Fuente: (Oliveto, 2015)

- **Fuente de Cuerpo**

Las fuentes de cuerpo tienen las siguientes características:

- Rango es 0.5-4.5 metros a 30 fps.
- Los datos de *frame* es una colección de los objetos del cuerpo.

Cada cuerpo tiene:

- 25 puntos (cada punto tiene una posición 3D).
- El *hand tracking* puede ser abierto, cerrado o lazo.
- *Face tracking* y expresiones.
- Orientación de esqueleto.
- Hasta 6 cuerpos simultáneos.

- El estado de la mano hasta 2 cuerpos.

En la figura 9 se muestra una imagen con el reconocimiento de cuerpo proporcionado por los sensores del dispositivo Kinect:

Figura 9 Reconocimiento de Cuerpo

Fuente: (Oliveto, 2015)

2.2.9 Información del Cuerpo

La clase del cuerpo contiene propiedades útiles, la figura 10 muestra la información del *skeleton tracking* y se detallan como:

- `ClippedEdges`: contiene los bordes del campo de visión del cuerpo.
- `HandState`: representa el estado de la mano, izquierda/derecha como: (Desconocida, no reconocida, cerrada, abierta, lazo).
- `IsRestricted`: si se encuentra un objeto cerca, valor binario.
- `IsTracked`: si se encuentra la persona reconocida, su valor es binario.
- `TrackingId`: ID de 64-bits único.
- `Joints`: Muestra la posición en el espacio de cada punto.
- `JointOrientations`: detalla la orientación en el espacio de la articulación.
- `Lean`: Identifica la inclinación del vector de cuerpo.
- `LeanTrackingState`: estado de inclinación puede ser: (inferido, no reconocido, reconocido).

Figura 10 Orientación de las articulaciones Kinect.

Fuente: (López, 2015)

2.2.10 Escenarios de uso de Kinect v2

El dispositivo Kinect v2 suele ser empleado en varios campos en la vida de la gente, entre ellos se listan los siguientes:

- Ventas al por mayor.
- Terapia.
- Cuidado de la salud.
- Educación.
- Entrenamiento. (López, 2015)

2.3 Modelos de Detección

Los modelos de detección son técnicas que utilizan algoritmos para identificar en una imagen partes del cuerpo como brazos, manos, etc. A continuación se describen dichos modelos de detección aplicados al reconocimiento de signos.

2.3.1 Modelos de clasificación para el reconocimiento de signos

En el reconocimiento de lenguaje de signos, es fundamental una recolección de datos y extracción de características. Muchos de los actuales modelos de reconocimiento de lenguaje de signos se realizan en base a guantes o acelerómetros para reconocimiento de manos como en (Estrada Jimenez, 2016). Considerando que es más natural el uso de la visión para el reconocimiento de lenguaje de signos debido que no es necesario el uso de sensores adicionales, se puede realizar la extracción de características en base a imágenes de color y profundidad, información relacional y puntos de interés. Los modelos de reconocimiento utilizan diversos algoritmos para la extracción de datos, uno de ellos es la extracción de características de profundidad (Keskin, Kirac, Kara, & Akarun, 2013), las muestras utilizadas en estos algoritmos emplean una imágenes de profundidad basadas en tiempo real utilizando posiciones del esqueleto para la mano usando Kinect y el Lenguaje de Signo Americano (del inglés *American Sign Language*- ASL) para la aplicación de reconocimiento. (Zafrulla, Brashear, Thad, Hamilton, & Peter, 2011) Investigó la aplicación de Kinect en base al mapa de profundidad de la cámara para un reconocimiento de lenguaje de signos concluyendo que Kinect puede ser una opción viable para el reconocimiento de signos.

Un modelo de clasificación es importante en reconocimiento de lenguaje de signos el cual determina como usar un bajo nivel de características para describir un signo según sea el caso. Un modelo de trabajo utilizó la técnica red neuronal artificial (del inglés *Artificial Neuronal Net*- ANNs) (Murakami & Taguchi, 1991).

En el trabajo de (Wang, Lan, & Mori, 2011) se presentó un método general de extracción de trayectorias de posición y uso de ellos con un tiempo de retardo para una red neural (de inglés *Time Delay Neural Net*- TDNN) en el reconocimiento de lenguaje de signos. Derivado desde un discurso de reconocimiento automático, se utilizaron métodos como Modelos ocultos de Markov (del inglés *Hidden Markov Models*- HMMs) para realizar el procesamiento de los datos para el reconocimiento de lenguaje de signos a nivel-mundial, el cual se basa en los estados de selección en el modelo HMM (Vogler & Metaxas, 1997). El rendimiento depende del problema tratado como gesto de reconocimiento y son enfocados en modelos estáticos de signos. Muchas mejoras se han

dado para el problema de reconocimiento y enfocado a modelos estáticos (Tianzhu, Jing, Si, Changsheng , & Hanqing , 2009-2011).

Un ejemplo de los modelos de clasificación para la extracción de características utilizando Kinect se encuentra (Yauri Vidalón & De Martino, 2014) el cual utiliza el reconocimiento de señas de forma aislada para un posterior reconocimiento de ocurrencia dentro de las oraciones. Los datos utilizan el modelo HMM. El algoritmo de reconocimiento se muestra en la figura 11. Se puede observar que después del procesamiento de imágenes se utiliza el modelo HMM para las señas y oraciones para una traducción del modelo gramatical, en este caso LIBRAS (Lenguaje Signos Brasileño).

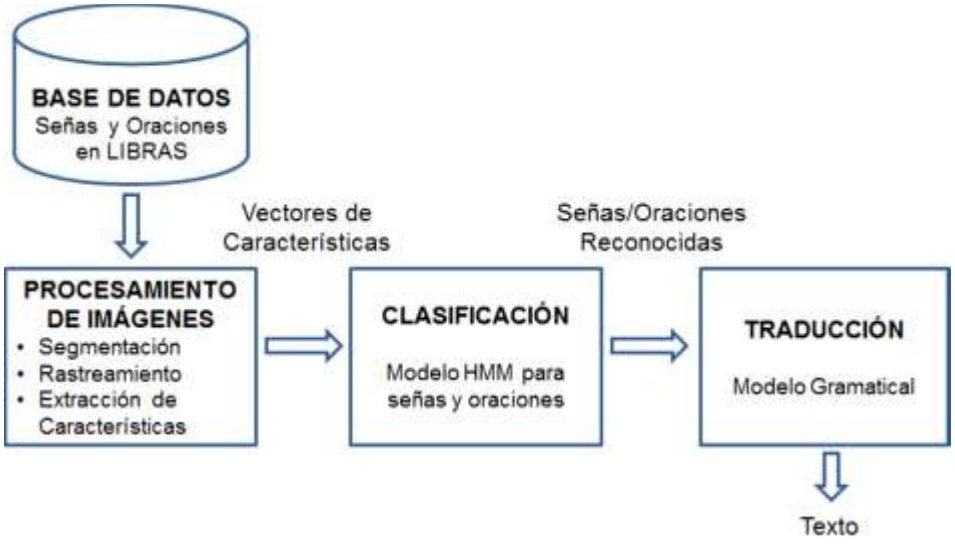


Figura 11 Diagrama de flujo para el reconocimiento de lenguaje de signos brasileño

Fuente: (Yauri Vidalón & De Martino, 2014)

Actualmente existen nuevos métodos de clasificación los cuales han demostrado una alta confiabilidad para el reconocimiento de lenguaje de signos, conocidos como Maquinas de Soporte Vectorial SVM (del inglés *Suport Vector Machine*) las cuales han sido satisfactoriamente aplicadas para varias aplicaciones como (Felzenswalb, McAllester, & Ramanan, 2010). Una ventaja de utilizar SVM y sus variantes es que permite una supervisión de las partes principales con un elemento a ser reconocido.

En (Sun, Zhang, Bao, Xu , & Mei, 2013) se introduce un modelo llamado *discriminative exemplary coding* (DEC) para aprovechar el modelo de lenguaje de signos en base al video con una característica de profundidad de Kinect. No existe un lenguaje de signos a nivel universal por lo que se debe establecer el país o región en donde se desarrollará la interpretación de palabras en signos para su respectiva captura de datos, ya que regionalmente existen diferencias entre los signos de lenguaje en el que se desempeñan, como ASL entre otros, es necesario considerar que las posiciones de las manos cambian de país en país o región en región.

En el trabajo anterior debido a la gran cantidad de palabras reconocidas utilizando el clasificador SVM se decidió utilizar el mismo para la validación de la base de datos a desarrollar, a continuación, se da una explicación del funcionamiento de las Máquinas de Vectores de Soporte.

2.3.2 Máquina de vectores de soporte (SVM)

Una máquina de vectores de soporte (Abe, 2010) (del inglés *Support Vector Machine*- SVM) es un método de aprendizaje supervisado que, a partir de ciertas muestras de entrada, es capaz de reconocer patrones permitiendo resolver problemas de clasificación y regresión. Utilizando SVM se puede llegar a identificar manos (posición dedos, etc.), en este caso se usará este modelo para clasificar las características que entrega el dispositivo Kinect v2 en su *skeleton tracking*, para reconocer, los movimientos y las articulaciones de interés, el cual se trata de un problema de clasificación de dos clases: una la cual es la clase asociada a todas las anteriores y otra a todo aquello que no lo sea.

Existen diversas formas para clasificación de dos clases como se muestra en la figura 12. Cuando se tiene un conjunto de muestras iniciales positivas o negativas (en este caso palabras que tengan diferente significado), con `n` características cada uno, dicha muestra se colocará como un punto en el espacio n-dimensional. El objetivo es encontrar, si es posible, una línea o hiperplano que separe los puntos de ambas clases en dos grupos, procurando que la distancia entre puntos de las mismas sea un máximo posible.

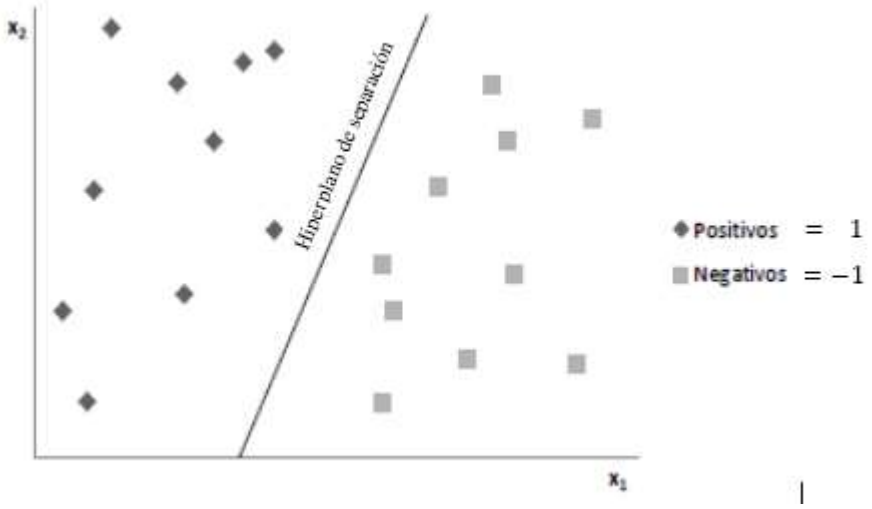


Figura 12 Hiperplano con X_1, X_2 como planos de separación.

Fuente: (Arranz, Liu Yin, & López Cámara, 2012)

Su expresión matemática es basada a partir de una entrada compuesta por M muestras de entrenamiento $x_i \epsilon\ \mathbb{R}^n$ $(i = 1, \dots . M)$ pertenecientes a las clases C_1 o C_2 y las etiquetas asociadas $y_i \in \{1, -1\}$, siendo $y_i = 1$ si x_i pertenece a la clase C_1 y $y_i = -1$ si pertenece a la clase C_2, se dice que las M muestras son $linealmente\ separables$ si es posible determinar una función de decisión como:

$$D(x) = w \cdot x + b \tag{2.1}$$

Donde $w\ \epsilon\ \mathbb{R}^n$, $b\ \in\ \mathbb{R}$ y se cumple que para $i = 1, \dots ., M$

$$w \cdot x_i + b \begin{cases} > 0 & para\ y_i = 1 \\ < 0 & para\ y_i = -1 \end{cases} \tag{2.2}$$

Suponiendo que los datos de entrada son linealmente separables, ninguna entrada x_i satisface $w \cdot x_i + b = 0$. Para lograr la separabilidad para cada clase y las muestras según su etiqueta, se sustituyen las desigualdades (2.2) por:

$$w \cdot x_i + b \begin{cases} \geq\ 1 & para\ y_i = \ 1 \\ \leq -1 & para\ y_i = -1 \end{cases} \tag{2.3}$$

Simplificando las ecuaciones de desigualdad:

$$y_i(w \cdot x_i + b) \geq 1 \ \ para\ i = 1, \dots ., M \tag{2.4}$$

La ecuación 2.5

$$D(x) = w \cdot x + b = c\ para - 1 < c < 1 \quad (2.5)$$

Genera un hiperplano de separación entre datos de entrada correspondientes a x_i. Si $c = 0$, el hiperplano de separación se encuentra en medio de los hiperplanos con $c = 1\ y\ c = -1$. Se denomina $margen$ a la distancia existente entre el hiperplano de separación y la de muestra de entrenamiento más cerca al mismo, esto se observa en la figura 13. (Arranz, Liu Yin, & López Cámara, 2012).

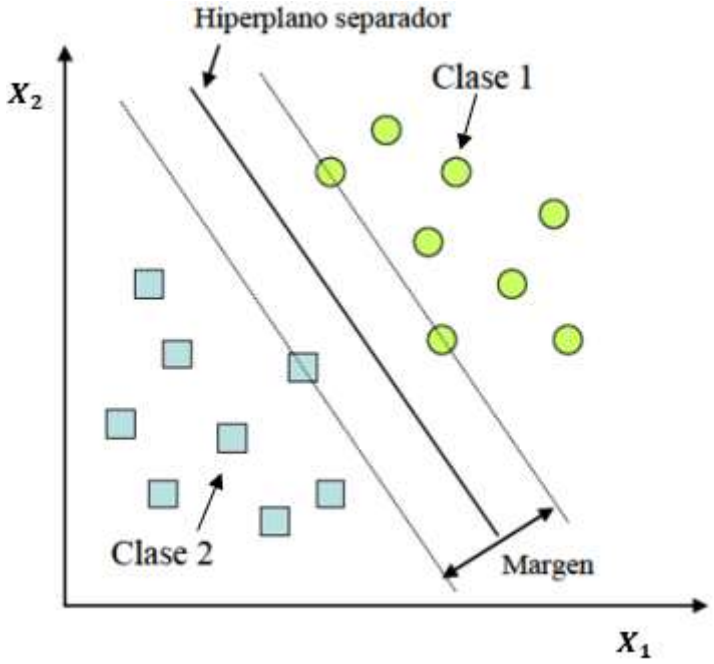


Figura 13 Hiperplano 2 con X_1, X_2 como planos de separación.

Fuente: (Otero, 2012)

Los datos de la figura 13 representan datos de diferentes clases separados por el hiperplano que maximiza la distancia entre ellos. Dicha distancia marcada como margen, es máxima para un hiperplano obtenido, para cualquier otro presentaría un margen de separación de clases menor. (Otero, 2012)

Para una mayor facilidad en la resolución de problema usando Máquina de Soporte Vectorial es necesario que los datos sean linealmente separables, no siempre se

podrá tener este tipo de entradas por lo que esto genera un cierto nivel de ruido. Existen dos tipos de SVM de clasificación binaria los cuales son:

- SVM lineal con margen máximo
- SVM para la clasificación no lineal

Solo se hablará del SVM lineal con margen máximo ya que es el utilizado en este trabajo, el cual se detalla a continuación.

2.3.3 SVM lineal con margen máximo

Las máquinas de soporte vectorial está formado de hiperplanos separando datos de entrada como un par de subgrupos con una etiqueta propia si son linealmente separables, dicho de otro modo, en medio de todos los posibles planos de separación de las dos clases que se etiquetan como {-1, +1}, existe únicamente un hiperplano de separación óptimo, de tal forma que suceda la maximización del margen y en donde se apoyan puntos denominados vectores de soporte (Colmenares, 2010), tal y como se observa en la figura 14. Cuando los datos son:

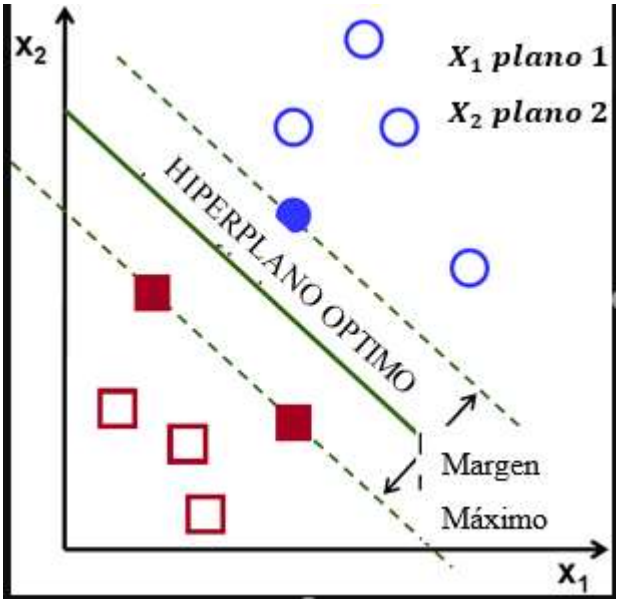


Figura 14 SVM linealmente separable.

Fuente: (Zaforas, 2017)

Este caso solo se debería emplear cuando los datos son linealmente separables de tal manera que se cumpla la ecuación previamente obtenida:

$$y_i(w \cdot x_i + b) \geq 1 \quad para\ i = 1, \ldots\ldots, M \tag{2.6}$$

Con el hiperplano $D(x) = w \cdot x + b = h(x) = c\ para - 1 < c < 1$

La distancia $dist(h, x)$ de un punto al hiperplano es:

$$dist(h, x) = \frac{|h(x)|}{\|\omega\|} \tag{2.7}$$

Como los puntos más próximos al hiperplano cumplen $|h(x)| = 1$ para $y_i = 1$ o $y_i = -1$, su distancia al hiperplano seria:

$$dist(h, x) = \frac{1}{\|\omega\|} \tag{2.8}$$

Entonces para hallar los valores de ω y b hay que resolver un problema de optimización que consiste básicamente en maximizar la distancia $dist(h, x)$ entre el hiperplano y el punto de entrenamiento más próximo, entonces:

El máximo de:

$$\frac{1}{\|\omega\|} \tag{2.9}$$

Sujeto a:

$$yi\ (\omega \cdot x_i + b) \geq 1, i = 1, \ldots n \tag{2.10}$$

La cual es la condición que indica que ningún vector de entrenamiento debe quedar dentro del margen que separa a las dos clases. La figura 15 muestra las diferentes situaciones con los elementos característicos (hiperplano y el margen máximo), los círculos de color verde y cuadros de color azul, representan elementos fuera del hiperplano de separación y margen máximo, los de color negro muestran los vectores de soporte de la máquina entrenada.

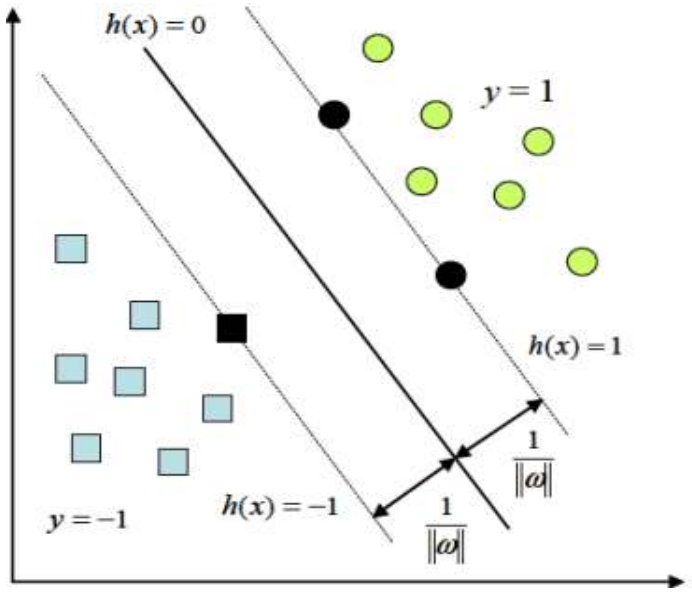


Figura 15 SVM con margen máximo (en negro los vectores de soporte).

Fuente: (Otero, 2012).

La ecuación del hiperplano depende únicamente de los vectores soporte, del cual surge el nombre del algoritmo, máquinas de soporte vectorial.

2.3.4 Técnica de extracción de características.

La detección y clasificación de objetos debe ser distinguida dentro de la adquisición de imágenes, existen distintos tipos de cámaras, dotadas de diversos sensores, con el fin de lograr captar la información en tiempo estimado o real. Existen varios métodos que permiten la detección de objetos, personas y análisis de las imágenes, como pueden ser detectores de borde, histograma de gradientes orientados (HOG), flujo óptico (OP), que puede describir la apariencia e información de movimiento. (Arranz, Liu Yin, & López Cámara, 2012)

Basado en los datos de salida del Kinect, es posible conocer la posición de las manos y obtener su información de estado y características de movimiento. (Zhang, Xu, & Sun, 2015).

Se utilizó las características propias de Kinect para la extracción de características en base a la estimación que este ofrece para el movimiento de las manos,

brazos y hombros. En este caso se extrajeron los datos de salida de esqueleto en esos puntos, considerando:

- Los vectores unitarios de los codos con respecto a los hombros, las muñecas con respecto a los codos, las manos con respecto a las muñecas y la mano izquierda con respecto a la mano derecha.
- Los ángulos de articulación de los hombros, los codos y las muñecas.
- La distancia entre la mano derecha, mano izquierda, normalizados y divididos dos veces por el tamaño de los hombros.

Para una mayor precisión para un futuro reconocimiento es posible realizar una combinación de extracción de características ordinarias utilizando la extracción de descriptores HOG y OP con las características propias de Kinect (videos de color y videos de profundidad).

2.4 MATLAB

Matlab es un entorno software que mediante análisis interactivo y procesos de diseño tiene como propósito el desarrollo integrado de aplicaciones. (MathWorks, Matlab Products, 1994). A continuación se explica sobre las características de MATLAB

2.4.1 Características principales de MATLAB

Este programa dispone de gran cantidad de características como es el trabajo con matrices, datos, funciones, algoritmos, editor de interfaces gráficas de usuario y plataformas de simulación multidominio, capacidad de asociarse con programas de diferente tipo de lenguaje. Por último, mediante *toolboxes* de diferentes tipos puede aumentar sus capacidades y prestaciones.

Entre los diversos recursos que posee Matlab los principales son:

- Capacidad de escritura en lenguaje matemático.
- Capacidad de implementación de matrices permitiendo reducción de líneas de código.
- Capacidad de implementación de aritmética compleja.

- Herramientas *toolboxes* que permiten comunicación con otros elementos hardware.
- Ampliación y adaptación a través de ficheros de script y funciones .m.

Gracias a estos recursos la herramienta a Matlab es empleado bastante en el campo de la ingeniería para análisis de datos, aprendizaje profundo, procesamiento de señales, Finanzas Cuantitativas y Gestión de Riesgos, Robótica, Sistemas de control y Comunicaciones Inalámbricas, entre otros.

2.4.2 Toolboxes de MATLAB

Los *toolboxes* son programas que permiten extender las capacidades de MATLAB, brindando funciones adicionales a tareas y aplicaciones específicas, como comunicación con otros módulos, algoritmos adicionales y aplicaciones interactivas. Entre ellas existen aplicaciones destinadas para Kinect como es *Image Adquisition Tool* (MathWorks, Acquire images and video from industry-standard hardware, 2015), que brinda una interfaz gráfica desplegando las diferentes funciones que dispone el dispositivo y añade las librerías necesarias para la elaboración de un programa para la captura de datos de video y características del *skeleton tracking* dentro de la base de datos.

Otra aplicación es el *Classification Learner* (MathWorks, Acquire images and video from industry-standard hardware, 2015), el cual permite realizar capacitaciones automatizadas para buscar el mejor tipo de modelo de clasificación, incluidos árboles de decisión, análisis discriminante y máquinas de vectores de soporte (SVM). Como se mencionó con anterioridad por el desarrollo de trabajos previos se opta por SVM para la clasificación de características de los datos obtenidos por Kinect v2 en la base de datos de ciertas palabras de uso común, validando la misma al realizar un reconocimiento de signos, este proceso se explicará con mayor detalle en el capítulo 5.

2.4.3 Image Acquisition Tool

Es una herramienta que sirve para la conexión entre una aplicación de escritorio y un scrip enlazados directamente a un *hardware*. Las aplicaciones son desarrolladas en un lenguaje de programación propio, que son interpretadas, y ejecutadas en un entorno interactivo mediante un archivo de script (*.m). (MathWorks, Acquire images and video from industry-standard hardware, 2015)

Image Acquisition Tool permite adquirir imágenes usando una Kinect para Windows, lo cual es usualmente empleado para aplicaciones de robots, interacciones computadora humano, seguridad de sistemas, entretenimiento, diseño de juegos y el campo de la ingeniería. Entre los usos que suele darse son análisis del cuerpo, mapeo 3-D, reconocimiento de gestos y patrones humanos. La instalación de esta herramienta se explicará con mayor detalle en el capítulo 3

Image Acquisition Tool posee una interfaz como se muestra en la figura 16**,** que permite maniobrar de forma cómoda el dispositivo Kinect, obteniendo acceso directo a todas sus funciones, se utilizará este software debido a las características que posee para facilidad en la elaboración de un programa para captura de datos.

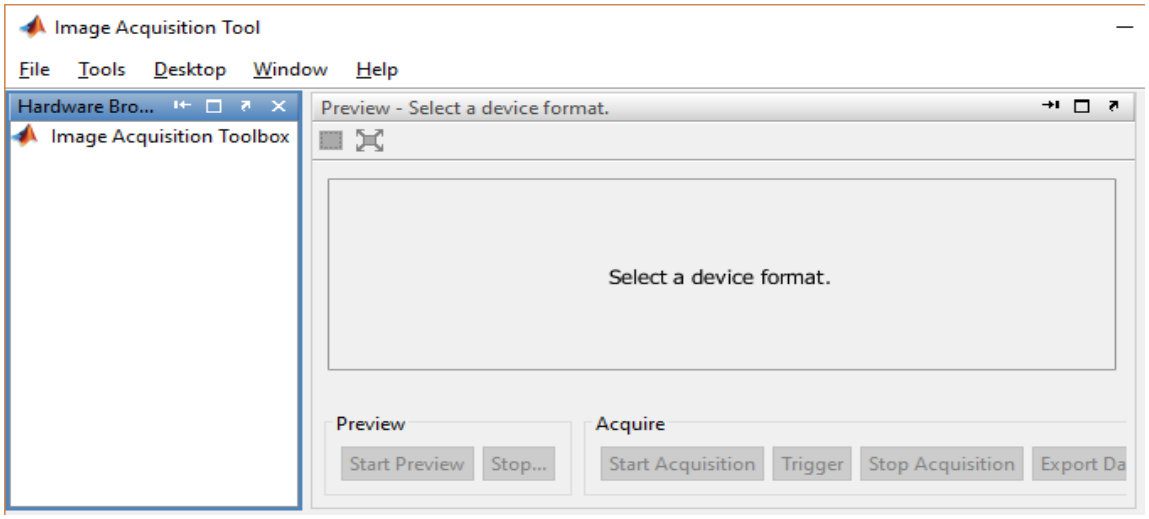


Figura 16 Interfaz Image Acquisition Tool

Fuente: (MathWorks, Acquire images and video from industry-standard hardware, 2015)

La razón del uso de *Image Acquisition Tool* para este trabajo, es el manejo de las funciones ofrecidas para la captura de videos en color, videos en profundidad y extracción de *skeleton tracking* mencionado anteriormente, en el capítulo 3 se detallará el programa desarrollado para extracción de características. Se explicarán los requerimientos de instalación para el *toolbox* mencionado

2.4.4 Requerimientos para Image Acquisition Tool

Dentro de los requerimientos para un funcionamiento correcto del dispositivo Kinect con Matlab es necesario: (MathWorks, Acquire images and video from industry-standard hardware, 2015)

- Windows de 64-bit.
- Tener Puerto USB 3.0 en el computador para el controlador de Kinect V2.
- Kinect for Windows Runtime version 2.0 instalado.
- Es posible conectar varios dispositivos Kinect V2 al mismo tiempo, pero solo es posible utilizar una función de Kinect V2 con *Image Acquisition Toolbox*.
- Es posible utilizar Kinect V2 y Kinect V1 al mismo tiempo, pero debido a limitaciones de hardware solo una función de Kinect V2.

2.5 Classification Learner de MATLAB

Classification Learner es una aplicación que entrena modelos para clasificar datos. Entre las diferentes opciones que brinda existan: (Mathworks, Classification Learner, 2016)

- Aprendizaje automático supervisado utilizando varios clasificadores.
- Exploración de datos, selección de características para especificar esquemas de validación, entrenamiento modelos y evaluación de resultados.
- Formación automatizada para buscar el mejor tipo de modelo de clasificación, incluidos los árboles de decisión, el análisis discriminante, las máquinas de

vectores de soporte, la regresión logística, los vecinos más cercanos y la clasificación de conjuntos.

- Aprendizaje automático supervisado suministrando un conjunto conocido de datos de entrada (observaciones o ejemplos) y respuestas conocidas a los datos. Mediante estos datos se logra entrenar un modelo que genera predicciones para la respuesta a nuevos datos.

Se explica el uso de Classification Learner para el presente trabajo en el capítulo 4

A continuación, se muestra la interfaz de la aplicación en la figura 17:

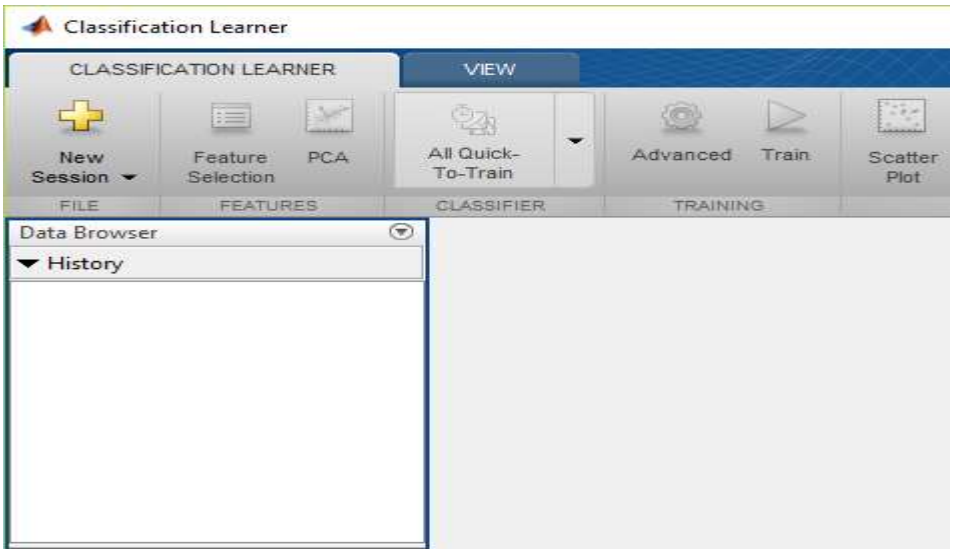


Figura 17 Interfaz Classification Learner

Fuente: (Mathworks, Classification Learner, 2016)

2.5.1 Diferencias entre versiones Kinect para MATLAB

El empleo de la Kinect en Windows es similar a una cámara, sin embargo cuenta con algunas diferencias como: (MathWorks, Key Features and Differences in the Kinect V2 Support, 2017)

- El dispositivo Kinect de Windows posee 2 sensores físicos por separado. El sensor de color se encarga de la captura de datos de imágenes a color, y el

sensor de profundidad se encarga de datos de profundidad y detección del esqueleto como *frames*.

- Para Kinect, es posible acceder a los datos del cuerpo mediante el sensor de profundidad

En el caso de Kinect V1:

- Para Kinect V1, entrega 4 tipos de datos diferentes, como son datos de color, profundidad, esqueleto y audio, sin embargo este último no es captado por las herramientas de Matlab.
- Kinect V1 puede rastrear hasta 6 personas, verificando su posición y orientación hasta 4 personas.

En el caso de Kinect V2:

- Para Kinect V2, entrega 5 tipos de datos diferentes, como son datos de color en RGB_1920x1080, profundidad en DEPTH_512X424, cuerpo, índices del cuerpo y audio, sin embargo este último no es captado por las herramientas de Matlab.
- Kinect V2 puede rastrear hasta 6 personas con su posición y orientación, incluye el rastreo de manos y el estado del cuerpo.

La principal utilidad de estas diferencias para el presente trabajo es el manejo de los datos propios de Kinect V2 junto con las imágenes de color y profundidad, además que se añade el estado de la mano en tres diferentes tipos (abierta, cerrada, lazo) lo cual puede mejorar la predicción del signo a reconocer si se deseara realizar un reconocimiento más preciso, y es posible extraer mayor cantidad de datos dentro de las imágenes de profundidad y color al tener una mayor resolución que la versión 1.

Una ventaja de tener dichas resoluciones en RGB y en profundidad es que mejoran la captura de datos para la extracción de características en una distancia mayor a 1.5 metros, aumentando el alcance del estado del cuerpo y facilitando el manejo de datos del *skeleton tracking* el cual la versión 1 no logra detectar a una distancia mayor a 1.5 metros por sus parámetros de resolución.

El capítulo 3 explicará cómo se realizó la elaboración de la base de datos.

CAPITULO III
CREACIÓN DE LA BASE DE DATOS

Este capítulo se centra en la explicación y detalles de la creación base de datos que sirva para el reconocimiento de signos del lenguaje dactilológico español. Se empieza explicando la problemática, seguido del tipo de palabras que corresponderán a la base de datos, tipo, características y cantidad de muestras a tomadas, seguido de los procedimientos y programas necesarios para poder realizarlo. Posterior a ello se indica el tipo de codificación que tendrá la distribución de muestras de color y profundidad con sus respectivas características para un acceso más dinámico y automatizado.

3.1 Elementos de la Base de datos

Para la base de datos se contará tanto de palabras (saludos, alimentos, juguetes&cosas, colores, adjetivos) como de letras del abecedario (A-Z) dactilológico español ecuatoriano. Que fueron recomendados por el Instituto Nacional de Audición y Lenguaje (INAL), los mismos que establecieron que signo realizar, junto con la ayuda de 2 docentes especializados y 23 estudiantes para la elaboración de los mismos. Se explica a continuación cada parte de la base de datos:

3.1.1 Palabras de la Base de datos

La base de datos consiste de 5 tipos de palabras de uso común del lenguaje dactilológico ecuatoriano, como son: adjetivos, alimentos, saludos, juguetes & cosas y colores. Estas palabras fueron establecidas por el Instituto Nacional de Audición y Lenguaje (INAL), tomando en cuenta que son de uso común, sencillas y conocidas por la mayoría de personas con discapacidad auditiva.

La base consiste de 10 palabras diferentes para cada uno de los tipos en total son 50 palabras, como se muestran en la tabla 1:

Tabla 1

Palabras de la Base de Datos

Adjetivos	Alimentos	Saludos	Colores	Juguetes & Cosas
Alto	Huevo	Buenos días	Amarillo	Avión
Bajo	Leche	Buenas noches	Azul	Barco
Bonito	Limón	Buenas tardes	Blanco	Carro
Feo	Manzana	Chao	Café	Casa
Grande	Naranja	Como estas	Gris	Mesa
Limpio	Pan	Disculpa	Morado	Muñeco
Nuevo	Papaya	Gracias	Negro	Pelota
Pequeño	Pera	Hola	Rojo	Reloj
Sucio	Plátano	Mucho gusto	Rosado	Silla
Viejo	Sandia	Por favor	Verde	Tren

3.1.2 Letras de la Base de datos

La base de datos cuenta con muestras de abecedario completo del lenguaje dactilológico español, los cuales están compuestos de las letras indicadas en la tabla 2:

Tabla 2

Tabla de letras del Abecedario Español de la base de datos

Letras					
A	E	J	N	R	V
B	F	K	Ñ	RR	W
C	G	L	O	S	X
CH	H	LL	P	T	Y
D	I	M	Q	U	Z

La base de datos, está compuesta por letras estáticas que se mantienen constante sin cambio o movimientos adicionales, y letras dinámicas que son “ll”, “rr”, “ch”, “j” y “z”, estas letras son consideradas como tales debido que cuentan con movimiento en su ejecución.

Se realizó un programa que se explicará más adelante que permitió tomar los datos, y para realizar los mismos es necesario utilizar el dispositivo Kinect V2.0 por lo que se explicará como se realiza la obtención de datos.

3.2 Kinect v2 de Xbox One

Para la obtención de datos se optó por trabajar con Kinect v2.0 sobre Kinect v1.0 por las siguientes razones:

- **Características de rastreo de *skeleton tracking* a partir de video de profundidad**

Kinect v2.0 tiene mejores capacidades de resolución, rastreo de persona, rastreo de esqueleto, rastreo de manos, y estados de la mano a partir de videos de profundidad, lo cual en comparación a su versión anterior únicamente posee detección solo de la palma de la mano. Esta capacidad de detección manos es fundamental al momento de trabajar con lenguaje dactilológico que prioriza los signos formados a través de gestos manuales.

- **Mayor Precisión en la detección de datos**

Kinect v2.0 rastrea de forma más precisa el esqueleto de la persona lo cual facilita la diferenciación de la dirección de los dedos de la mano dentro del *skeleton tracking*. Estos datos son importantes al momento de emplear la aplicación *Classification Learner*.

3.3 Adquisición de datos

3.3.1 Instalación del paquete de compatibilidad de Kinect

Para la adquisición de datos, es necesario establecer como primer paso la comunicación del PC con el dispositivo Kinect para que exista un correcto funcionamiento y manipulación del sensor desde el programa MATLAB, para ello se deberá instalar los paquetes requeridos para Kinect for Windows Sensor (Mathworks, Detect the Kinect V2 Devices, 2016), los pasos se describen a continuación:

- Para la instalación de los paquetes se ingresa en la ventana de comandos de MATLAB el comando `supportPackageInstaller`.

Con ello se desplegará una ventana como se muestra en la figura, en donde se selecciona la opción *Install from Internet* por recomendación, como se muestra en la figura 18.

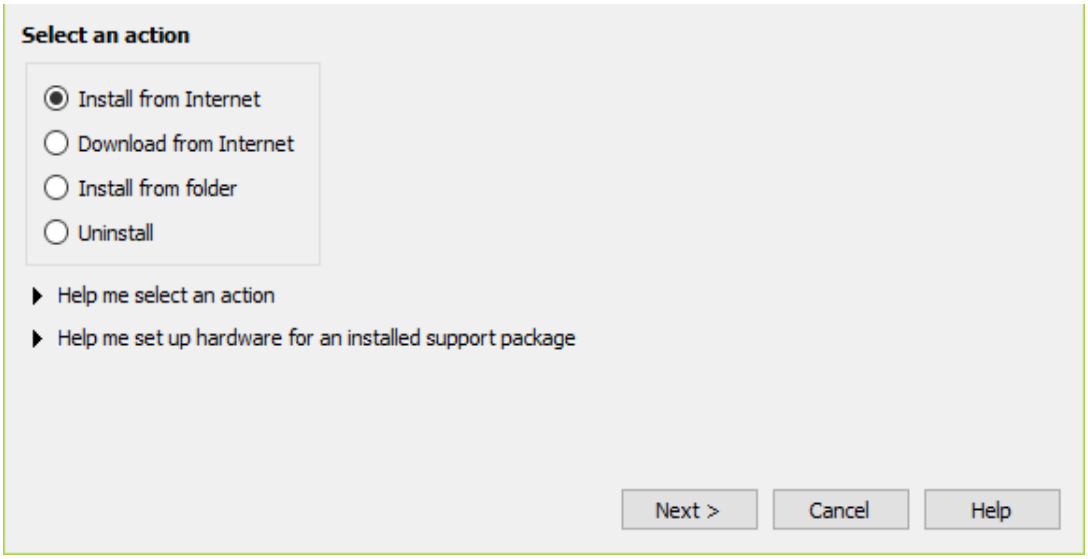


Figura 18 Instalación del paquete de soporte

Fuente: (MathWorks, Support Package Installation, 2015)

Como paso siguiente, se selecciona el paquete que se desea instalar, en este caso *Kinect for Windows Sensor*, como se muestra en la figura 19:

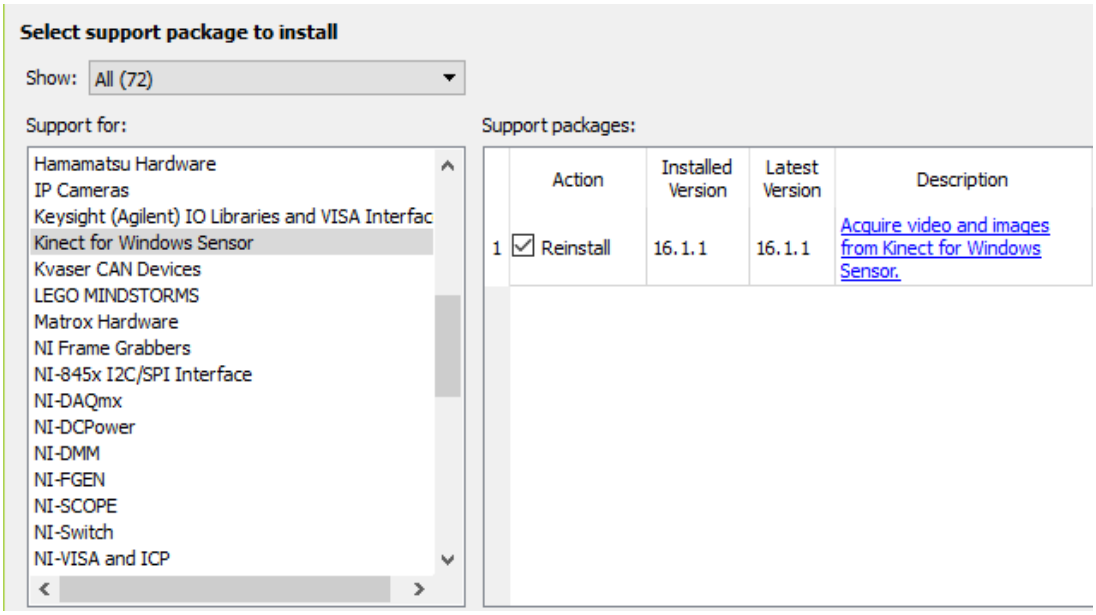


Figura 19 Instalación del paquete Kinect for Windows Sensor

Fuente: (MathWorks, Support Package Installation, 2015)

Seguido a esto, Matlab requerirá el ingreso de usuario y contraseña de una cuenta de MATLAB, para descargar e instalar el paquete seleccionado (Si no se posee una cuenta de MATLAB, es necesario crear una nueva de forma gratuita).

Una vez instalado se ingresa en la ventana de comandos de MATLAB con el comando `imaqhwinfo` para verificar la correcta instalación del paquete. A continuación, se muestra el resultado que despliega el comando, indicando las características del conector instalado que en este caso es el dispositivo Kinect:

```
          InstalledAdaptors: {'kinect'} %indica el adaptador instalado
              MATLABVersion: '9.0 (R2016a)' %indica la versión de Matlab en
este caso la versión 2016a
                ToolboxName: 'Image Acquisition Toolbox' %Indica el nombre del
toolbox utilizado para esta aplicación
             ToolboxVersion: '5.0 (R2016a)' %Indica la versión del toolbox
para la versión de Matlab en este caso 5 para la versión 2016a.
```

La versión 2016a de Matlab cuenta con el *toolbox Classification Learner* incluido dentro del paquete de instalación, por lo que no es necesario volverlo a instalar desde el internet.

3.3.2 Período de toma de datos

Para la base de datos se realizaron grabaciones en el Instituto Nacional de Audición y Lenguaje, divididas en sesiones de 4 horas diarias por 3 semanas con un total de 27 personas diferentes, entre ellas 2 instructores del lenguaje dactilológico, 23 estudiantes de capacidades especiales que dominan el lenguaje, y 2 personas entrenadas para la comprobación de la base de datos, las grabaciones se realizaron tanto para las palabras de uso común como para las letras del abecedario español.

3.3.3 Proceso para la creación de la base de datos

Para realizar la base de datos se procedió primeramente con una solicitud realizada por el director de la carrera de ingeniería en electrónica y telecomunicaciones, el cual se detalla en el anexo 8.

Aprobada la solicitud para la captura de las muestras requeridas, se procedió a conocer la necesidad del instituto, el requisito fue realizar 50 palabras en video para diferentes categorías como adjetivos, saludos, colores, alimentos, juguetes & cosas.

Las palabras de la base de datos fueron tomadas dependiendo de la disponibilidad de estudiantes y maestros del Instituto Nacional de Audición y Lenguaje (INAL).

Se contó con el apoyo de 2 docentes especializados en el lenguaje de signos para el desarrollo de la base de datos junto con sus estudiantes dentro de las horas de clases,

por lo que se establecieron días y hora acorde a la disposición de los mismos. Establecidos los días respecto a la disponibilidad del docente, estudiantes, edades, problemas de discapacidad y aprendizaje, por la disponibilidad se procedió a capturar una mayor cantidad de datos con el Técnico Ivan Unda, conocido como persona 1 dentro de la base de datos.

Las sesiones con el primer docente se las realizó en un período de 4 horas diarias por 3 semanas y 20 estudiantes de edades de 16-22 años.

El segundo docente tenía mayores responsabilidades al encargarse de estudiantes de edades inferiores a los 14 años que tienen una tendencia a distraerse fácilmente, por lo cual se procedió a grabar a 5 de sus estudiantes. Estas sesiones se realizaron por 2 semanas, los días lunes y jueves en el horario de 9:30 – 10:30.

El número total de muestras tomadas fueron de 6 capturas de video de color RGB, video de profundidad y datos de *skeleton tracking* (metadata) por cada palabra. Cada uno de los videos RGB y profundidad están formados por 100 *frames*.

El conjunto de palabras “saludos” por ser los más comunes entre personas con discapacidades auditivas y las más sencillas dentro del aprendizaje para personas que no presentan esta discapacidad, se procedió a tomar una mayor cantidad de muestras dando un total de 18 repeticiones.

Se explicarán los factores que se debe tomar en cuenta para la captura de la base de datos.

3.3.4 Factores en la captura de datos

Se debe tomar en cuenta algunos factores previos a la captura de datos, los cuales son:

- Procurar que el ambiente donde se capturan los datos no sea muy iluminado con los rayos del sol, ya que evitan que el sensor de profundidad del dispositivo capture de manera adecuada el *skeleton tracking.*
- La distancia del dispositivo Kinect v2.0 respecto a la persona se tomo de 3 metros ya que se verificó que a esa distancia la captura del *skeleton tracking* es más precisa respecto de la imagen de color y profundidad.

- Verificar que el video de profundidad esté centrado respecto al video de color y que no existan personas alrededor del dispositivo para evitar detección de personas adicionales.

3.3.5 Características principales de las muestras

Para la elaboración de la base de datos se tomarán 3 tipos diferentes de muestras, que se guardarán en archivos. mat para que sean cargadas con el comando `Load` para que puedan ser usadas por el lenguaje MATLAB.

Video de escala de colores: Estos datos guardan el conjunto de *frames* como video a color de las grabaciones, en este trabajo se la conocerá a esta variable como `ColorImg` en formato 4-D *uint8*.

Video de profundidad: Estos datos guardan el conjunto de *frames* como el video de profundidad de las grabaciones, estas contienen las características de profundidad y altura, , en este trabajo se la conocerá a esta variable como `DepthImg` en formato 4-D *uint16*.

Datos de *Skeleton Tracking*: Estos parámetros son los datos de las partes del esqueleto humano que fueron detectadas y reconocidas por el dispositivo Kinect, en este trabajo se la conocerá a esta variable como `Metadata` en formato *struct* 100x1 como matriz de datos de tipo *double*.

3.3.6 Resumen de la base de datos

En la tabla 3 se muestra los datos generales de las personas que fueron grabadas para crear la base de datos y la tabla 4 muestra de forma resumida la cantidad de muestras obtenidas por palabras:

Tabla 3

Resumen de personas por género y edad

Género	N° Personas	Rango	Edad
Masculino	12	Min.	14
Femenino	13	Max.	45
Total	25		

Tabla 4

Resumen de número de muestras por tipo de palabra

Palabras	Adjetivos	Alimentos	Colores	Juguetes & Cosas	Saludos	Total
#Muestras	60	60	60	60	180	420

3.4 Descripción del programa de Adquisición de datos.

Como se mencionó en la sección 3.3.3 para el proceso de la creación de la base de datos, se desarrolló un programa en Matlab el cual junto con la herramienta de *Image Adquisition Tool* el cual se explicó en el capítulo 2, detallando a continuación las principales características del programa para extracción de características de los videos.

3.4.1 Detalles Principales

Para la obtención de muestras se realizó un programa en Matlab R2016a, el cual pueda capturar y guardar 3 datos diferentes para cada muestra que son videos en escala de color, video de profundidad y sus correspondientes datos de *Skeleton Tracking* (*Metadata*). A continuación se muestra la interfaz gráfica del programa en la figura 20 y sus respectivas características:

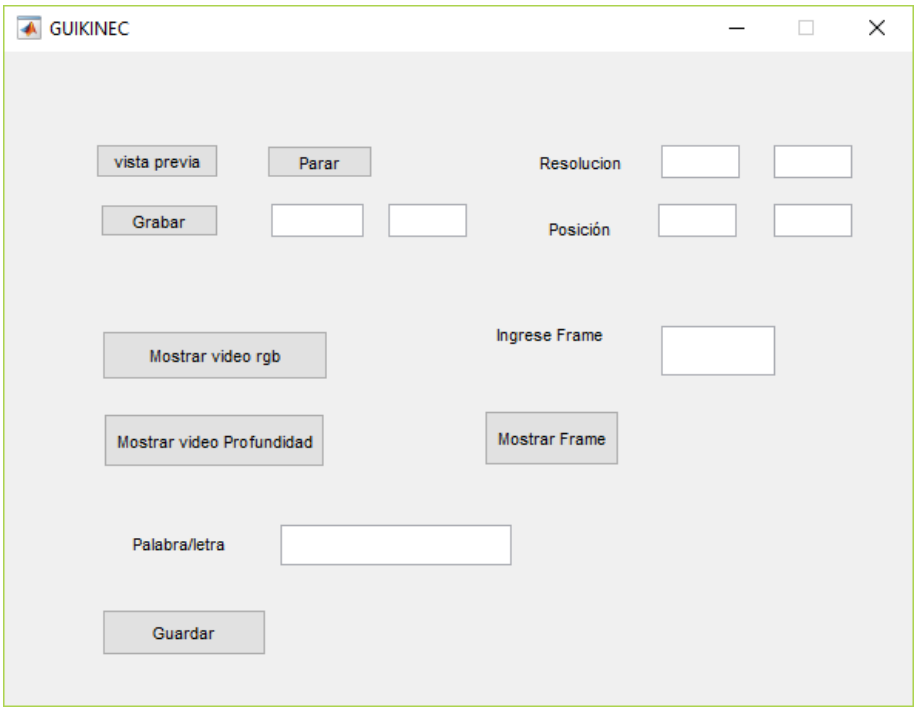


Figura 20 Interfaz Gráfica de Adquisición de Datos

Vista previa: Esta opción permite visualizar el video, ayudando posicionar a la persona ante la Kinect y su entorno para que no exista problemas al momento de la adquisición de datos. Existe también el botón parar para detener la visualización. Se muestra los botones en la figura 21:

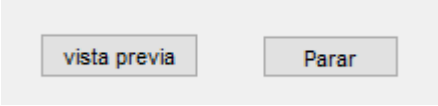


Figura 21 Vista Previa

Resolución: Esta opción permite reducir o aumento el enfoque del recuadro de la cámara del dispositivo Kinect para obtener solo muestras de interés que se requieran para la base de datos. Se muestra los recuadros en la figura 22:

Figura 22 Resolución

Posicionamiento de los sensores de la Kinect v2.0: La Kinect posee 2 sensores, los cuales no están completamente alineados a la cámara, por ello la función de posicionamiento permite realizar un enfoque preciso para que la toma de datos tenga mayor fiabilidad al mantener bajo las mismas condiciones la captura de datos. Se muestra los recuadros en la figura 23:

Figura 23 Posición

Grabar: Esta opción permite empezar a tomar las muestras que conforman el video, posee 2 recuadros que indican el tiempo cuando empiezan y termina la grabación para facilitar la toma de muestras. Se muestra los recuadros en la figura 24:

Figura 24 Botón Grabar

Mostrar de video RGB: Esta opción permite visualizar el video completo a color tomado de la captura de datos, esto permite revisar si se produjo algún error en la toma de datos, ya sea por debido al ambiente u objetos que obstruyan una visualización correcta durante las capturas de datos. El reproductor de video por defecto de MATLAB utilizado por Windows es *Movie Player* con el comando `implay(colorImg)`, el cual se explicará con mayor detalle en el capítulo 4. A continuación se muestra una captura de un video de color en la figura 25:

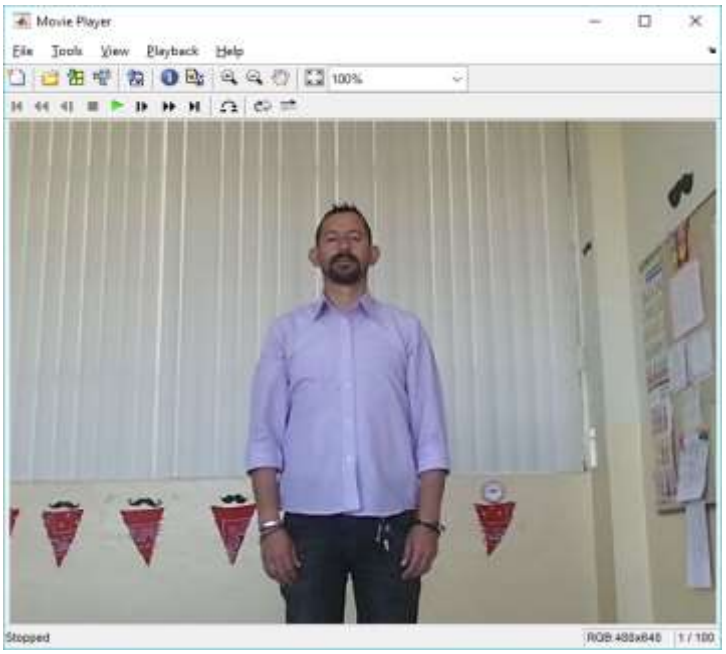


Figura 25 Movie Player de video de color

Mostrar video de profundidad: Esta opción permite visualizar el video completo de profundidad.

En cuanto a la resolución de videos a color y de profundidad no es la misma al ser capturados por distintos sensores de la Kinect, esto permite revisar si se tomaron erróneamente esqueletos que no pertenezcan a los datos de interés. El reproductor de video por defecto de MATLAB utilizado por Windows es *Movie Player* con el comando `implay(depthImg)` el cual se explicará con mayor detalle en el capítulo 4. Una captura del video de profundidad se muestra en la figura 26:

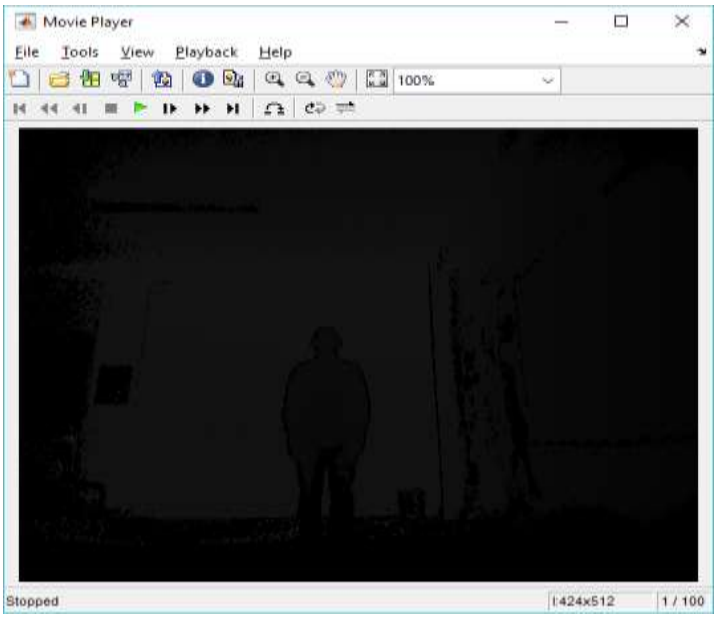


Figura 26 Movie Player video de Profundidad

Mostrar *Frame*: Esta opción permite la visualización de la posición de un *frame* en específico al ingresar el número en la secuencia de video, esto permite analizar uno a uno los datos tomados con sus respectivo *skeleton tracking*. Se muestra en la figura 27 una captura de un *frame* y su visualización dentro del programa:

Figura 27 Ventana Mostrar Frame y skeleton tracking

Guardar: Esta opción permite guardar tanto los datos de los videos a color y profundidad, y *skeleton tracking* de las señas realizadas. El programa mostrará un

mensaje informando que los datos fueron guardados después de un breve momento dependiendo del tamaño de las muestras. Se muestra el botón la figura 28:

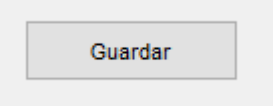


Figura 28 Botón Guardar

El código completo de la aplicación de la adquisición de datos se encuentra en anexos.

3.4.2 Elaboración del Programa

Al instalar la herramienta *Image Adquisition Tool* dentro de Matlab, se agregan librerías que interactúan directamente con el *SDK V2.0* mencionado en el capítulo 2, el cual elaboran funciones que son independientes del lenguaje de programación C++ y *Visual Studio*, permitiendo trabajar directamente con Matlab, evitando importar librerías de un lenguaje con el otro y traduciéndolas para su futuro uso. Por lo que es posible utilizar las funciones de adquisición de imágenes con Kinect v2 disponibles en sus barras de ayuda (Mathworks, Detect the Kinect V2 Devices, 2016). A continuación, se muestran las funciones empleadas para el programa de obtención de datos de Kinect v2.0 utilizando el *software* Matlab.

3.4.3 Funciones empleadas para el programa de obtención de datos

`Imaqhwinfo('kinect'):` El Kinect para Windows tiene dos sensores, un sensor de color y un sensor de profundidad. Esta función permite visualizar las características que posee Kinect v2.0 para detectar y comprobar que el dispositivo se encuentre en buen funcionamiento. Se muestra a continuación las características desplegadas por Matlab. (Mathworks, Detect the Kinect V2 Devices, 2016).

- **Información de sensor de video a color:**

```
DefaultFormat: 'RGB_1920x1080'
DeviceFileSupported: 0
DeviceName: 'Kinect V2 Color Sensor'
```

```
DeviceID: 1
VideoInputConstructor: 'videoinput('kinect', 1)'
VideoDeviceConstructor: 'imaq.VideoDevice('kinect', 1)'
SupportedFormats: 'RGB_1920x1080'
```

- **Información de sensor de video de profundidad:**

```
DefaultFormat: 'Depth_512x424'
DeviceFileSupported: 0
DeviceName: 'Kinect V2 Depth Sensor'
DeviceID: 2
VideoInputConstructor: 'videoinput('kinect', 2)'
VideoDeviceConstructor: 'imaq.VideoDevice('kinect', 2)'
SupportedFormats: 'Depth_512x424'
```

`Videoinput:` construye el objeto de entrada. Un objeto de entrada de video representa la conexión entre MATLAB y un dispositivo de adquisición de imágenes en particular. (Mathworks, Documentation videoinput, 2006).

Para habilitar la adquisición independiente de cada uno de los dispositivos de color y profundidad, se tratan como dos dispositivos independientes de la caja de herramientas de adquisición de imágenes, por lo cual se debe crear un objeto *Videoinput* para cada uno de los dispositivos. Ejemplo:

```
colorVid = videoinput('kinect', 1); %inicializa el video de color
depthVid = videoinput('kinect', 2); %inicializa el video de profundidad
```

`EnableBodyTracking`: Esta función permite habilitar la opción de sensado y reconocimiento del esqueleto, estos datos contienen la posición y el tiempo. Contiene la posición general del cuerpo y la posición 3-D respecto a las 25 articulaciones (posición en metros), está relacionado directamente con el video de profundidad ya que extrae las características de fuente del mismo. Ejemplo:

```
depthSource = getselectedsource(depthVid); %extrae la fuente de
profundidad
depthSource.EnableBodyTracking = 'on'; %utiliza la fuente de
profundidad para activar el tracking
```

`SkeletonConnectionMap`: Es un mapa de las conexiones del cuerpo que se desea reconocer a partir del video de profundidad, además de enlazar los diferentes puntos que lo componen para formar el esqueleto de la persona. La tabla 5 muestra los valores que corresponden a cada articulación del cuerpo.

Tabla 5

Numeración de articulaciones reconocidas por Kinect v2

Partes del cuerpo	Numeración
Base de la espina dorsal	1
Espina dorsal central	2
Cuello	3
Cabeza	4
Hombro derecho	5
Codo derecho	6
Muñeca derecha	7
Mano derecha	8
Hombro izquierdo	9
Codo izquierdo	10
Muñeca izquierda	11
Mano derecha	12
Cadera derecha	13
Derecha	14
Tobillo derecho	15
Pie derecho	16
Cadera izquierda	17
Rodilla izquierda	18
Tobillo izquierdo	19
Pie izquierdo	20

Hombro de la columna vertebral	21
Punta de la mano derecha	22
Pulgar derecho	23
Punta de la mano izquierda	24
Pulgar izquierdo	25

`ROIPosition`: Esta propiedad especifica la ventana de adquisición de región de interés. El ROI define el tamaño real del marco registrado por la caja de herramientas, medido con respecto a la esquina superior izquierda de un marco de imagen.

`ROIPosition` se especifica como un vector de elemento de 1x4, un ejemplo del enfoque de cámara se muestra como:

```
colorVid.ROIPosition = [650 300 600 600]; % [posición x, posición y, ancho, alto]
```

[X_Offset Y_Offset Anchura Altura]

Cada uno cumple con un rol diferente que se muestra a continuación:

- X_**`Offset`**: Posición de la esquina superior izquierda del ROI, medida en píxeles.
- Y_**`Offset`**: Posición de la esquina superior izquierda del ROI, medida en píxeles.
- Anchura: Ancho del ROI, medido en píxeles. La suma de X_Offset y ancho no puede exceder el ancho especificado en la resolución del video.
- Altura: Altura del ROI, medido en píxeles. La suma de Y_Offset y altura no puede exceder la altura especificada en la resolución de video.

A continuación se muestra la figura 29 que explica cada uno de los campos: (MathWorks, Documentation ROIPosition, 2016)

Video frame

Resolution

Figura 29 Configuración del enfoque de cámara.

Fuente: (MathWorks, Documentation ROIPosition, 2016)

El enfoque de cámara utilizado en el presente trabajo fue `[600 400 640 480]`, se varió el enfoque de la cámara a color y no el de profundidad, para evitar un almacenamiento excesivo de disco por la base de datos y mejorar el alcance del *skeleton traking.*

`Preview()`: Es una función que crea una ventana de vista previa de video que muestra datos de video en vivo para objetos de entrada de video, en este caso la Kinect. La ventana muestra la los datos de video al 100% de aumento. El tamaño de la imagen de vista previa está determinado por el valor de la propiedad *ROIPosition* del objeto de entrada de video explicado anteriormente.

`StopPreview()`: Es una función que detiene la vista previa de los datos de video del objeto de adquisición de imágenes.

`Implay()`: Es una función que abre la aplicación *Movie Player*. *Movie Player* puede ser usada para mostrar videos, secuencias o pilas de imágenes de MATLAB. Esta se empleará para visualizar como se indica en la figura 30 tanto el video de escala de colores como el video de profundidad para la comprobación de una correcta captura de datos. Ejemplo:

Sintaxis:

- `Implay`
- `implay(filename)`
- `implay(I)`

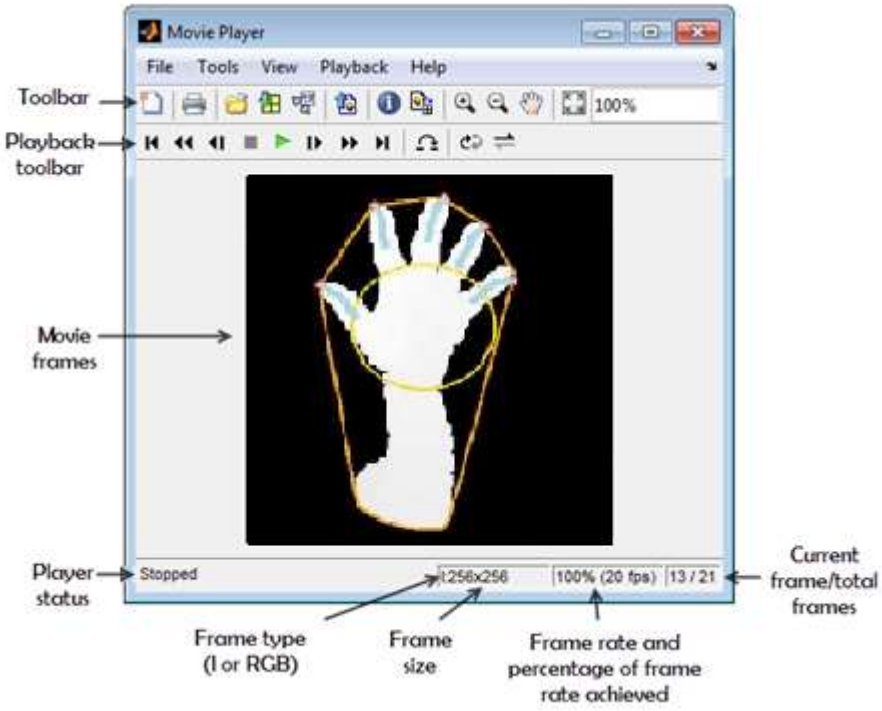


Figura 30 Ventana Movie Player de Implay()

Fuente: (Mathworks, Implay, 1997)

`Imshow`(): Es una función que permite mostrar imágenes, estas mismas se emplearán para la revisión de la obtención de datos de una forma más específica. Se indica con un ejemplo en la figura 31:

Sintaxis:

- `imshow(I)`
- `imshow(X,map)`
- `imshow(filename)`

Figura 31 Visualización de Imagen mediante Imshow()

Fuente: (Mathworks, Imshow, 1994)

`save`(): Es una función que guarda todas las variables del espacio de trabajo actual en un archivo binario formateado MATLAB (archivo .mat) llamado nombre de archivo. Se indica con un ejemplo con la figura 32:

Sintaxis:

- `save(h)`

Figura 32 Ventana de guardado de la función save()

Fuente: (Mathworks, Save, 1994)

3.5 Características de la base de datos

La base de datos está formada de 50 palabras diferentes que fueron codificados para tener un acceso más sencillo y organizado, el formato de los nombres correspondiente a las muestras de la base de datos se indica en la tabla 6 y su orden a continuación:

Para el orden de palabras:

tipo de signo _ tipo de palabra _género _# de persona _tipo de dato _palabra _número (#) de repetición

Ejemplo:

Para la repetición 1 de la palabra morado del grupo de colores de la persona 1 para el tipo de dato ColorImg

P_Cl_M_P1_Co_morado_r1

Para el orden de letras:

tipo de signo _género _número (#) de persona _tipo de dato _letra

Ejemplo:

Para la letra “a” de la persona 1 para el tipo de dato ColorImg

Lt_M_P1_Co_a

Tabla 6

Codificación de muestras

Variable	Representación
Tipo de signo	Este parámetro se estableció de la siguiente manera. P: Palabra Lt: letra
Tipo de palabra	Este parámetro viene dado por las 2 primeras letras del tipo de palabra: Ad: Adjetivos Al: Alimentos Sa: Saludos Ju: Juguetes & Cosas C: Colores
Género	Este parámetro viene dado por la primero letra del género. Ej: F: Femenino M: Masculino
Número (#) de persona	El formato viene dado por una P seguido de su número correspondiente. Ej: P1 para persona 1
Palabra	Este parámetro será la palabra que se desea escoger.
Tipo de dato	Este parámetro se establece de la siguiente manera: ColorImg: Co DepthImg: Dp *Skeleton tracking* (Metadata) : Mt
Número (#) de repetición	El formato viene dedo por una R seguido de su número correspondiente. Ej: R1 para repetición 1.

3.6 Descripción del buscador de base de datos

Para realizar búsquedas con un acceso más rápido y dinámico a los elementos de la base de datos, se elaboró un programa que clasifique las distintas muestras de forma ordenada devolviendo listas con los nombres de los archivos para que puedan ser utilizados posteriormente para entrenamiento de máquinas de soporte vectorial, a

continuación se muestra la interfaz gráfica completa del buscador de la base de datos en la figura 33 y la explicación de sus diferentes características:

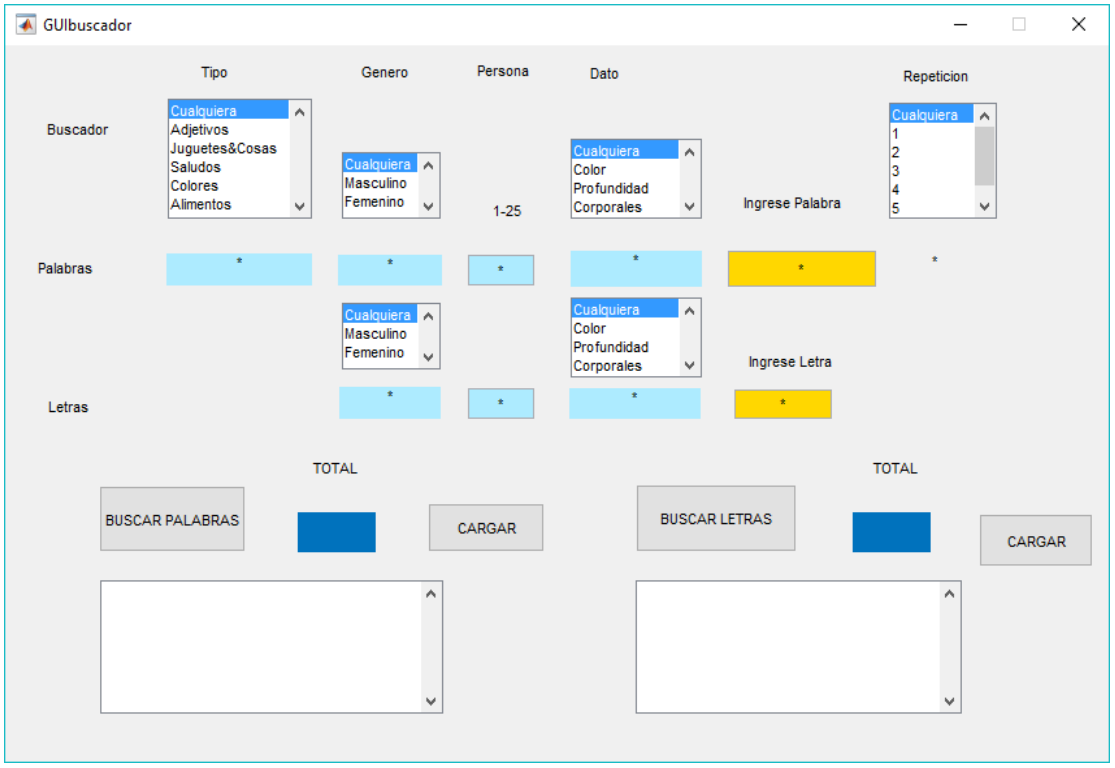


Figura 33 Interfaz Gráfica del Buscador de datos

Lista del tipo de palabras: Esta opción muestra las diferentes palabras disponibles en la base de datos que son adjetivos, juguetes & cosas, saludos, colores, alimentos y/o otras. Según la opción seleccionada se mostrará la codificación correspondiente debajo de la lista del tipo de palabra. Se muestra la lista en la figura 34.

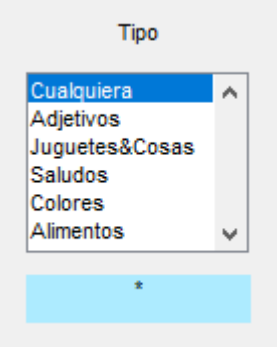


Figura 34 Lista tipo de palabra

Lista del tipo de género: Esta opción muestra el género de la persona que ejecutó las muestras de las palabras disponibles en la base de datos que son: masculino, femenino y/o cualquiera. Según la opción seleccionada se mostrará la codificación correspondiente debajo de la lista género. Se muestra la lista en la figura 35.

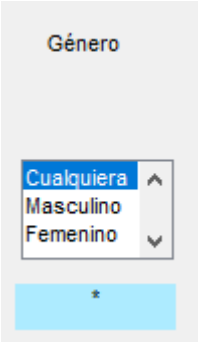


Figura 35 Lista tipo de género

Ingreso del número (#) de persona: Permite ingresar el número de la persona que se desea revisar. Si se ingresa "*", indica que puede ser cualquiera, por lo tanto se tomarán todas las personas. Se muestra una captura en la figura 36.

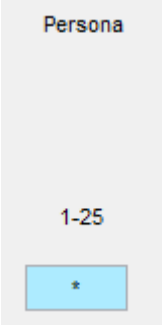


Figura 36 Número de persona

Lista del número (#) de repetición: Cada una de las palabras fueron tomadas varias veces por la misma persona, y cada una de estas repeticiones van a ser codificadas con un número, esta lista muestra el número (#) de repetición de dicha palabra que se busca trabajar entre 5 posibles repeticiones. Según la opción seleccionada se mostrará la codificación correspondiente debajo de la lista del número (#) de repetición. Se muestra la lista en la figura 37.

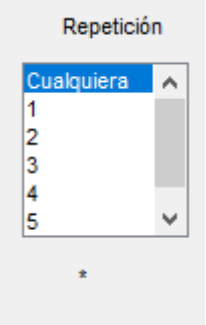


Figura 37 Número tipo de repetición

Lista del tipo de dato: Esta opción muestra el tipo de dato que se desea obtener entre ellos esta color, profundidad, corporales y/o cualquiera. Según la opción seleccionada se mostrará la codificación correspondiente debajo de la lista del tipo de dato. Se muestra la lista en la figura 38.

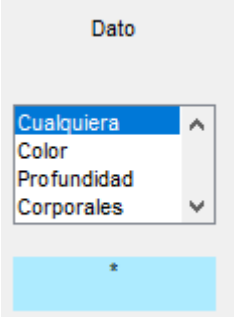


Figura 38 Lista del tipo de dato

Ingreso de la palabra: Permite ingresar una palabra en específico que se desea revisar los datos. Si se ingresa "*", indica que puede ser cualquiera, por lo tanto se tomarán todas las palabras. Se muestra la lista en la figura 39.

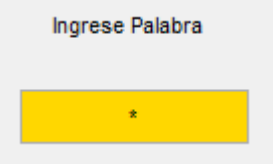


Figura 39 Ingreso de palabra

Buscar palabra: Indica al buscador revisar dentro de la base de datos, las palabras que cumplan con los parámetros especificados de las diferentes listas e ingresos, además se indicará en un cuadro azul la cantidad de muestras que existentes. Se muestra una captura en la figura 40.

Figura 40 Buscar palabras

Buscar letra: Indica al buscador revisar dentro de la base de datos, las letras que cumplan con los parámetros especificados de las diferentes listas e ingresos, además se indicará en un cuadro azul la cantidad de muestras que existentes. Se muestra una captura en la figura 41.

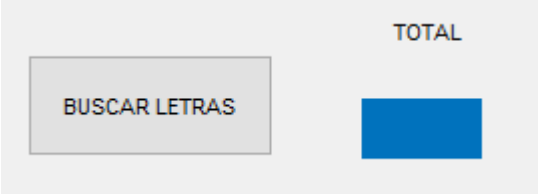


Figura 41 Buscar Letras

Cargar (palabras y letras): Una vez seleccionado características del dato de las opciones de las listas, el botón cargar toma dicho dato y lo carga al *workspace* de MATLAB para tener acceso a su información. Se muestra una captura en la figura 42.

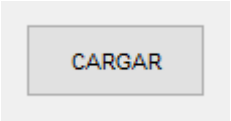
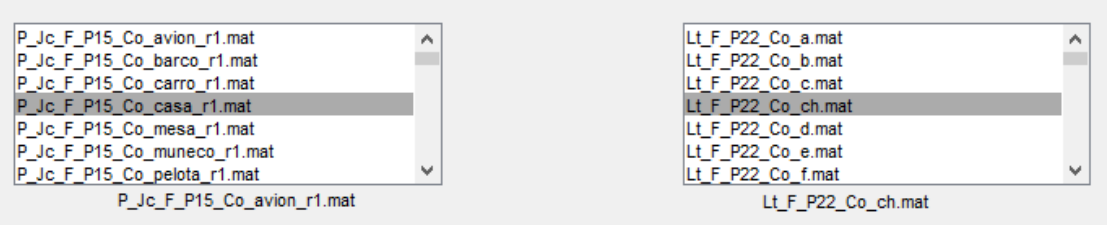


Figura 42 Botón cargar

Cuadros de opciones: Después de pulsar el botón buscar palabras o letras se desplegará una lista de las palabras que cumplan dichas características. Al seleccionar y cargar de las diferentes opciones, la opción seleccionada se mostrará al final debajo de la lista. La figura 43 (a) muestra la lista de opciones de palabras y la figura 43 (b) muestra la lista de opciones de letras.



(a) (b)

Figura 43 Lista de opciones

El buscador, permite cargar palabras y letras por separado, para una organización más sencilla y amigable para el usuario.

El código completo del buscador de la base de datos se encuentra en el anexo 2.

CAPITULO IV
VERIFICACIÓN DE BASE DE DATOS

El propósito de este capítulo es verificar si la base de datos es útil para realizar un reconocimiento del lenguaje de señas utilizando la base de datos mediante un ejemplo de extracción, entrenamiento de máquina y evaluación de resultados. Como se mencionó en el capítulo 2 y 3, se utilizaron los datos de *skeleton tracking* los cuales son pre-procesados obteniendo sus respectivos valores de posición de las articulaciones detectadas. Estos datos son empleados para la aplicación *Classification Learner* de MATLAB en el entrenamiento de varios modelos de máquinas de soporte vectorial para conseguir un algoritmo de predicción con 5 palabras del conjunto de "saludos". Además dependiendo del modelo de clasificación y predicción existirá un porcentaje de aciertos para el algoritmo elaborado con las diferentes palabras. El proceso para realizar el reconocimiento con las muestras de la base de datos se indica de forma más simple en la figura 44:

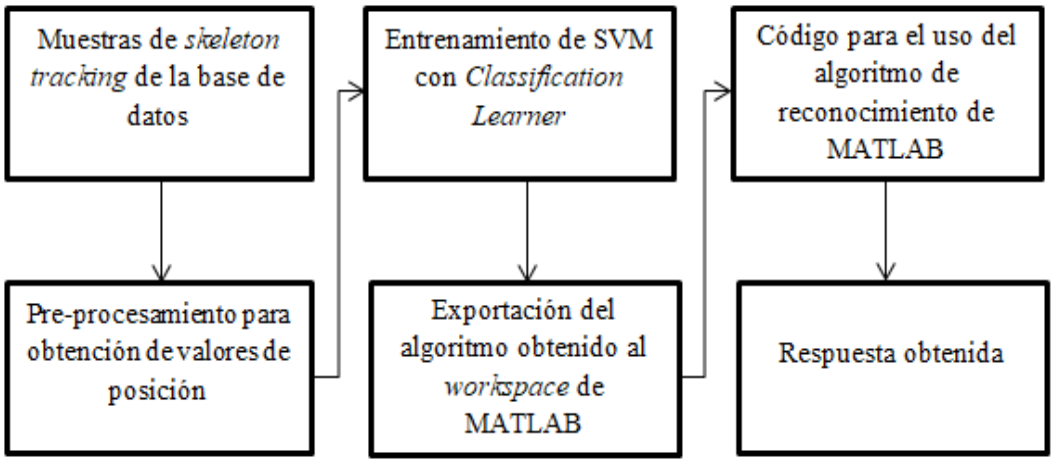


Figura 44 Proceso de reconocimiento mediante datos de *skeleton tracking*

4.1 Etapas del procesamiento de datos de interés previo a aprendizaje de máquina

Antes de utilizar el *classification learner* de MATLAB, es necesario procesar los datos que se pueden obtener a partir de los videos a color y videos de profundidad con su respectivo *skeleton tracking* (metadata), para tomar únicamente los datos de interés, esto puede variar dependiendo de cada palabra debido a su propio patrón característico. Para ello se desarrolló un programa que procese cada uno de los videos de la base de datos para las 5 palabras saludos, que se explicará a continuación:

4.1.1 Clasificación de datos de Skeleton Tracking

Antes de realizar el procesamiento de los datos, es necesario realizar una adecuada clasificación. Para la clasificación de datos de *Skeleton Tracking* (Metadata) se analizará cada video de color y profundidad de cada palabra por separado, teniendo en cuenta las siguientes cuestiones para el análisis y procesamiento de los videos:

- *Frames* de interés: el intervalo de *frames* significativos en donde la palabra se produce.
- *Skeleton Tracking* de brazos y manos: como se explicó en el capítulo 2 sobre las características de Kinect y su capacidad de detectar las partes del cuerpo humano, dicha detección se realizará para los brazos y manos, esto se debe a que las partes del esqueleto como el tronco y la cabeza son despreciables dado que no forman signos que puedan caracterizar a una palabra, al contrario puede producir errores debido a similitudes entre ellos.

4.1.2 Tipos de datos

Para que la aplicación *Classification Learner* pueda procesar datos son necesarios dos tipos de datos:

- Tipo de Dato "*Double*": Son variables numéricas como valores de coma flotante de precisión doble que son de 8 bytes (64 bits).
- Tipo de Dato "*Categorical*": Es un tipo de dato que compone de valores de un conjunto finito de categorías discretas. Dichas categorías pueden estar en orden matemático. Este tipo de matriz categórica proporciona

almacenamiento eficiente y manipulación conveniente de datos no numéricos, al tiempo que es capaz de mantener nombres significativos. Un uso común de matrices categóricas es especificar grupos de filas en una tabla, el cual será utilizado para el tratamiento de los datos.

- Tipo de dato "*Table*": Es un tipo de dato orientado a columnas que a menudo se almacenan en un archivo de texto o en una hoja de cálculo. Las tablas consisten en filas y variables orientadas a columnas. Además cada una de estas variables puede tener un tipo de datos diferente y un tamaño diferente pero deben tener el mismo número de filas.

4.1.3 Programa de pre-procesamiento

A partir de los videos de la base de datos se trabaja el *skeleton tracking*, para verificar que es válida. Se emplea un programa para procesar cada video, los cuales se analizan con el propósito de tomar solo los *frames* de interés explicados anteriormente, y se extrae las características significativas que contienen información relacionada al signo, finalmente se obtiene una variable "`k`" que tiene todos los datos para ser empleado en la aplicación *Classification Learner* que se explicará más adelante. Se detalla de forma más simple en la figura 45 sobre el algoritmo empleado para pre-procesamiento. Para ello se explica a continuación las funciones empleadas y los datos de interés dentro del programa.

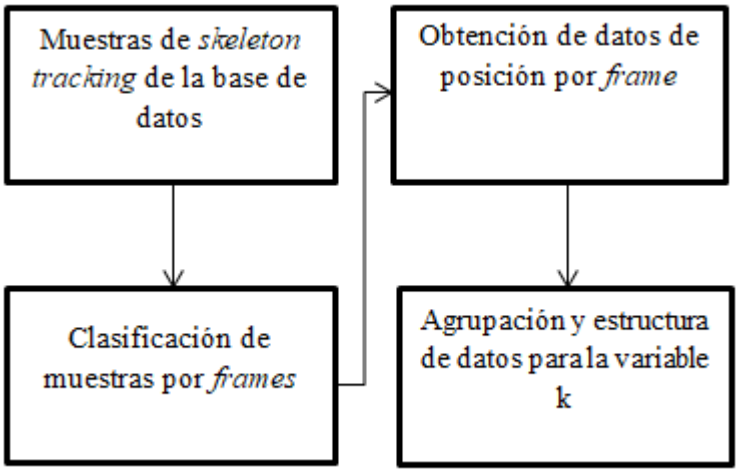


Figura 45 Diagrama de flujo de pre-procesamiento

4.1.3.1 Funciones Empleadas para el programa

Los datos de *skeleton tracking* de cada palabra deben ser pre-procesados, la función empleada para ello es el siguiente:

```
function preprocesamiento()  %nombre de la función

for i=1:m                    %división de muestras por persona

n=strcat(d,'\',Lista{i});    %brindar acceso a la base de datos
load(n)                      %cargar los datos de interés
c=[];                        %inicialización de la variable
for j=17:3:40                %empleo de frames significativos(17-40)

anyBodiesTracked=any(metadata(j).IsBodyTracked~=0);
trackedBodies = find(metadata(j).IsBodyTracked);

%obtención de datos de cuerpos que fueron detectados por Kinect

a= metadata(j).DepthJointIndices(5:12,:, trackedBodies);
a1= metadata(j).DepthJointIndices(22:24,:, trackedBodies);

%divide los datos para que utilicen solo los datos de mano, hombro,
codo, muñeca, pulgar e índice de la mano.

b=[a(:,1);a(:,2);a1(:,1);a1(:,2)];
b=b';
c=[c,b];

%guarda los datos generados por la función for en una matriz
```

```
    end
    k=[k;c];
    %guarda los datos en una matriz con los datos de todas las personas
    end


f={'gracias';'gracias';'gracias';'gracias';'gracias';'gracias';'gracias
';'gracias';'gracias';'gracias';'gracias';'gracias';'gracias'}

%nombres correspondiente a cada signo empleado en el código

    end    %fin de la función
```

`IsBodyTracked`: Esta función nos permite tomar únicamente los datos de la matriz de muestras del *skeleton tracking* (metadata) en donde se detecta a una persona, debido que la capacidad de Kinect puede llegar a tomar hasta 6 personas.

`DepthJointIndices`: Esta función nos permite tomar los puntos de las articulaciones que conforman el esqueleto de la persona, el rango empleado para tomar los datos de interés que son hombro, codo, muñeca, mano es de 5 a 12, la punta y pulgar de la mano es de 22 a 24 respectivamente, como se indica en la tabla 6 del capítulo 3. En estos rangos están incluidos los datos tanto para la parte derecha como para la parte izquierda del esqueleto.

Todos estos datos son guardados como matriz, los cuales convertidos posteriormente a tablas para trabajar con el programa *Classification Learner*.

4.1.3.2 Datos de interés

Entre los datos de interés del programa e pre-procesamiento se destacan las siguientes variables:

`Variable a`: Esta variable contiene los datos individuales de las posiciones "x" y "y" de cada uno de los puntos del *skeleton tracking* de hombro, codo, muñeca y mano que fueron reconocidos. Estos datos están en formato "*double*".

`Variable al`: Esta variable contiene los datos individuales de las posiciones "x" y "y" de cada uno de los puntos del *skeleton tracking* de la punta de la mano y el pulgar que fueron reconocidos. Estos datos están en formato "*double*".

`Variable b`: Esta variable contiene las variables "`a`" y "`al`" en forma de matriz en una sola fila para facilidad al momento de emplear *classification Learner*. Estos datos están en formato "*double*".

`Variable f`: Esta variable es de tipo "*categorical*", se emplea debido que posee las diferentes tipos por los cuales se clasificará unos datos de otros. A continuación en la figura 46 un ejemplo de la variable "`f`", y su estructura cuando se desea realizar un reconocimiento con las palabras "gracias", "hola" y "mucho gusto".

70x1 categorical

	1	2
1	gracias	
2	gracias	
3	gracias	
4	gracias	
5	gracias	
6	gracias	
7	gracias	
8	gracias	
9	gracias	

70x1 categorical

	1	2
14	hola	
15	hola	
16	hola	
17	hola	
18	hola	
19	hola	
20	hola	
21	hola	
22	hola	

70x1 categorical

	1	2
28	muchogusto	
29	muchogusto	
30	muchogusto	
31	muchogusto	
32	muchogusto	
33	muchogusto	
34	muchogusto	
35	muchogusto	
36	muchogusto	

Figura 46 Estructura de la variable f

`Variable k`: Esta variable es la que será tomada y evaluada por el *Classification Learner* para el entrenamiento de máquina con los diferentes modelos de máquinas de vectores de soporte. Esta variable es de tipo "*table*", formado tanto de variables "*double*" y variables "categorical", y contendrá toda la información de las variable "`b`" y la variable "`f`" en formato de filas, que irán incrementados dependiendo de la cantidad de *frames* tomadas de las muestras. A continuación se muestra un ejemplo de la variable "`k`" en la figura 47 para las palabras "gracias" y "hola":

Dedos	Codos y hombros	Mano	Muñeca	Palabra
269.8008	307.9341	298.4298	275.7993	hola
263.3918	309.5002	300.8929	306.7236	hola
276.0010	318.8746	306.1138	327.9993	hola
274.2027	318.1400	304.9874	319.0002	gracias
261.0010	301.5006	293.7679	302.5007	gracias
265	306.0618	298.8970	304	gracias

Figura 47 Variable k

El código completo para realizar el pre-procesamiento se encuentra en anexo 3.

4.2 Pasos para el aprendizaje de máquina

Una vez realizado el pre-procesamiento de los datos de interés de las diferentes muestras de video y *skeleton tracking* para obtener la variable "k", se debe trabajar con la aplicación de *toolbox Classification Learner* de MATLAB. Los pasos se describen a continuación:

4.2.1 Acceso a la aplicación

Para acceder a la aplicación en MATLAB R2016a, se debe dirigir a la pestaña APPS en la parte superior y hacer click en la pestaña *Classification Learner* como se muestra en la figura 48, de lo cual se desplegará la ventana de la aplicación que se puede observar en la figura 49.

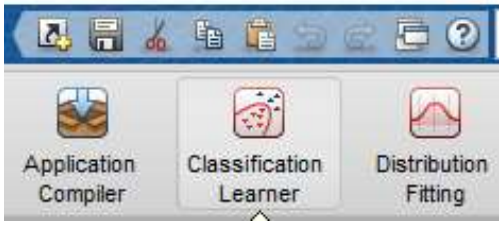


Figura 48 Acceso Classification Learner

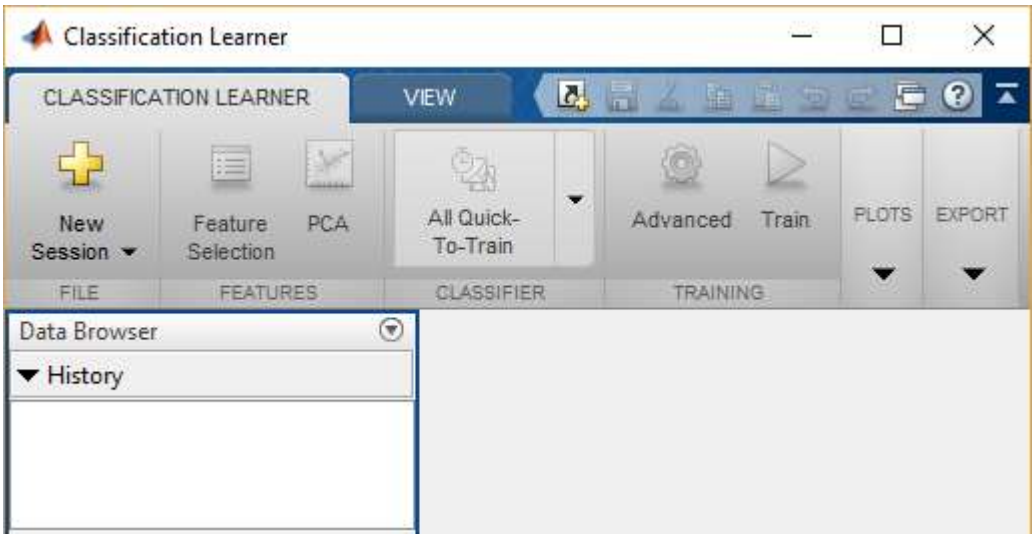


Figura 49 Ventana de la aplicación Classification Learner

4.2.2 Comienzo de una sesión

Una vez abierta la aplicación se debe dirigir a la pestaña *New Session*, la cual desplegará una ventana de opciones, esta se divide en 3 pasos.

Paso 1: En este paso 1 se desplegará las diferentes variables existentes que se encuentran cargadas en MATLAB, de las cuales varias son creadas por defecto al compilar el programa de pre-procesamiento, se escoge en este paso la variable "k", debido que es la que posee toda la información del procesamiento previo anteriormente explicado, se muestra a continuación el paso 1 en la figura 50.

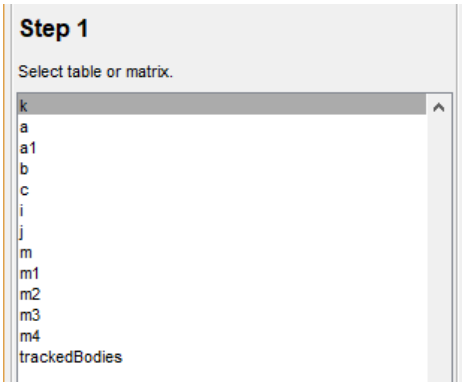
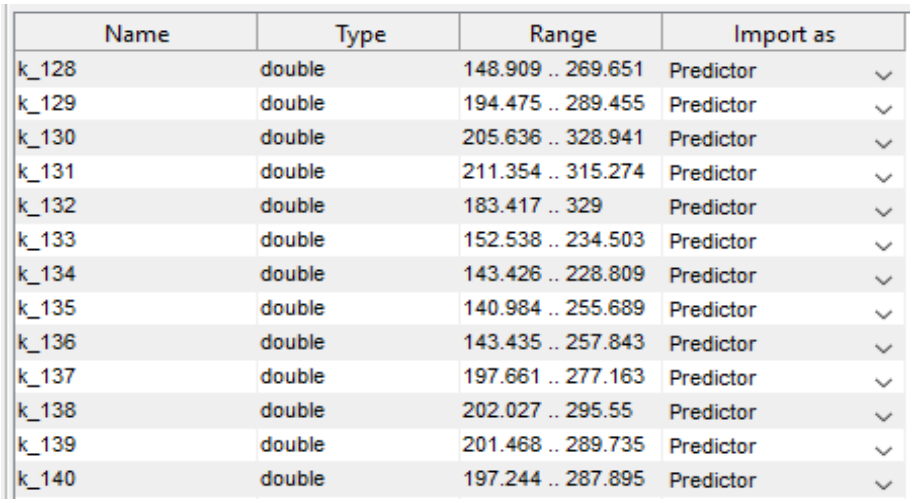


Figura 50 Ventana Classification Learner Paso 1

Paso 2: En este paso se pueden revisar y comprobar los datos dentro de la variable `k`, como establecer cuáles serán las variables que se utilizarán para el entrenamiento del algoritmo al establecerlas como predictores como se muestra en la siguiente figura 51.



Name	Type	Range	Import as
k_128	double	148.909 .. 269.651	Predictor
k_129	double	194.475 .. 289.455	Predictor
k_130	double	205.636 .. 328.941	Predictor
k_131	double	211.354 .. 315.274	Predictor
k_132	double	183.417 .. 329	Predictor
k_133	double	152.538 .. 234.503	Predictor
k_134	double	143.426 .. 228.809	Predictor
k_135	double	140.984 .. 255.689	Predictor
k_136	double	143.435 .. 257.843	Predictor
k_137	double	197.661 .. 277.163	Predictor
k_138	double	202.027 .. 295.55	Predictor
k_139	double	201.468 .. 289.735	Predictor
k_140	double	197.244 .. 287.895	Predictor

Figura 51 Ventana Classification Learner Paso 2

Paso 3: En este paso se realiza la validación del algoritmo que estima el rendimiento del modelo en datos nuevos en comparación con los datos de entrenamiento que en este caso es la variable "`k`", y ayuda a elegir el mejor modelo. La validación protege contra el sobreajuste, que es el entrenamiento excesivo de un algoritmo de aprendizaje con muestras que ya se conocen el resultado deseado.

La variable de interés es "`k`", y se tomará una validación cruzada de 50 particiones, debido que se busca tener el mayor número posible de los datos para el entrenamiento del algoritmo para mejorar la precisión y efectividad de la misma. Se muestra una captura de las opciones en la figura 52.

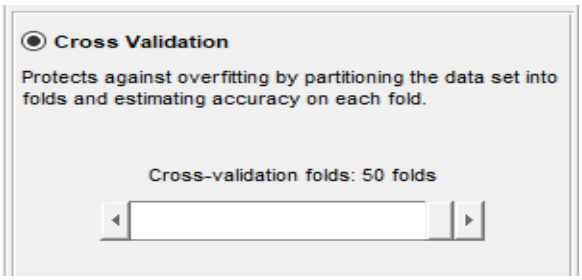


Figura 52 Ventana Classification Learner Validación

Una vez verificado este detalle, se hace click en *Start Session* para que se empiece a clasificar las diferentes muestras y evaluar los resultados.

4.2.3 Entrenamiento del Algoritmo

La aplicación *Classification Learner* cuenta con varios modelos de entrenamiento de aprendizaje de máquina, entre ellos máquinas de vectores de soporte (SVM), para evaluar los resultados de las opciones que proporciona MATLAB con SVM se trabajará con *Linear* SVM, *Cuadratic* SVM y *Cubic* SVM. Se muestra una captura de los diferentes algoritmos en la figura 53.

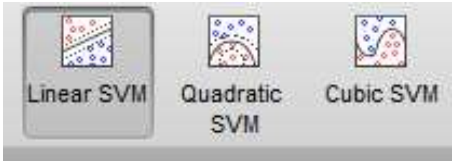


Figura 53 Tipos de clasificador para aprendizaje de máquina

Una vez seleccionado el modelo que se desea entrenar se debe dirigir a la pestaña *Train* y hacer click, como se muestra en la figura 54.

Figura 54 Entrenamiento del algoritmo

Una vez completado el proceso, el modelo mostrará en el lado derecho la precisión del algoritmo y las características que es la cantidad de datos utilizados para el aprendizaje de máquina como se muestra en l siguiente figura 55:

Figura 55 SVM Classification Learner

La matriz de confusión es una herramienta que muestra los porcentajes de aciertos y fallos dependiendo del número y tipo de muestras utilizadas para el procesamiento, representando de forma más simple y resumida el modelo del algoritmo de aprendizaje obtenido por aprendizaje de máquina. Esta ayuda a entender si un modelo es aceptable al tener valores altos de acierto o si es desechable al tener varias muestras no reconocidas correctamente. Para analizar la matriz de confusión se observan los porcentajes de similitud en los casilleros de la matriz, la parte vertical indica la palabra y la sección horizontal muestra la palabra comparada. Se explica más detalladamente con un ejemplo con la figura 56.

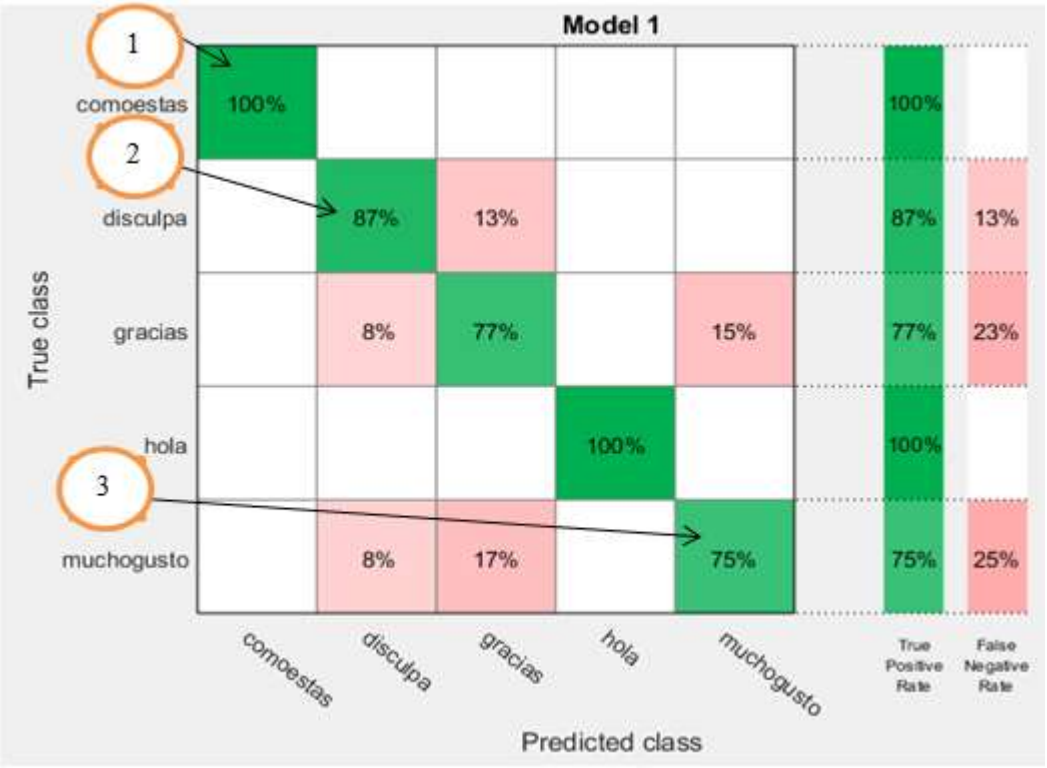


Figura 56 Matriz de confusión

La matriz de confusión indica que la palabra "como estas" del casillero 1 comparada con la palabra "como estas" tiene un 100%, es decir que dentro de las muestras empleadas para el entrenamiento del algoritmo no se tuvo problema y fueron distinguidas entre ellas. La palabra "disculpa" que se indica en el casillero 2 tiene un 87%, eso quiere decir que el 13% de muestras de dicha palabra son parecidas a "gracias". En el casillero 3 indica que la palabra "mucho gusto" tiene un 75% de precisión, un 8% de las muestras empleadas en el entrenamiento son parecidas a la palabra "disculpa" y un 17% son parecidas a la palabra "gracias". Esta matriz se empleará para análisis de algoritmos SVM que se verán en el capítulo 5.

4.2.4 Extracción del modelo de predicción

Una vez realizada el análisis y comprobación con la matriz de confusión, es necesario extraer el modelo para trabajar con él. Para ello se va a ala pestaña *Export*

Model y se elige el tipo de exportación que se necesite como se muestra en la siguiente figura 57.

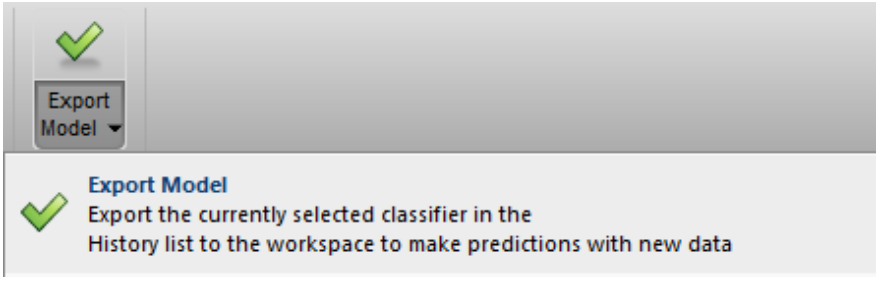


Figura 57 Pestaña Export Model

Una vez seleccionado, este desplegará otra ventana en la cual pedirá el nombre con el cual guardar el modelo, después de eso, el modelo se guardará en el *Workspace* de MATLAB como archivo temporal para la utilización del modelo como se muestra en la figura 58.

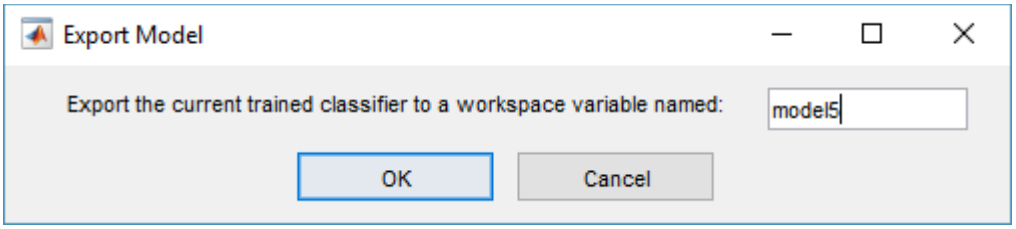


Figura 58 Ventana Export Model

4.3 Comprobación del algoritmo de aprendizaje

Para realizar la comprobación se debe utilizar el algoritmo adquirido del entrenamiento de máquina y mediante un código de MATLAB evaluar muestras de la base de datos que no hayan sido empleadas durante el aprendizaje de máquina. Para ello se siguió los siguientes pasos:

4.3.1 Pre-procesamiento de datos para el código de aprendizaje

Al igual que el pre-procesamiento de datos antes de emplear entrenamiento automático, se debe pasar por un procesamiento aquellos videos que se desean utilizar para el reconocimiento, empleando el mismo algoritmo utilizado para los datos de

entrenamiento, esto permite que los datos sean reconocibles para el modelo generado a través de máquinas de vectores de soporte.

4.3.2 Empleo del modelo de algoritmo de aprendizaje

Una vez procesados datos, y guardados en forma de tablas se procede a emplear el algoritmo, para el ejemplo se tomó como muestra de la base de datos el signo "gracias" de la persona 15, el código empleado se muestra a continuación:

```
k=table(k)                    % transforma la variable k en tipo table
k.Properties.VariableNames = {'k'} %cambia los nombres de la tabla a "k"
yfit=traductorf.predictFcn(k)     %código para evaluar el algoritmo con
el dato k
```

- `Variable traductorf`: Es el nombre con el cual se guardó el algoritmo en el *Workspace* de MATLAB. El nombre "traductorf" puede variar dependiendo del usuario, debido que es el nombre con el que se guardó el algoritmo.
- `VariablesNames`: Permite que las variables dentro la tabla se guarde como "k", debido que las muestras utilizadas en el entrenamiento del algoritmo tienen el nombre "k", las muestras de prueba deben tener el nombre "k", para para que exista compatibilidad y el algoritmo pueda reconocer los datos.
- `Variable yfit`: Es la respuesta del programa, esta entregará el string de la palabra reconocida por el modelo:

```
yfit = gracias
```

El código completo del empleo del modelo de reconocimiento se encuentra en anexo 4.

CAPITULO V
PRUEBAS Y ANÁLISIS DE RESULTADOS

Este capítulo se centra en la elaboración, pruebas y análisis de resultados de varios algoritmos de aprendizaje de máquina obtenidos a partir de las diferentes modelos de máquina de soporte vectorial (SVM) ofrecidos por MATLAB, las matrices de confusión se toman en cuenta para los análisis de resultados, para varios tipos de aprendizaje entre los 5 tipos de palabras explicados en el capítulo anterior. En referencia al trabajo (Zhang, Xu, & Sun, 2015) se procede a extraer las características únicamente de los *frames* relevantes, es decir donde existe el movimiento y se forma la palabra como tal.

5.1 Especificaciones generales para extracción de características

Se desarrolló un algoritmo para la extracción de características desde el movimiento de la mano hasta la culminación de la palabra, en promedio las personas tardan 2 segundos en formular un signo, que es el equivalente a 60 *frames*. Se desprecian los primeros *frames* debido que no son importantes hasta que se genere la seña representativa, bajo las siguientes condiciones:

Tabla 7

Palabras de evaluación

Palabra
Como estas
Disculpa
Hola
Mucho gusto
Gracias

- Rango de *frames* tomados 17-40, debido que es el rango en donde se produce el signo de la palabra.

- Modelo desarrollado en base a los datos de *skeleton tracking* (metadata) a partir de los videos de profundidad.
- Número de muestras aprendidas 13 y por analizar 5.

5.1.1 Algoritmo SVM 1

El algoritmo SVM 1 se desarrolló utilizando el sistema Lineal de SVM, tomando en cuenta las características que provee Kinect v2, el cual utiliza un total de 25 articulaciones y 800 características como se muestra en la figura 59 correspondiente a todas las palabras analizadas, obteniendo un porcentaje de eficiencia del 87.5%

Figura 59 Precisión algoritmo SVM 1

Los resultados de los porcentajes de aciertos y fallos durante el entrenamiento del algoritmo se puede observar en la tabla 8. Como se puede observar en la figura 60, la palabra "gracias" presenta una mayor probabilidad de error debido a que tiene cierta similitud con "mucho gusto" ya que las posiciones de las manos son similares en base al promedio de *frames* utilizado como se puede ver en el anexo 5, la predicción puede mejorar si se aumenta el número de muestras para el aprendizaje de máquina.

Tabla 8

Porcentaje de aciertos y fallos algoritmo SVM 1

Palabra	Porcentaje de aciertos	Porcentaje de fallos
Como estas	100%	0%
disculpa	91%	9%
gracias	70%	30%
hola	92%	8%
Mucho gusto	80%	20%

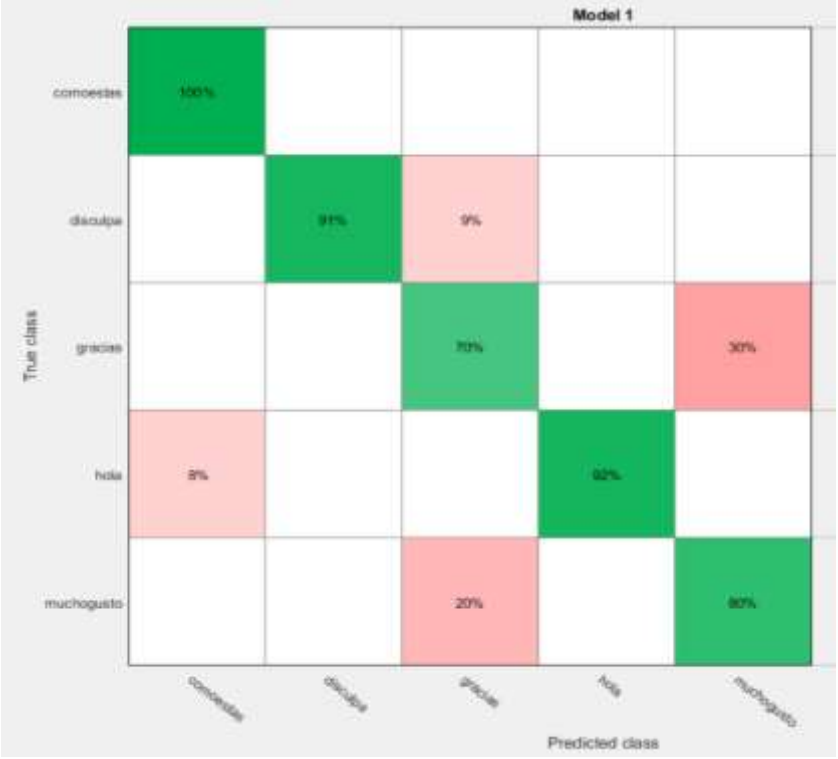


Figura 60 Matriz de confusión algoritmo SVM 1

5.1.2 Algoritmo SVM 2

El algoritmo SVM 2 se desarrolló utilizando el sistema cuadrático de SVM, tomando en cuenta las 25 articulaciones reconocibles que provee Kinect v2 para un total de 800 características correspondiente a todas las palabras analizadas, en este caso se obtuvo una menor precisión del 76,8% respecto al algoritmo SVM 1 como se muestra en la figura 61.

Figura 61 Precisión algoritmo SVM 2

Los resultados de los porcentajes de aciertos y fallos del algoritmo se puede observar en la tabla 9. Se observa que la precisión bajo en comparación con el algoritmo SVM 1.

Como se puede observar en la figura 62, las palabras “gracias” y “mucho gusto” presentan una mayor probabilidad de error debido ya que las posiciones de las manos en ambas palabras son similares en base al promedio de *frames* utilizado, la predicción puede mejorar si se aumenta el número de muestras para el aprendizaje de máquina.

Tabla 9

Porcentaje de aciertos y fallos algoritmo SVM 2

Palabra	**Porcentaje de aciertos**	**Porcentaje de fallos**
Como estas	92%	8%
disculpa	73%	27%
gracias	60%	40%
hola	92%	8%
Mucho gusto	60%	40%

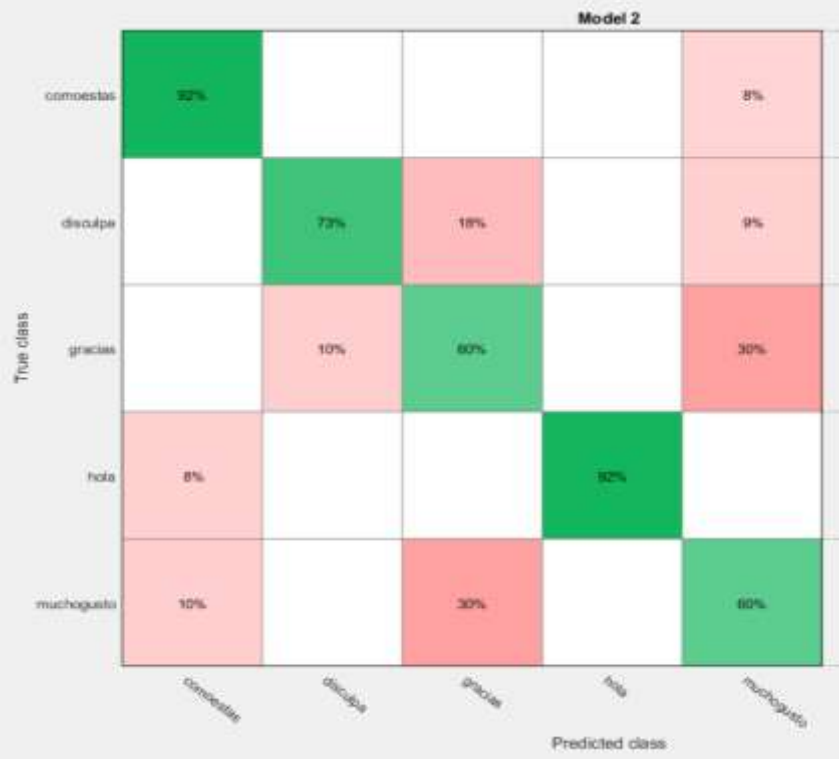


Figura 62 Matriz de confusión SVM 2

5.1.3 Algoritmo SVM 3

El algoritmo SVM 3 se desarrolló utilizando el sistema cúbico de SVM, tomando en cuenta las 25 articulaciones reconocibles que provee Kinect v2 para un total de 800 características correspondiente a todas las palabras analizadas, en este caso se obtuvo la misma precisión en comparación al algoritmo 2 de 76,8% como se muestra 63.

Figura 63 Precisión algoritmo SVM 3

Los resultados de los porcentajes de aciertos y fallos del algoritmo se puede observar en la tabla 10. Se observa que este modelo tiene la misma precisión respecto al algoritmo SVM 2, no obstante como se puede observar en la figura 64, en este modelo las palabras “disculpa” y “mucho gusto” presentan una probabilidad de error de 20% que es mayor en comparación al algoritmo 2, pero se mejoró el resultado para “gracias” y

"mucho gusto" al reducirse la probabilidad de error al 10%, la predicción puede mejorar si se aumenta el número de muestras para el aprendizaje de máquina.

Tabla 10

Porcentaje de aciertos y fallos algoritmo SVM 3

Palabra	Porcentaje de aciertos	Porcentaje de fallos
Como estas	92%	8%
disculpa	73%	27%
gracias	60%	40%
hola	92%	8%
Mucho gusto	60%	40%

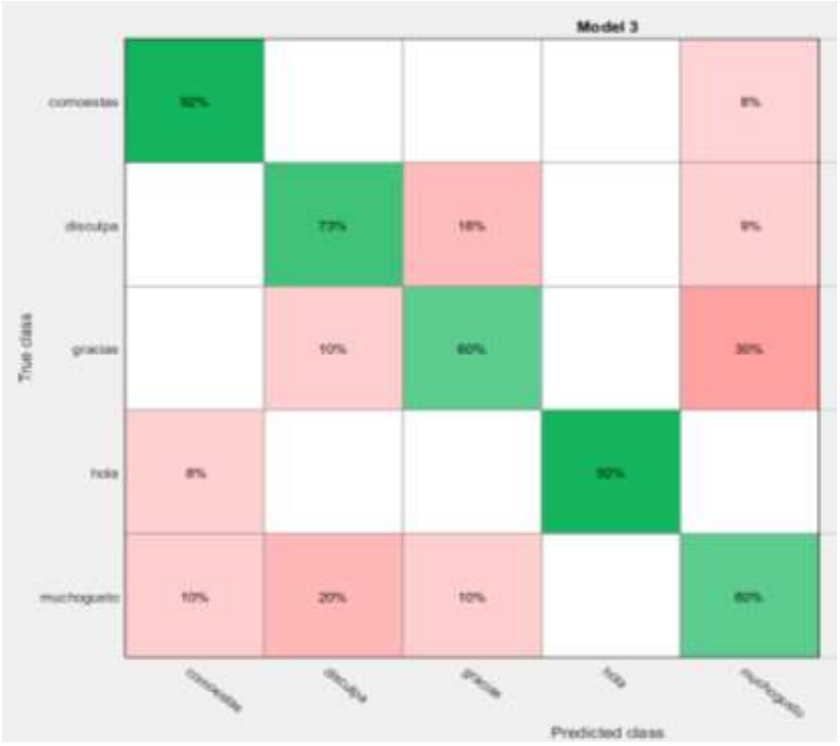


Figura 64 Matriz de confusión algoritmo SVM 3

5.1.4 Algoritmo SVM 4

El algoritmo SVM 4 se desarrolló utilizando el sistema lineal de SVM, tomando en cuenta solo las articulaciones mano, hombro, codo, pulgar, muñeca y punta de dedo de las partes derecha e izquierda que son las partes del cuerpo utilizadas para realizar el signo y únicamente los *frames* característicos 17 y 40, que son los *frames* en donde se inicia y termina el movimiento de la palabra, para un total de 154 características correspondiente a las muestras analizadas, en este caso se obtuvo una mayor precisión en comparación a algoritmos anteriores de 88,6% como se muestra 65.

Figura 65 Precisión algoritmo SVM 4

Los resultados de los porcentajes de aciertos y fallos del algoritmo se puede observar en la tabla 11. Se puede observar que en este modelo se obtiene mayor cantidad de precisión en el reconocimiento de palabras en comparación a los algoritmos mostrados anteriormente, como se puede observar en la figura 66, la palabra "como estas" y "hola" tienen un 100% de aciertos, esto es debido que los *frames* iniciales y finales de cada video (cuando los brazos se encuentran abajo) tienen gran similitud lo cual causa dificultad para distinguir entre palabras al momento de entrenar el algoritmo, y al ser eliminados se resuelve este problema obteniendo un algoritmo con mayor precisión.

Tabla 11

Porcentaje de aciertos y fallos algoritmo SVM 4

Palabra	**Porcentaje de aciertos**	**Porcentaje de fallos**
Como estas	100%	0%
disculpa	87%	13%
gracias	71%	29%
hola	100%	0%
Mucho gusto	82%	18%

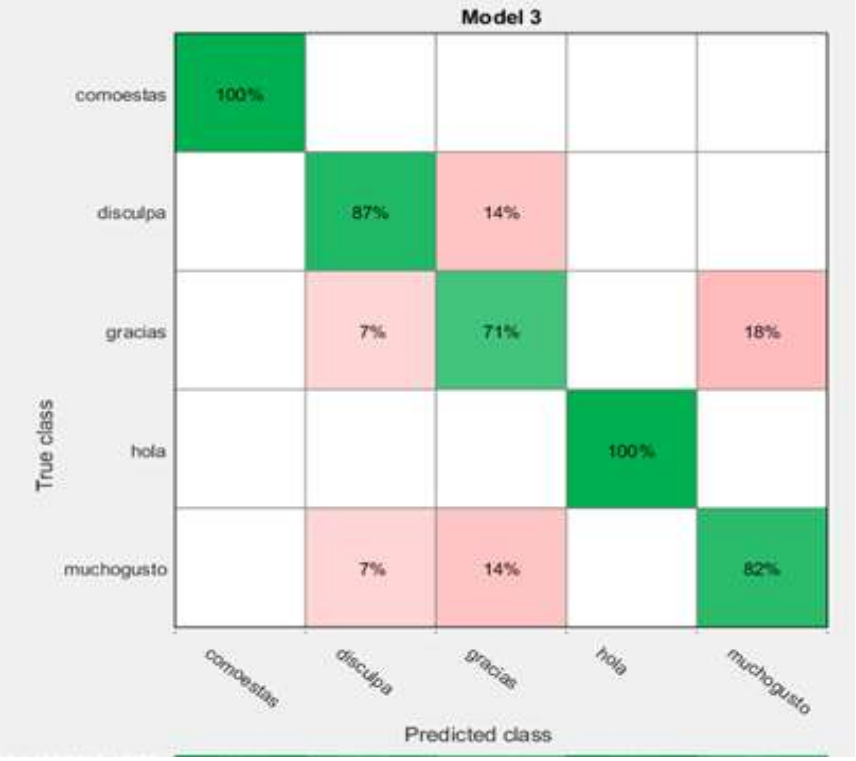


Figura 66 Matriz de Confusión Algoritmo SVM 4

5.1.5 Algoritmo SVM 5

El algoritmo SVM 5 se desarrolló utilizando el sistema cúbico de SVM, tomando en cuenta solo las articulaciones mano, hombro, codo, pulgar, muñeca y punta de dedo de las partes derecha e izquierda que son las partes del cuerpo utilizadas para realizar el signo y únicamente los *frames* característicos 17 y 40, que es el rango de *frames* en donde se produce el movimiento de la palabra, para un total de 154 características correspondiente a las muestras analizadas, en este caso se obtuvo una menor precisión en comparación al algoritmo 4 de 74,3% como se muestra 67.

Figura 67 Precisión algoritmo SVM 5

Los resultados de los porcentajes de aciertos y fallos del algoritmo se puede observar en la tabla 12. Se puede observar en la figura 68, en este modelo la palabra

"gracias" tiene una precisión menor al momento de entrenar y comparar con las palabras "disculpa" y "mucho gusto" con 43% y 14% de confusión respectivamente, por lo cual se recomienda aumentar el número de *frames* en el entrenamiento del algoritmo si se desea trabajar con SVM cúbico para aumentar su probabilidad de aciertos.

Tabla 12

Porcentaje de aciertos y fallos algoritmo SVM 5

Palabra	Porcentaje de aciertos	Porcentaje de fallos
Como estas	94%	6%
disculpa	64%	36%
gracias	43%	57%
hola	100%	0%
Mucho gusto	67%	33%

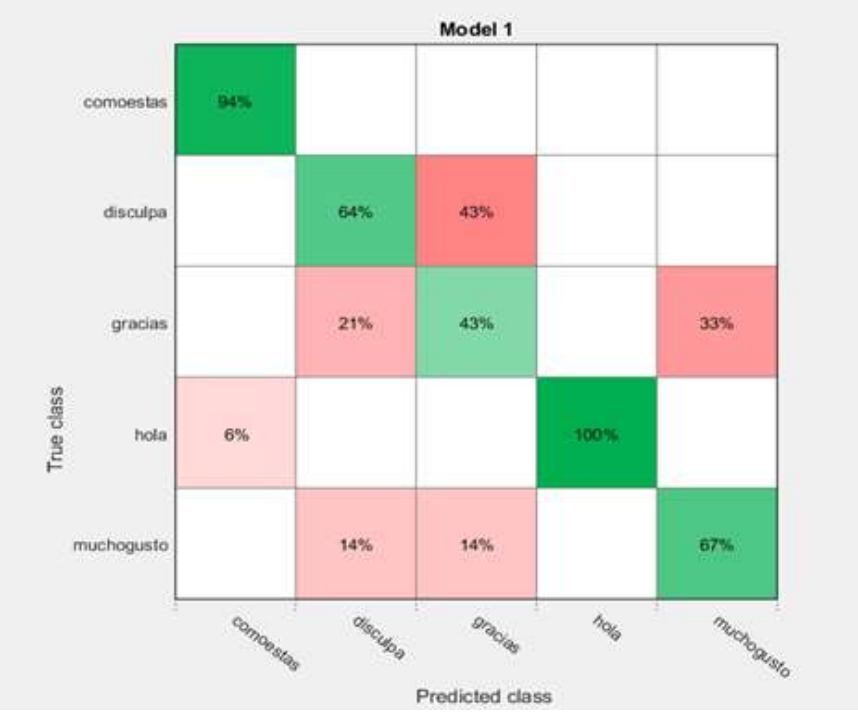


Figura 68 Matriz de confusión algoritmo SVM 5

5.1.6 Algoritmo SVM 6

El algoritmo SVM 6 se desarrolló utilizando el sistema cuadrático de SVM, tomando en cuenta solo las articulaciones mano, hombro, codo, pulgar, muñeca y punta de dedo de las partes derecha e izquierda que son las partes del cuerpo utilizadas para realizar el signo y únicamente los *frames* característicos 17 y 40, que es el rango de *frames* en donde se produce el movimiento de la palabra, para un total de 154 características correspondiente a las muestras analizadas, en este caso se obtuvo una mayor precisión en comparación al algoritmo SVM 5 de 77,1% como se muestra 69.

Figura 69 Precisión algoritmo SVM 6

Los resultados de los porcentajes de aciertos y fallos del algoritmo se puede observar en la tabla 13. Se puede observar que las los porcentajes se mantienen relativamente similares con el algoritmo SVM 5, sin embargo se puede observar en la figura 70, la palabra “gracias” tiene una precisión mayor en comparación al algoritmo SVM 5, obteniendo una menor confusión del 29% y 14% para las palabras “disculpa” y “mucho gusto” respectivamente, sin embargo la palabra “mucho gusto” presenta mayor probabilidad de error con la palabra “gracias” del 23%, por ello se recomienda aumentar el número de *frames* en el entrenamiento del algoritmo para mejorar la probabilidad de aciertos al trabajar con SVM cuadrático.

Tabla 13

Porcentaje de aciertos y fallos algoritmo SVM 6

Palabra	Porcentaje de aciertos	Porcentaje de fallos
Como estas	94%	6%
disculpa	71%	29%
gracias	57%	43%
hola	100%	0%
Mucho gusto	62%	38%

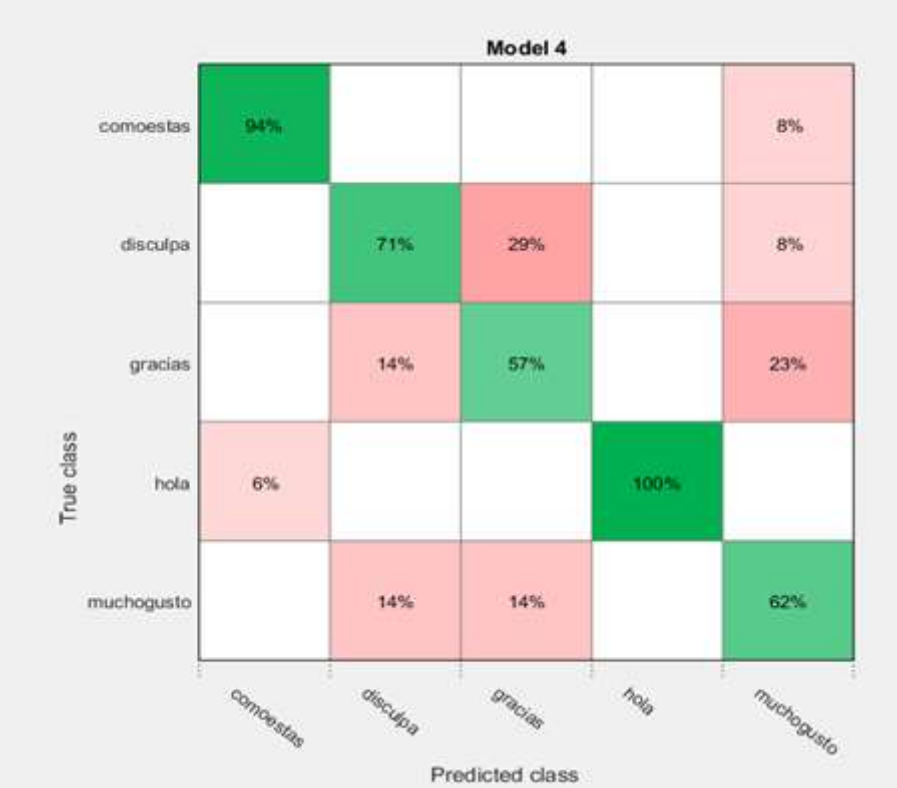


Figura 70 Matriz de confusión algoritmo SVM 6

5.1.7 Algoritmo SVM 7

El algoritmo SVM 7 se desarrolló utilizando el sistema lineal de SVM, tomando en cuenta las articulaciones mano, hombro, codo, pulgar, muñeca y punta de dedo de las partes derecha e izquierda que son las partes del cuerpo utilizadas para realizar el signo y el rango de *frames* característicos (donde se produce el movimiento de la palabra) de 17 a 40, para un total de 176 características correspondiente a las muestras analizadas, en este caso se obtuvo la mejor precisión del 91,1% en comparación a algoritmos anteriores como se muestra 71.

Figura 71 Precisión Algoritmo SVM 7

Los resultados de los porcentajes de aciertos y fallos del algoritmo se puede observar en la tabla 14. Como se puede observar en este modelo la precisión de todas las palabras es mayor a modelos anteriores, esto es debido que al tomar un rango de *frames* en donde se produce la palabra, los rasgos característicos de cada una como posición de manos son más significativos por lo cual al momento de entrenar el algoritmo se vuelve más sencillo diferenciarlos. Adicionalmente, como se muestra en la figura 72, el algoritmo todavía presenta errores para la palabra "mucho gusto" al tener similitudes con las palabras "gracias" y "como estas" del 10% para cada una, por lo cual es posible aumentar el número de muestras en el entrenamiento para mejorar la precisión del algoritmo.

Tabla 14

Porcentaje de aciertos y fallos algoritmo SVM 7

Palabra	**Porcentaje de aciertos**	**Porcentaje de fallos**
Como estas	100%	0%
disculpa	91%	9%
gracias	90%	10%
hola	92%	8%
Mucho gusto	80%	20%

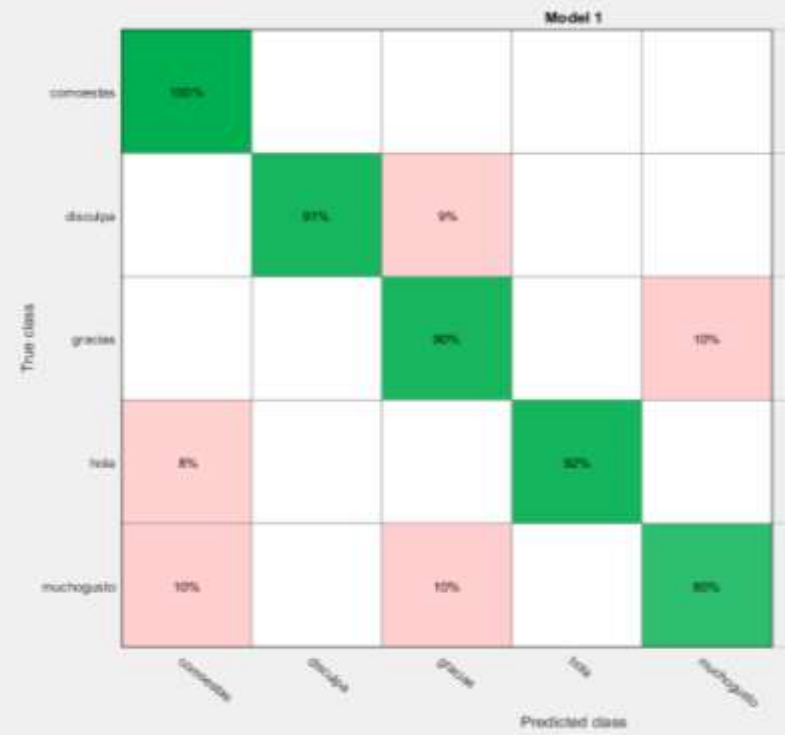


Figura 72 Matriz de confusión algoritmo SVM 7

En la figura 73 se muestra el tiempo de entrenamiento que se utilizó para entrenar al algoritmo y sus características:

Model number 1
Status: Trained
Accuracy: 91.1%
Prediction speed: ~59 obs/sec
Training Time: 8.1966 secs

Classifier
Preset: Linear SVM
Kernel function: Linear
Kernel scale: Automatic
Box constraint level: 1
Multiclass method: One-vs-One
Standardize data: true

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Figura 73 Características algoritmo SVM 7

5.1.8 Algoritmo SVM 8

El algoritmo SVM 8 se desarrolló utilizando el sistema cuadrático de SVM, tomando en cuenta las articulaciones mano, hombro, codo, pulgar, muñeca y punta de dedo de las partes derecha e izquierda que son las partes del cuerpo utilizadas para realizar el signo y el rango de *frames* característicos (donde se produce el movimiento de la palabra) de 17 a 40, para un total de 176 características correspondiente a las muestras analizadas, en este caso se obtuvo una menor precisión respecto al algoritmo 7 del 82,1% como se muestra en la figura 74.

Figura 74 Precisión algoritmo SVM 8

Los resultados de los porcentajes de aciertos y fallos del algoritmo se puede observar en la tabla 15. En este modelo, como se indica en la figura 75, la palabra "mucho gusto" tiene menor precisión en comparación al algoritmo anterior debido que presenta confusión con las palabras "como estas", "disculpa" y "gracias" con 10%, 10% y 20% respectivamente, por lo cual se sugiere trabajar con mayor número de muestras al entrenar el algoritmo para que sea capaz de distinguir con mayor facilidad entre signos mejorando la probabilidad de aciertos para la palabra "mucho gusto".

Tabla 15

Porcentaje de aciertos y fallos algoritmo SVM 8

Palabra	Porcentaje de aciertos	Porcentaje de fallos
Como estas	92%	8%
disculpa	91%	9%
gracias	70%	30%
hola	92%	8%
Mucho gusto	60%	40%

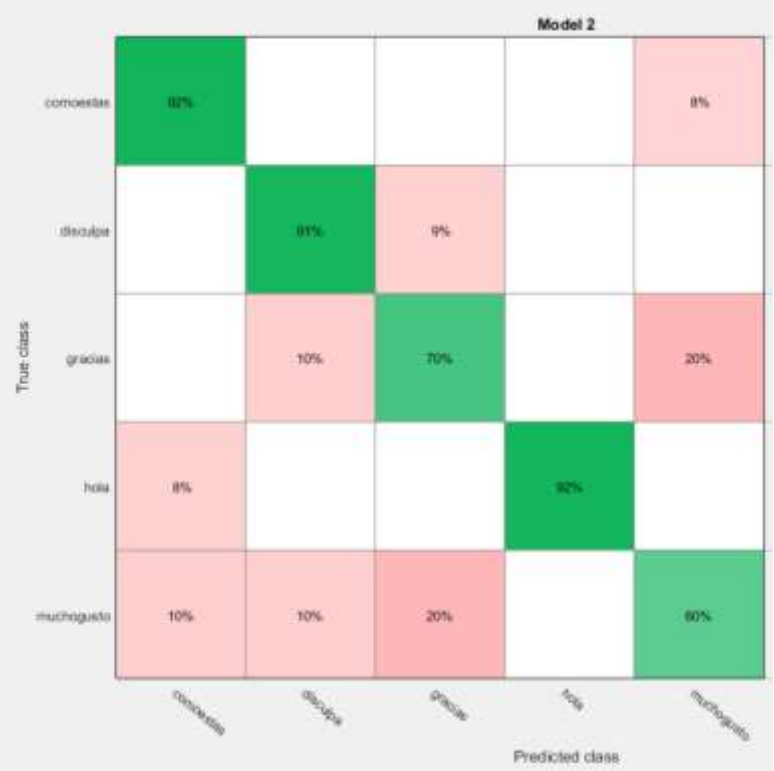


Figura 75 Matriz de confusión algoritmo SVM 8

5.1.9 Algoritmo SVM 9

El algoritmo SVM 9 se desarrolló utilizando el sistema cúbico de SVM, tomando en cuenta las articulaciones mano, hombro, codo, pulgar, muñeca y punta de dedo de las partes derecha e izquierda que son las partes del cuerpo utilizadas para realizar el signo y el rango de *frames* característicos (donde se produce el movimiento de la palabra) de 17 a 40, para un total de 176 características correspondiente a las muestras analizadas, en este caso se obtuvo la misma precisión respecto al algoritmo 8 del 82,1% como se muestra en la figura 76.

Figura 76 Precisión algoritmo SVM 9

Los resultados de los porcentajes de aciertos y fallos del algoritmo se puede observar en la tabla 16. Como se puede observar en la figura 77, en este modelo

"disculpa" tiene menor precisión al presentar mayor confusión con la palabra "gracias", por lo cual se sugiere emplear un mayor número de muestras al entrenar con la palabra "disculpa" en el algoritmo para que sea capaz de distinguir con mayor facilidad entre signos.

Tabla 16

Porcentaje de aciertos y fallos algoritmo SVM 9

Palabra	Porcentaje de aciertos	Porcentaje de fallos
Como estas	100%	0%
disculpa	82%	18%
gracias	70%	30%
hola	92%	8%
Mucho gusto	60%	40%

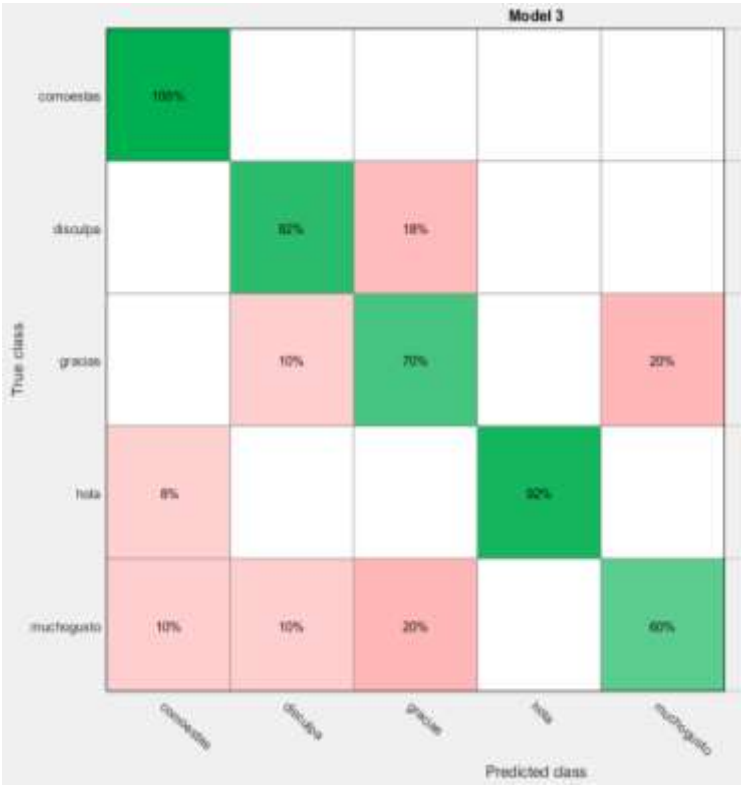


Figura 77 Matriz de confusión algoritmo SVM 9

5.1.10 Algoritmo SVM 10

El algoritmo SVM 10 se desarrolló utilizando el sistema lineal de SVM, tomando en cuenta las articulaciones mano, hombro, codo, pulgar, muñeca y punta de dedo de las partes derecha e izquierda que son las partes del cuerpo utilizadas para realizar el signo y el rango de *frames* característicos (donde se produce el movimiento de la palabra) de 17 a 40, para un total de 176 características correspondiente a las muestras analizadas, para las palabras con mayor probabilidad de error que son “disculpa”, “gracias” y “mucho gusto”, en este caso se obtuvo una precisión del 90,3% como se muestra en la figura 78.

Figura 78 Precisión algoritmo SVM 10

Los resultados de los porcentajes de aciertos y fallos del algoritmo se puede observar en la tabla 17. Como se puede observar en la figura 79, en este modelo la precisión de cada palabra aumento, esto es debido que la máquina tiene menor número de muestras que aprender, por lo cual es más sencillo para la máquina entrenar y diferenciar entre las 3 palabras obteniéndose así una mayor probabilidad de aciertos, por lo cual se sugiere que al emplear un mayor número de palabras en el entrenamiento de un algoritmo se aumente el número de muestras por palabra para tener una mejor precisión.

Tabla 17

Porcentaje de aciertos y fallos algoritmo SVM 10

Palabra	Porcentaje de aciertos	Porcentaje de fallos
disculpa	91%	9%
gracias	90%	10%
Mucho gusto	90%	10%

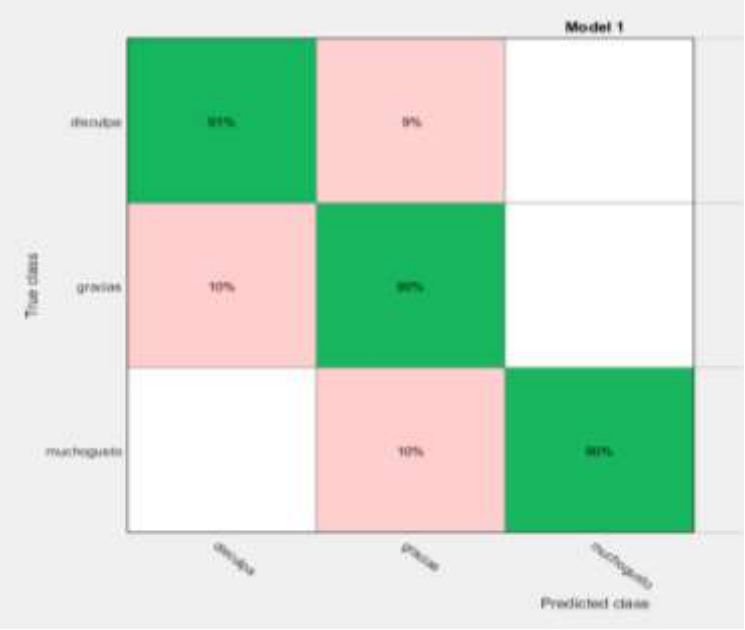


Figura 79 Matriz de confusión algoritmo SVM 10

5.1.11 Algoritmo SVM 11

El algoritmo SVM 11 se desarrolló utilizando el sistema lineal de SVM, tomando en cuenta las articulaciones mano, hombro, codo, pulgar, muñeca y punta de dedo de las partes derecha e izquierda que son las partes del cuerpo utilizadas para realizar el signo y el rango de *frames* característicos (donde se produce el movimiento de la palabra) de 17 a 40, para un total de 176 características correspondiente a las muestras analizadas, para las palabras del algoritmo 10 que son “disculpa”, “gracias” y “mucho gusto”, y la palabra “como estas”, en este caso se obtuvo una precisión del 90,9% como se muestra en la figura 80.

Figura 80 Precisión algoritmo SVM 11

Los resultados de los porcentajes de aciertos y fallos del algoritmo se puede observar en la tabla 18. Como se puede observar en la figura 81, en este modelo la probabilidad de aciertos de “disculpa”, “gracias” y “mucho gusto” se mantuvo igual respecto al algoritmo anterior, esto es debido que la palabra “como estas” al no presentar similitudes con dichas palabras es discriminada al entrenar el algoritmo de aprendizaje y la precisión de los aciertos se mantienen constante.

Tabla 18

Porcentaje de aciertos y fallos algoritmo SVM 11

Palabra	**Porcentaje de aciertos**	**Porcentaje de fallos**
Como estas	100%	0%
disculpa	91%	9%
gracias	90%	10%
Mucho gusto	80%	20%

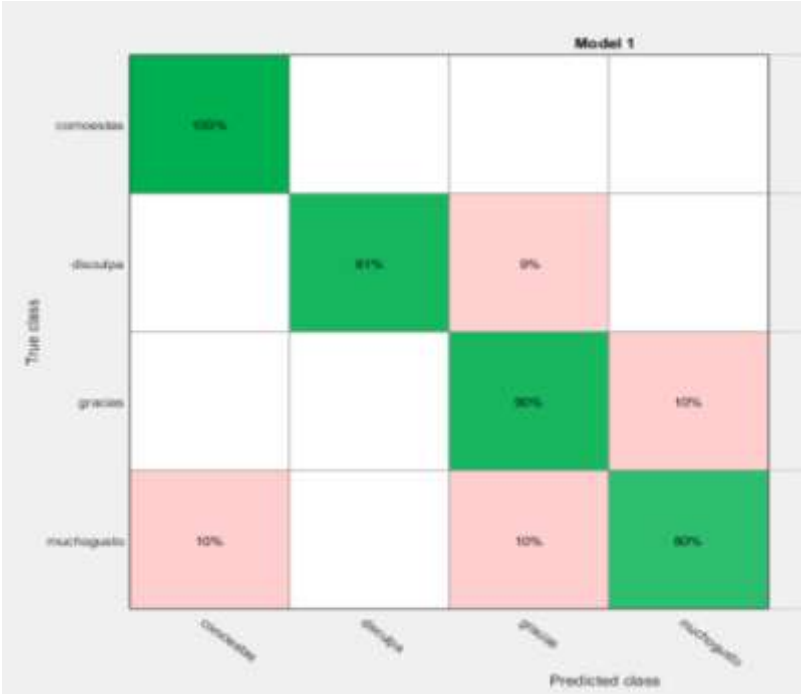


Figura 81 Matriz de confusión algoritmo SVM 11

5.1.12 Algoritmo SVM 12

El algoritmo SVM 12 se desarrolló utilizando el sistema lineal de SVM, tomando en cuenta las articulaciones mano, hombro, codo, pulgar, muñeca y punta de dedo de las partes derecha e izquierda que son las partes del cuerpo utilizadas para realizar el signo y el rango de *frames* característicos (donde se produce el movimiento de la palabra) de 17 a 40, para un total de 176 características correspondiente a las muestras analizadas, para las palabras “gracias” y “hola”, en este caso se obtuvo una precisión del 100% como se muestra en la figura 82.

Figura 82 Precisión algoritmo SVM 12

Los resultados de los porcentajes de aciertos y fallos del algoritmo se puede observar en la tabla 19. Como se puede observar en la tabla y la figura 83, en este modelo las palabras “gracias” y “hola” tienen 100% de aciertos cada una, esto es debido que los *frames* utilizados no tienen similitudes lo cual facilita al momento de entrenar el algoritmo a discriminar las muestras de dichas palabras volviendo sencillo el aprendizaje de máquina para la creación del algoritmo. Además de trabajar con menor número de resultados posibles, existe menor probabilidad error.

Tabla 19

Porcentaje de aciertos y fallos algoritmo SVM 12

Palabra	Porcentaje de aciertos	Porcentaje de fallos
Gracias	100%	0%
Hola	100%	0%

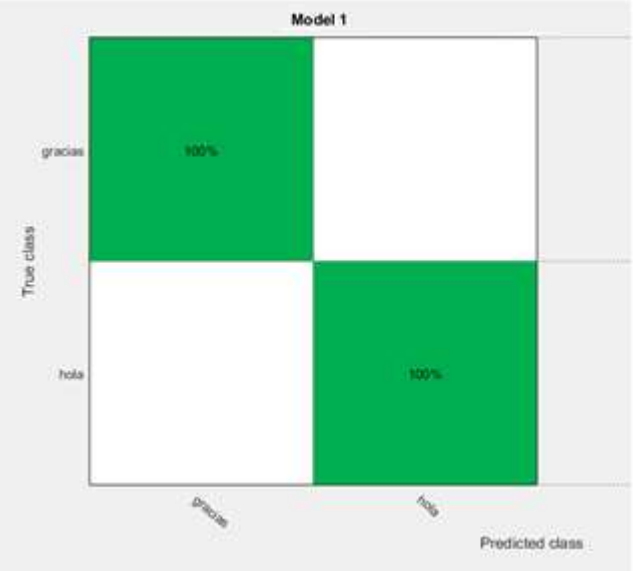


Figura 83 Matriz de confusión algoritmo SVM 12

A continuación se muestra la evaluación de los algoritmos para el reconocimiento de signos con muestras de la base de datos con su correspondiente análisis de los resultados.

5.2 Evaluación del algoritmo en palabras no consideradas para el entrenamiento de máquina

Como se explicó en el capítulo 4 para la evaluación es necesario guardar el algoritmo aprendido y colocarle un nombre, en este caso el nombre de los datos entrenados se llamará `traductorf`, el cual debe ser utilizado con la extensión `predictFcn(k)` donde k será la matriz donde se colocarán los datos extraídos de las características de Kinect v2, se requerirá evaluar el mismo número de *frames* utilizado para el aprendizaje de máquina de manera que no exista incongruencia al momento de evaluar los datos, finalmente transformar la matriz k en una tabla categorizando cada uno de los valores entregados por el dispositivo y utilizar la extensión `.Properties.VariableNames`. Se comprueba el algoritmo aprendido utilizando 5 palabras no aprendidas y 3 palabras aprendidas. A continuación se indica en las

siguientes tablas las palabra predicha, tipo de predicción, intervalo de corrección (de ser necesario) y observaciones conseguidas a partir de las pruebas realizadas con el algoritmo `traductorf`:

Tabla 20

Evaluación del algoritmo con la palabra "como estas"

Palabra “como estas”				
Persona	Palabra Predicha	Tipo de Predicción	Intervalo de Corrección	Observación
p15_r2	Comoestas	Correcta		
p19	Comoestas	Correcta		
p22	Comoestas	Correcta		
p24_r1	Comoestas	Correcta		
p24_r2	Comoestas	Correcta		
p1_r1	Comoestas	Correcta		
p18	Comoestas	Correcta		
p18_r2	Comoestas	Correcta		
	Porcentaje de aciertos (%)	100%		

La tabla 20 evalúa el porcentaje de aciertos de la palabra “como estas” mostrando que las palabras predichas son correctas, lo cual implica tener un 100% de aciertos y sin observaciones.

Tabla 21

Evaluación del algoritmo con la palabra "gracias"

Palabra “gracias”				
Persona	Palabra predicha	Tipo de predicción	Intervalo de corrección	Observación
p15_r2	Gracias	correcta		
p19	Gracias	correcta		
p22	Gracias	correcta		
p24_r1	Muchogusto	incorrecta	7-30	Predicción satisfactoria con corrección
p24_r2	Muchogusto	incorrecta	7-30	Predicción satisfactoria con corrección
p1_r1	Muchogusto	correcto		
p18	Gracias	correcta		
p18_r2	Gracias	correcta		
	Porcentaje de aciertos (%)	75%	100%	

La tabla 21 puntualiza el porcentaje de aciertos de la palabra “gracias” mostrando que las palabras predichas son correctas en un 75% (5) e incorrectas un 25% (3), en este caso lo que se procedió a realizar es un intervalo de corrección, es decir se varió el rango de evaluación para los 7 *frames* necesarios para la comparación con el algoritmo. Se colocó un rango de 7 a 30 para la persona 24 en su repetición 1 y 2 (p24_r1 y p24_r2 respectivamente) obteniendo una predicción satisfactoria al extraer los *frames* en ese rango. Haciendo hincapié en el algoritmo SVM 7. Existe una relación directa con la tabla de confusión mostrada en la figura 72 el cual indicaba que “gracias” era similar a “muchogusto”, demostrando que es posible eliminar dicha confusión si se realiza un aprendizaje para las palabras no acertadas en el intervalo de ejecución real y no estimada, o más número de muestras con los elementos que se tienen dentro de la base de datos.

El porcentaje de aciertos considerando las correcciones de los intervalos aumentó en 87.50% (7), las figuras (a)(b) del anexo 5 muestran el intervalo inicial y final de evaluación para el intervalo de predicción entre 17-40 y la figura (c)(d) del anexo 5

indica los *frames* comparados en el intervalo 7-30 para p24_r1, 24_r2 y p20-70 para p1_r1.

Tabla 22

Evaluación del algoritmo con la palabra "mucho gusto"

Palabra “mucho gusto”				
Persona	Palabra Predicha	Tipo de Predicción	Intervalo de Corrección	Observación
p15_r2	gracias	Incorrecta	27-50	Predicción satisfactoria con corrección
p19	gracias	Incorrecta	22-45	Predicción satisfactoria con corrección
p22	gracias	Incorrecta	22-45	Predicción satisfactoria con corrección
p24_r1	muchogusto	Correcta		
p24_r2	muchogusto	Correcta		
p1_r1	muchogusto	Correcta		
p18	muchogusto	Correcta		
p18_r2	muchogusto	Correcta		
	ACIERTOS%	62,50%	100%	

La tabla 22 ajusta el porcentaje de aciertos de la palabra “mucho gusto” mostrando que las palabras predichas son correctas en un 62.50% (5) e incorrectas un 37.50% (3), de manera análoga a la tabla 21 se coloca un intervalo de corrección, los cuales fueron 22-45 para la persona 19 (p19), persona 22 (p22) y 27-50 para la persona 15 repetición 2 (p15_r2). Para este caso se obtuvo un porcentaje de aciertos con corrección del intervalo del 100%, por lo que no es necesario añadir otro entrenamiento con estas palabras dentro del aprendizaje de máquina. Las figuras (a)(b) del anexo 6 muestran el intervalo inicial y final de evaluación para el intervalo predicción entre 17-40 y la figura (c)(d) del anexo 6 indica los *frames* comparados en el intervalo 22-45 para p19, p22y 27-50 para p15_r2.

Tabla 23

Evaluación del algoritmo con la palabra "disculpa"

Palabra “disculpa”				
Persona	Palabra predicha	Tipo de predicción	Intervalo de corrección	Observación
p15_r2	disculpa	correcta		
p19	gracias	incorrecta	ninguno	Incluir en el entrenamiento
p22	disculpa	correcta		
p24_r1	disculpa	correcta		
p24_r2	disculpa	correcta		
p1_r1	disculpa	correcta		
p18	disculpa	correcta		
p18_r2	disculpa	correcta		
	Prob. De Aciertos%	87,50%		

La tabla 23 revela el porcentaje de aciertos de la palabra “disculpa” mostrando que las palabras predichas son correctas en un 87.50% (7) e incorrectas un 12.50% (1), debido a que existe una similitud con “gracias” dentro de la ejecución y evaluación de la palabra no se incluye un intervalo de corrección, la figura (a)(b) del anexo 7 de la p19_r1 muestra el *skeleton tracking* en el intervalo 17-40, verificando la similitud de una palabra con otra, por lo que será necesario añadir este caso dentro del aprendizaje.

Tabla 24

Evaluación del algoritmo con la palabra "hola"

Palabra “hola”				
Persona	Palabra predicha	Tipo de predicción	Intervalo de corrección	Observación
15_r2	hola	correcta		
P19	hola	correcta		
P22	hola	correcta		
24_r1	hola	correcta		
24_r2	hola	correcta		
p1_r1	hola	correcta		
p18	hola	correcta		
p18_r2	hola	correcta		
	Porcentaje de aciertos (%)	100%		

La tabla 24 evalúa el porcentaje de aciertos de la palabra “hola” mostrando que las palabras predichas son correctas, lo cual implica tener un 100% de aciertos y sin observaciones.

Tabla 25

Porcentaje de aciertos

Tipo de evaluación	**Porcentaje**
Total de aciertos intervalo 17-40 *frames*	85,00%
Total de aciertos con intervalo de corrección	97,50%

La tabla 25 indica el porcentaje total de aciertos en el intervalo 17-40 *frames* y dentro de un intervalo de corrección, los cuales son 85.00% y 97.50% respectivamente. Corroborando que el algoritmo funciona para las palabras excluidas y determinando que la base de datos es válida, afirmando la posibilidad de utilizar la misma para un futuro reconocimiento de todas las palabras almacenadas considerando mejoras en su elaboración.

En esta sección se mostró una de varias posibilidades que se pueden realizar para el aprendizaje de máquina de manera simple únicamente con componentes del Kinect v2 utilizando la función de *skeleton tracking* para índices de profundidad sin considerar los estados de la mano, por lo cual es posible añadir estos en trabajos futuros y complementarlos con índices de color, imágenes de profundidad y color.

CAPITULO VI
CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

6.1 Conclusiones

- Se creó una base de datos utilizando el dispositivo Kinect v2 de Xbox One que dispone de 50 palabras dividas en “saludos”, “colores”, “adjetivos”, “alimentos” y “juguetes & cosas”, incluyendo el abecedario considerando las letras “ch”, “ll” y “rr” que son utilizadas en Ecuador, para el lenguaje de dactilológico español. La base de datos contiene un número de muestras total de 420 componentes. Adicionalmente, la base de datos se encuentra clasificada en videos de profundidad, videos de color (RGB) y datos de detección de articulaciones del cuerpo humano junto con los complementos adicionales ofrecidos por el sensor para el reconocimiento de estado de las manos.
- Se puede concluir que la base de datos es fiable y puede ser empleada para un futuro reconocimiento del lenguaje de señas español ecuatoriano, como se mostró en este trabajo al realizar un reconocimiento de 5 palabras diferentes del conjunto “saludos” obteniendo resultados favorables para cualquier tipo de persona (con o sin discapacidad auditiva).
- Mediante el análisis de videos de color y videos de profundidad, se determinó que el intervalo promedio de ejecución de una palabra es entre el frame 17 y 40, extrayendo un total de 7 frames para el entrenamiento de máquina realizado para la validación de la base de datos en este trabajo.
- Se determinó ejecutando varios algoritmos SVM para el reconocimiento del lenguaje de signos español ecuatoriano que el clasificador SVM lineal ofrecido por el *toolbox* de MATLAB proporciona un mejor rendimiento con un tasa de predicción del 91.1%. El algoritmo entrenado se realizó para 5 saludos que son “mucho gusto”, “gracias”, “como estas”, “disculpa” y “hola” utilizando un total de 13 muestras de video para las extremidades superiores del cuerpo humano

como mano, muñeca, codo, hombro, dedo anular y pulgar. Se utilizan únicamente componentes propios ofrecidos por Kinect v2 de Xbox One evidenciando que la base de datos es válida.

- Se determinó que existe semejanza entre ciertas palabras del lenguaje de signos español ecuatoriano por lo cual es necesario considerar los factores de movimientos de cada signo y distancia de las manos en el entrenamiento de máquina para el algoritmo de reconocimiento. En este caso, es necesario un re-entrenamiento del algoritmo añadiendo las muestras que presenten similitud.
- Se comprobó que la captura de datos debe ser a una distancia menor o igual a 3 metros ya que la luminosidad por parte del sol, u obscuridad total, causa una detección errónea de datos del *skeleton tracking*, lo cual implica realizar una nueva grabación.
- Se realizaron pruebas utilizando 5 muestras no aprendidas y 3 aprendidas para la validación de la base de datos, se observó que en el intervalo 17-40 *frames* existe un porcentaje de aciertos total del 85.00% de palabras probadas. Evaluando en un intervalo corregido (según el tiempo de ejecución de la palabra), el porcentaje de aciertos incrementó a un 97.50% fallando en la palabra “disculpas”, lo cual implica que es necesario un re-entrenamiento considerando la distancia de separación entre las manos.

6.2 Recomendaciones

- Se recomienda desarrollar un buscador para palabras deseadas dentro de la base de datos como se realizó en este trabajo, para las diferentes categorías existentes: “saludos”, “juguetes & cosas”, “colores”, “adjetivos”, “alimentos”, tanto para videos de color, videos de profundidad, y datos de *skeleton tracking.*
- Revisar mediante la vista previa del programa de adquisición de datos que el sensor de profundidad del dispositivo Kinect v2.0 se encuentre centrado respeto a la cámara, procurando que las personas no se encuentren dentro del alcance, evitando detección de cuerpos no deseados.

- Se recomienda que al aumentar las palabras para la base de datos revisar el video obtenido y comprobarlo con una persona especializada evitando el almacenamiento de datos erróneos como se realizó en este trabajo.
- Verificar que el *skeleton tracking* de la muestra esté acorde a la posición del signo capturado por el video de profundidad, como se utilizó en el trabajo con la opción de visualización por *frames* del programa de adquisición de datos, ya que evitaría futuros errores en el reconocimiento de signos.
- Se recomienda para tener diversidad en la creación de la base de datos que los signos tomados sean ejecutados con varias personas y realizados en varias sesiones, evitando una mala ejecución de signos causado por cansancio.

6.3 Trabajos futuros

Como trabajos futuros se propone la creación de una base de datos los cuales abarquen un mayor conjunto de palabras a parte de las empleadas en este proyecto, para tener una cantidad de información más completa del lenguaje español ecuatoriano.

Estas bases pueden incluir *face tracking* que es la detección de gestos faciales, empleando estos datos es posible realizar un reconocimiento más completo del lenguaje de signos debido que algunas palabras de todo el lenguaje español tienen gran similitud en su ejecución y es difícil diferenciar entre ellos utilizando únicamente *skeleton tracking*. Se incluye la posibilidad de incluir librerías de voz de MATLAB para que sean utilizadas para un traductor bidireccional, la cual tiene el propósito de interpretar las palabras mediante voz obteniendo en la salida el signo y viceversa.

BIBLIOGRAFÍA

Abe, S. (2010). Support Vector Machines for Pattern Classification. *springer*, 113-161, 227-303.

Arranz, F., Liu Yin, Q., & López Cámara, J. M. (2012). *Interacción persona-computador basada en el reconocimiento visual de manos.* Madrid: Universidad Complutende de Madrid.

Colmenares, G. (2010). Máquina de Vectores de Soport. *Inteligencia Artificial*, 1-11.

CONADIS, C. N. (Septiembre de 2017). *personas con discapacidades registradas*. Obtenido de http://www.consejodiscapacidades.gob.ec/estadistica/index.html

CONADIS, C. y. (2013). Manual práctico para intérpretes en lengua de señas ecuatoriana. En C. y. CONADIS, *Manual práctico para intérpretes en lengua de señas ecuatoriana* (pág. 81). Obtenido de Manual práctico para intérpretes en lengua de señas ecuatoriana: http://www.cordicom.gob.ec/wp-content/uploads/downloads/2016/04/Manual_de_interprete_de_lengua_de_senas_ecuatoriana.pdf

Criollo, G., Wilfrido, C., & Cargua, D. (2015). *Desarrollar un Sistema de Traducción del Lenguaje de Señas Utilizando el Sensor Kinect para las Personas Sordo Mudas.* Riobamba, Chimborazo: Escuela Superior Politécnica de Chimborazo. Obtenido de http://dspace.espoch.edu.ec/handle/123456789/4565

Estrada Jimenez, L. A. Diseño e Implementación de un prototipo para la traducción de lenguaje de señas mediante la utilización de guante sensorizado. *Dise{o e implementación de un prototipo para la traducción de lenguaje de señas mediante la utilización de guante sensorizado.* Escuela Politécnica Nacional, Quito, Ecuador.

Felzenswalb, P., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32, 9, 1627-1645.

FENASEC, & UTI. (26 de Marzo de 2016). *Consejo Nacional de Igualdad de Discapacidades.* Obtenido de conadis.gob.ec: http://plataformaconadis.gob.ec/diccionario/

FENASEC, F. N. (2012). Glosario Básico de lengua de Señas Ecuatoriano. En F. N. FENASEC, *Glosario Básico de lengua de Señas Ecuatoriano* (pág. 87). Quito.

Horsky, J. (2014). Kinect For Windows v2. *INFINITE BLOCG*, 1-4.

Keskin, C., Kirac, F., Kara, Y. E., & Akarun, L. (2013). Real time hand pose estimation using depth sensors. *Advances in Computer Vision and Pattern Recognition*, 119-137.

López, D. (22 de Marzo de 2015). *DavidLpez60*. Obtenido de slideshare: https://es.slideshare.net/DavidLpez60/kinect-v2-descripcin

Marcal, M. (28 de Enero de 2013). *Características Kinect 2*. Obtenido de http://www.kinectfordevelopers.com/es/2014/01/28/caracteristicas-kinect-2/

Mathworks. (1994). *Imshow*. Obtenido de https://www.mathworks.com/help/images/ref/imshow.html

MathWorks. (1994). *Matlab Products*. Obtenido de https://es.mathworks.com/products/matlab.html?s_tid=hp_products_matlab

Mathworks. (1994). *Save*. Obtenido de https://www.mathworks.com/help/images/ref/imsave.html

Mathworks. (1997). *Implay*. Obtenido de Mathworks Documentation: https://www.mathworks.com/help/images/ref/implay.html

Mathworks. (2006). *Documentation videoinput*. Obtenido de https://www.mathworks.com/help/imaq/videoinput.html

MathWorks. (2013). *Creación de apps con interfaces gráficas de usuario en MATLAB*. Obtenido de Creación de apps con interfaces gráficas de usuario en MATLAB: https://es.mathworks.com/discovery/matlab-gui.html

MathWorks. (Septiembre de 2015). *Acquire images and video from industry-standard hardware*. Obtenido de https://www.mathworks.com/products/imaq.html

MathWorks. (2015). *Support Package Installation*. Obtenido de Support Package Installation: https://www.mathworks.com/help/matlab/matlab_external/support-package-installation.html

Mathworks. (2016). *Classification Learner*. Obtenido de https://www.mathworks.com/products/statistics/classification-learner.html

Mathworks. (2016). *Detect the Kinect V2 Devices*. Obtenido de https://www.mathworks.com/help/supportpkg/kinectforwindowsruntime/ug/detect-the-kinect-v2-devices.html

MathWorks. (2016). *Documentation ROIPosition*. Obtenido de https://www.mathworks.com/help/imaq/roiposition.html

Mathworks. (2016). *System Requirements for MATLAB R2016a*. Obtenido de https://www.mathworks.com/support/sysreq.html

MathWorks. (2017). *Key Features and Differences in the Kinect V2 Support*. Obtenido de https://www.mathworks.com/help/supportpkg/kinectforwindowsruntime/ug/key-features-and-differences-in-the-kinect-v2-support.html

Microsoft. (2014). *Desarrollar con Kinect para Windows*. Obtenido de https://developer.microsoft.com/es-es/windows/kinect/develop

Morocho, L. M. (2016). Construcción de un prototipo para el reconocimiento y traducción del lenguaje de señas a texto utilizando el sensor kinect. En L. M. Morocho.

Murakami, K., & Taguchi, H. (1991). Gesture recognition using recurrent neural networks. *Proccedings of the SIGCHI on Human Factors in Computing Systems* (págs. 237-242). New York: ACM.

Oliveto, L. (11 de Noviembre de 2015). *Kinect hands on.* Obtenido de https://www.slideshare.net/LuigiOliveto/kinect2-hands-on

Otero, V. R. (2012). *Reconocimiento de localizaciones mediante Máquinas de Soporte Vectorial.* Madrid: Universidad Carlos III de Madrid. Departamento de Ingeniería de Sistemas y Automática.

Ruben. (28 de 09 de 2015). *arcoresearchgroup*. Obtenido de arcoresearchgroup web: https://arcoresearchgroup.wordpress.com/2015/09/28/modulekinectv2capture-aun-o-terminado/

Sun, C., Zhang, T., Bao, B.-K., Xu , C., & Mei, T. (2013). Discrimitanive exemplar coding for sign language recognition with Kinect. *IEEE Transactions on Cybernetics*, 43, 1418-1428.

Tianzhu, Z., Jing, L., Si, L., Changsheng , x., & Hanqing , L. (2009-2011). Boosted exemplar learning for action recognition and annotation. *IEEE Transactions on Circuits and Systems for Video Technology* , 853-866.

Vilela, M. J. (2005). *La dactilología, ¿qué, cómo, cuándo...?* . Obtenido de La dactilología, ¿qué, cómo, cuándo...? : http://www.uco.es/~fe1vivim/alfabeto_dactilologico.pdf

Villacis, A. E. (2014). *Aplicación de DSP´s para la Transcripción de Lenguaje de Señas a Texto.* Ambato.

Vogler, C., & Metaxas, D. (1997). Adapting hidden Markov models for ASL recognition by using three-dimensional computer vision methods. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics: Computional Cybernetics and Simulation* (págs. 156-161). Los Alamitos, CA: Vol 1. IEEE.

Wang, Y., Lan, T., & Mori, G. (2011). Discriminative figure-centric models for joint action localization and recognition. *Proccedings of the IEEE International Conference on Computer Vision (ICCV'11)* (págs. 2003-2010). Los Alamitos, CA: IEEE.

Yauri Vidalón, J. E., & De Martino, J. M. (2014). Reconocimiento Continuo de la Lengua de Señas Brasileña en el Área de Salud. *ResearchGate*, 3-5.

Zaforas, M. (2017). Machine Learning para dummies. *Blog Tecnología para Negocio*, 1-10.

Zafrulla, Z., Brashear, H., Thad, S., Hamilton, H., & Peter, P. (2011). American Sign Languaje recognition with the Kinect. *Proccedings of the 13th International Conference on Multimodal Interfaces.*, 279-286.

Zhang, T., Xu, C., & Sun, C. (2015). Latent support vector machine modeling for sign language recognition with Kinect. *ACM Trans. Intell. Syst. Technol*, 1-20.