UNIVERSIDAD NACIONAL DE LOJA

ÁREA DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

"CONSTRUCCIÓN DE UN PROTOTIPO PARA EL RECONOCIMIENTO Y TRADUCCIÓN DEL LENGUAJE DE SEÑAS A TEXTO UTILIZANDO EL SENSOR KINECT"

TESIS DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Autor: Leydi Maribel Mingo Morocho

Director: Ing. Juan Manuel Galindo Vera, Mg. Sc.

LOJA-ECUADOR

2016

CERTIFICACIÓN

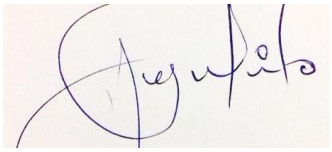
Señor Ingeniero
Juan Manuel Galindo Vera, Mg. Sc.
DIRECTOR DEL TRABAJO DE TESIS

CERTIFICA:

Que el presente proyecto fin de carrera elaborado previo a la obtención del Título de Ingeniería en Electrónica y Telecomunicaciones, titulado: **"Construcción de un prototipo para el reconocimiento y traducción del lenguaje de señas a texto utilizando el sensor Kinect."**, realizado por la estudiante Egresada **Leydi Maribel Mingo Morocho**, cumple con los requisitos establecidos por las normas generales para la graduación en la Universidad Nacional de Loja, tanto en aspecto de forma como de contenido.

Por lo tanto, autorizo proseguir los trámites legales para su presentación y defensa.

Loja, 06 de enero del 2016



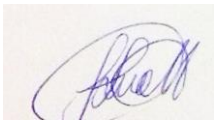
Ing. Juan Manuel Galindo Vera, Mg. Sc.

DIRECTOR DEL TRABAJO TESIS.

AUTORÍA

Yo **LEYDI MARIBEL MINGO MOROCHO** declaro ser autora del presente trabajo de tesis y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido de la misma.

Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi tesis en el Repositorio Institucional-Biblioteca Virtual.



Firma: ……

Cédula: 1105653792

Fecha: 07 de abril del 2016

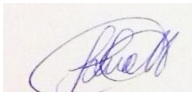
CARTA DE AUTORIZACIÓN DE TESIS POR PARTE DE LA AUTORA, PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO.

Yo LEYDI MARIBEL MINGO MOROCHO declaro ser autora de la tesis titulada: **"CONSTRUCCIÓN DE UN PROTOTIPO PARA EL RECONOCIMIENTO Y TRADUCCIÓN DEL LENGUAJE DE SEÑAS A TEXTO UTILIZANDO EL SENSOR KINECT",** como requisito para optar al grado de: **INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES**; autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional:

Los usuarios pueden consultar el contenido de este trabajo en el RDI, en las redes de información del país y del exterior, con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia de la tesis que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los siete días del mes de abril del dos mil dieciséis.



Firma: …….

Autor: Leydi Maribel Mingo Morocho.

Cédula: 1105653792

Dirección: Loja (Av. Isidro Ayora Km 2 ½ vía a la costa).

Correo Electrónico: leydi.mingo@gmail.com

Teléfono: 255-24-43 **Celular:** 0988943393

DATOS COMPLEMENTARIOS

Director de Tesis: Ing. Juan Manuel Galindo Vera, Mg. Sc.

Tribunal de Grado: Ing. Diego Vinicio Orellana Villavicencio, Mg. Sc.

Ing. Ximena Carolina Acaro Chacón, Mg. Sc.

Ing. Luis Enrique Chuquimarca Jiménez, Mg. Sc.

DEDICATORIA

La gratitud es una valor que engrandece al ser humano, por ello dedico este trabajo de Tesis a Dios, a la Virgen del Cisne, a mis padres, hermanos, compañeros y amigos, quienes fueron apoyo incondicional y pilar fundamental a lo largo de toda mi carrera.

A mi hermano Oscar que se ha convertido en un ejemplo de lucha y constancia y me enseña que pese a los problemas que nos presenta la vida siempre hay que sonreír.

A mis maestros quienes nunca desistieron al enseñarme e impartir sus conocimientos para hacer de sus estudiantes grandes profesionales.

Leydi Maribel Mingo Morocho

AGRADECIMIENTO

Le agradezco infinitamente a Dios por concederme fortaleza en los momentos de debilidad y por haberme acompañado y guiado a lo largo de toda mi carrera, la misma que ha sido llena de aprendizajes, experiencias y sobre todo muchas alegrías.

Les doy gracias también a mis padres y hermanos por apoyarme y confiar en mí en todo momento, por los valores que me han inculcado y por ser dignos ejemplos de vida a seguir.

Así mismo a agradezco a la Universidad Nacional de Loja y por ende a los maestros, quienes supieron impartir magnos conocimientos que serán de gran ayuda en la vida profesional, especialmente a mi director de Tesis, por la ayuda brindada para la finalización del mismo.

Leydi Maribel Mingo Morocho

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. TEMA

“CONSTRUCCIÓN DE UN PROTOTIPO PARA EL RECONOCIMIENTO Y TRADUCCIÓN DEL LENGUAJE DE SEÑAS A TEXTO UTILIZANDO EL SENSOR KINECT”

2. RESUMEN

En el presente trabajo de investigación se presenta un sistema para la traducción del lenguaje de señas ecuatoriano específicamente del abecedario de este lenguaje basado en técnicas de aprendizaje automático, para facilitar de cierta forma, la comunicación de las personas que padecen de algún tipo de discapacidad auditiva con aquellas que no manejan o conocen este lenguaje.

Para cumplir con el objetivo se combinaron ideas e investigaciones de varios autores con el fin de definir el procedimiento del que se constituye este prototipo, mismo que consta de cuatro fases: Detección de la mano, obtención de las características descriptivas de la misma, clasificación e interpretación de la información obtenida en las etapas anteriores.

Los datos adquiridos con Kinect mediante el sensor de profundidad se almacenan en una base de datos de entrenamiento para luego ser preprocesadas, adecuando su tamaño a uno fijo, y transformándolas a imágenes binarias de tal forma que el coste computacional sea el menor posible.

Los datos de entrenamiento son empleados para el entrenamiento de las máquinas de aprendizaje denominada Maquinas de Soporte Vectorial (SVM) , previa extracción de características a través del descriptor de Histogramas de Gradientes Orientados (HOG), una vez obtenido el detector se podrá aplicar a imágenes para identificar manos en ellas y realizar la traducción correspondiente.

La traducción se podrá visualizar en tiempo real a través de una Guide realizada en Matlab que muestra la seña realizada por el usuario, el redimensionamiento de la misma y finalmente la traducción que corresponde a la seña emitida originalmente.

2.1. ABSTRACT

On the following research work it is introduced a system the language of Ecuadorian signs specially the alphabet this language is based on automatic learning techniques, to facilitate in a way, the communication of people who suffer from a kind of hearing impairment for those who do not know this language.

In order to meet the objective a bunch of ideas and various authors investigations were combined in order to define the procedure of which this prototype is, it consists of four phases: hand detection, obtaining the descriptive characteristics of the same, classification and interpretation of the information obtained in previous stages.

The data acquired by Kinect depth were stored in a database so then it can be processed, adapting it to a fixed size, and transforming them into binary images so that the computational cost is as low as possible.

The training data is applied for child employee for the Training of Machine Learning called Support Vector Machine (SVM) after feature extraction through descriptor Histograms Oriented Gradients (HOG), once the detector is obtained images may be applied to identify hands and perform the translation accordingly.

The translation can be visualized in real time through a guide made in Matlab which shows the sign made by the user, resizing it and finally the translation corresponding to the signal originally issued.

3. INTRODUCCIÓN

En los últimos años la interacción entre el Computador y el Humano combinada con técnicas de visión por computador es uno de los temas que más despierta el interés dentro del campo investigativo, ya sea el reconocimiento de gestos o personas, ya que cada vez el ser humano pretende dotar de la capacidad de aprender a las computadoras con programas especializados de tal manera que puedan cumplir con funciones específicas asemejándose al pensamiento y comportamiento humano.

Existe una marcada tendencia a utilizar sistemas de reconocimiento gestuales que permiten la interacción de los usuarios con diversos dispositivos digitales, un ejemplo claro que se destaca es la guerra existente en el mundo del entretenimiento para crear y desarrollar este tipo de sistemas logrando que los dispositivos se adapten de un modo casi natural a los gestos de los usuarios, con esta premisa se encuentra hoy en día en el mercado el dispositivo Kinect que se basa principalmente en el uso de dos sensores para la realización de esta tarea: el sensor de color y el de profundidad.

Tales tecnologías se utilizan en la actualidad para muchos avances y desarrollos científicos, ya sea como ayuda en la medicina o en la robótica, pero principalmente investigadores y desarrolladores hacen uso de estos dispositivos para crear aplicaciones que sean útiles para la inclusión de ciertas poblaciones como las personas con algún tipo de discapacidad, específicamente con problemas auditivos, dentro de la sociedad en general.

Estas personas hacen uso del lenguaje de señas para comunicarse con las demás personas, pero no todas manejan o conocen este lenguaje, que es conocido también como lenguaje dactilológico; en el Ecuador según datos estadísticos del CONADIS aproximadamente 50.500 personas padecen de discapacidad auditiva, y la mayoría de estas se sienten aisladas de la sociedad en general.

Siguiendo esta tendencia, y con el fin de incorporar nuevas tecnologías como es el caso de Kinect, y de diseñar un sistema accesible económicamente al no requerir de hardware adicional y que ayude de cierta forma a que las personas con este tipo de discapacidad puedan incorporarse y logren ser entendidas por todas las personas ajenas a ellas, en el desarrollo del presente trabajo de investigación, se construyó un prototipo que reconoce

y traduce las letras del alfabeto del lenguaje de señas ecuatoriano en tiempo real, mostrando dicha traducción en una interfaz gráfica de usuario diseñada para tal fin en Matlab.

El proceso que fue llevado a cabo para conseguir la traducción de las señas que representan a cada letra de este lenguaje se basa en el procesamiento de una secuencia de imágenes (frames) obtenidas a partir del sensor de profundidad de Kinect para facilitar el procesamiento de las imágenes en cuanto a coste computacional se refiere; este proceso de dividió en cuatro fases principales: *Detección de la mano*, que en este caso fue de una persona diestra; *obtención de las características* descriptivas de la misma utilizando el método de Histogramas de Gradientes Orientados (HOG) para el cálculo de los gradientes de las imágenes previamente redimensionas y transformadas a imágenes binarias, para el entrenamiento de un modelo y la posterior *clasificación* haciendo uso de la técnica de aprendizaje automático denominada Maquinas de Soporte Vectorial (SVM) y finalmente se realizó la *interpretación* de la información obtenida en las etapas anteriores, es decir, se mostró la traducción de las letras correspondientes al lenguaje de señas ecuatoriano en una interfaz gráfica de usuario diseñada en Matlab lo más sencilla y amigable posible para que el usuario la pueda manejar con facilidad.

4. REVISIÓN DE LITERATURA

4.1. ANTECEDENTES

La comunicación es el medio por el cual dos o más personas pueden expresarse e interactuar, es decir es una transmisión de información, por medio de mensajes, entre emisor y receptor. Es por esto que el lenguaje de señas es un medio de expresión y comunicación para personas que sufren de deficiencias auditivas, dichas personas usan este lenguaje para poder transmitir sus ideas, pensamientos y demás información a los demás [1].

Según datos estadísticos del CONADIS actualmente en el Ecuador existen aproximadamente 401.538 personas que padecen algún tipo de discapacidad, entre ellas: visual, auditiva, física, intelectual o incluso varias de ellas a la vez, de las cuales 14.211 personas pertenecen a la provincia de Loja y 50.580 personas padecen de discapacidad auditiva [2].

4.1.1. Introducción al lenguaje de señas.

El Lenguaje de Señas o también conocido como lenguaje dactilológico es el medio de comunicación natural entre las personas sordas, y al igual que cualquier otro lenguaje, éste posee reglas y está estructurado en un código, de tal manera que las personas que lo usan, no sólo se expresan a través del cuerpo y de los gestos, sino también usando principalmente las manos con el objetivo de poder comunicarse con las personas a su alrededor [1].

Como todo tipo de lenguaje, tiene claramente definido las estructuras gramaticales, en donde se destaca un elemento importante que es la percepción visual, gestual y táctil, por medio de los cuales se establece un medio de comunicación con su alrededor. Este lenguaje ha pasado por cambios lingüísticos, al igual que todo sistema de lenguaje, lo que ha ocasionado su evolución y perfección en las distintas comunidades, lo que hace que existan más de 50 lenguas de señas, tal y como lo explica el "Sistema de Señas Internacionales (SII)" [3]. A continuación en la figura 1 se muestra todo el abecedario del lenguaje de señas ecuatoriano.

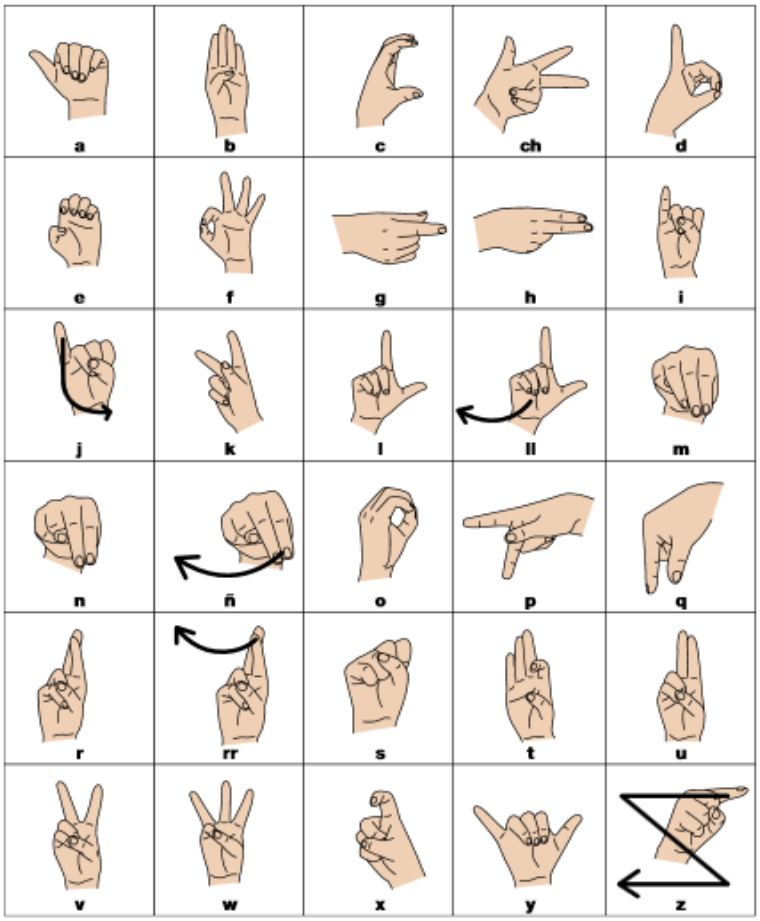


Figura 1. Lenguaje de señas ecuatoriano (LSEC) **[4]**.

4.1.2. Lenguaje dactilológico

Formalmente el término dactilología proviene del griego “daklitos”, que representa dedos y “logia”, ciencia, lo que significa “ciencia de los dedos”, es la representación manual de cada uno de las letras que componen el abecedario, es decir por medio de las manos el usuario ejecuta cada una de las señas lo que se conoce como deletreo manual logrando transmitir cualquier palabra a través del deletreo, el mismo que se realiza rápidamente por lo que resulta difícil diferenciar las letras individuales y generalmente la palabra se entiende por el conjunto de movimientos [1] [5].

4.1.2.1.¿Cómo se ejecuta la dactilología?

Para realizar la dactilología, se utiliza la mano dominante (derecha para los diestros, e izquierda para los zurdos), se procede a realizar la seña principalmente a la altura de la barbilla, complementándola con la articulación oral, por lo que es necesario que el rostro y la boca sean visibles [6]. En la figura 2 se puede observar claramente cómo debe ejecutarse cualquier seña, cabe recalcar que el brazo no debe hacer esfuerzo alguno, lo que significa que debe estar completamente relajado [1].

Figura 2. Posición del brazo (Dactilología) **[1]**.

Al momento de ejecutar cada una de las señas, existe cierto grado de libertad ya que no hay reglas que especifiquen que la palma de la mano deba siempre ir hacia el frente, hacia un lado o hacia a atrás, en lo que si se debe ser cuidadoso, es en el momento de ejecutar cada una de las señas, hacerlo de forma progresiva y continua, intercalando un cierto espacio de tiempo entre cada palabra, para evitar distorsiones en la información, cabe recalcar también que las letras dobles que se usan con más frecuencia se ejecutan haciendo un movimiento corto hacia la derecha tal y como se muestra en la figura 3 [1] .

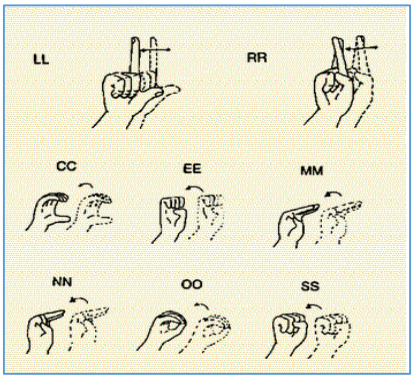


Figura 3. Representación de letras dobles frecuentes en el lenguaje de señas **[1]**.

4.1.2.2. Clasificación del lenguaje de señas.

El lenguaje de señas universal se clasifica en 4 aspectos básicos, los mismos que se detallan a continuación [1]:

- **Sistema Manual:** Es el conjunto de señales o formas de expresión lingüística mediante la utilización de una o ambas manos.
- **Sistema gestual – expresivo:** Es el conjunto de señales mediante el empleo de expresiones faciales o partes del rostro como: la boca, ojos, y lengua.
- **Sistema corporal:** Es el modo de expresión lingüística a través del uso de diversas partes del cuerpo exceptuando las manos, y las distintas partes del rostro, tales como: los pies, los hombros, el tronco, y el cuello.
- **Sistema de implementación de objetos:** Es la forma de lenguaje caracterizada, tal y como su nombre lo indica, por el uso de ciertos objetos o herramientas con el objetivo de disponer de un vocabulario mucho más exacto, amplio y simplificado.

4.1.3. La tecnología en el reconocimiento del lenguaje de señas

A medida que los años avanzan evoluciona claramente la tecnología y con ello la posibilidad de incluir dentro de los distintos ámbitos de la sociedad a las personas con algún tipo de discapacidad, especialmente las personas con discapacidad auditiva.

Hoy en día ya se cuenta con herramientas traductoras dotadas de software especializado que las personas sordas pueden usar para lograr comunicarse con las personas que no dominan el lenguaje de señas. Según explica Ernesto Compatangelo, profesor de ciencia de la computación de la Universidad de Aberdeen y fundador de Technabling, "El objetivo de la tecnología es darles a los usuarios de lenguaje de señas la posibilidad de superar los desafíos comunicacionales que experimenten a través de esta tecnología portátil" [7].

4.1.3.1. Traductor del lenguaje de Señas Portátil (PSLT)

Una de las aplicaciones que existe hoy en el mercado, es el traductor de lenguaje de señas portátil ***PSLT,*** por sus siglas en inglés, que mediante el uso de la cámara ya sea de un smartphone o de una laptop, se logra captar los movimientos que hace el emisor para su

posterior traducción, tal y como se muestra en la figura 4; cabe recalcar que detrás de este gran avance tecnológico están científicos informáticos de Technabling, una compañía surgida en la Universidad de Aberdeen, Escocia. [7].

Figura 4. PLST (Traductor del lenguaje de señas portátil) **[7]**.

4.1.3.2. Google Gesture

Otro gran avance se destaca por estudiantes de la Escuela de Comunicación Berghs en Estocolmo quienes junto a Google trabajan en una aplicación denominada "Google Gesture" para la traducción del lenguaje de señas a voz, esta app funciona con sensores que al ponerlos en el antebrazo, leerán los movimientos e impulsos de los músculos que se producen al ejecutar cada una de las señas, el objetivo de esta aplicación, según señala el video de Google Gesture, es "arreglar ese problema mediante la traducción del lenguaje de señas de manera inmediata, así no hay pausas en la conversación mientras las personas esperan a que la voz de audio se produzca", analizando la actividad muscular y la posición de la mano a través de una técnica denominada electromiografía para que se muestre la traducción en tiempo real en un Smartphone, en la figura 5 se muestra como se colocarían los sensores en el antebrazo del emisor [8].

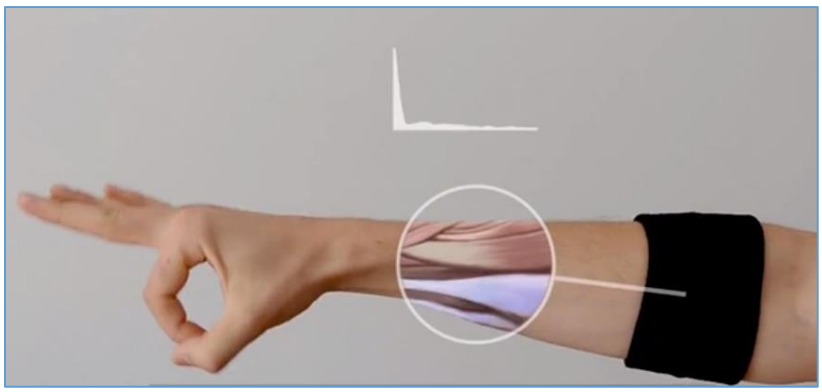


Figura 5. Sensor colocado en el antebrazo del emisor **[8]**.

4.2. PROCESAMIENTO DIGITAL DE IMÁGENES

El procesamiento digital de imágenes es un campo dotado a cambios constantes, debido a esto se destaca el hecho de que la evolución y el progreso de esta área se ha dado gracias al apoyo de otras áreas como las matemáticas, la computación y el profundo conocimiento de ciertas partes del cuerpo humano que actúan en la percepción y la manipulación de las imágenes; como resultado de éstas constantes variaciones, el procesamiento digital de imágenes se refleja en áreas como la medicina, astronomía, geología, microscopia, entre otras; gracias a estos numerosos avances se puede disfrutar de información meteorológica, transmisión y despliegue inmediato de diversas imágenes por internet [9].

El procesamiento digital de imágenes se encarga principalmente de mejorar el aspecto de las imágenes para resaltar en ellas características que se desean hacer notar, y de procesar dichas imágenes para su posterior almacenamiento, transmisión y representación; el procesamiento digital de imágenes consiste en dividir la imagen en un arreglo rectangular de elementos denominado pixel, se asigna entonces un valor numérico a la luminosidad promedio de cada pixel, de tal manera que los mencionados valores de luminosidad junto con sus coordenadas precisan la imagen por completo; el conjunto de elementos tanto software como hardware que procesan la señal visual lleva el nombre de Sistema de Procesamiento Digital de Imágenes [1] [10].

Para la manipulación de las imágenes, las matemáticas desempeñan un papel importante ya que tanto la computadora como los distintos algoritmos que se implementan sobre cada una de las imágenes ayuda a que el tratamiento de las mismas sea el correcto, para ello, los distintos métodos utilizados dentro del procesamiento digital de imágenes se basan en dos aspectos importantes como lo son: el mejoramiento de la información gráfica para la interpretación humana y el debido procesamiento de los datos de la imagen en donde se incluye la transmisión y/o almacenamiento de los mismos [9].

4.2.1. Imagen Digital

Una imagen digital consiste en la colección ordenada de valores que se representan en una matriz bidimensional $f(x,y)$ de cantidades finitas discretas, es decir, está compuesta de un número finito de elementos, cada uno con un lugar y valor específicos, dichos elementos conocidos con el nombre de pixel; como se muestra en la figura 6; es decir, se

refiere a una función de intensidad de luz en dos dimensiones, donde *x* e *y* indican las coordenadas espaciales y el valor de *f* en cualquier punto (x,y) es proporcional al nivel de gris de la imagen en ese punto; una imagen digital es entonces una imagen (función) *f(x,y)* que ha sido discretizada tanto en coordenadas espaciales como el luminosidad [9] [11] [12].

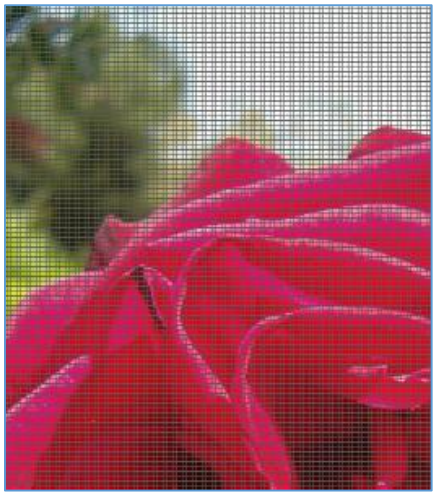


Figura 6. Descomposición de una fotografía en pixeles, a menor tamaño de los cuadros mayor precisión de la imagen **[13]**.

4.2.2. Imágenes Binarias

Como su nombre lo indica, las imágenes binarias se encuentran conformadas por pixeles que toman únicamente dos posibles valores, estos corresponden a 0 o 1, apagado o encendido, respectivamente. En la figura 7 se muestra una imagen binaria, donde se aprecia que la imagen es almacenada en un arreglo de pixeles de 1s o 0s [1].

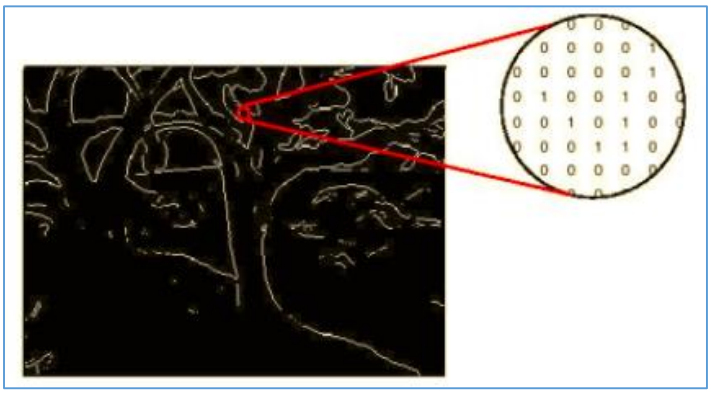


Figura 7. Ejemplo de imagen binaria **[14]**.

4.2.3. Modelos de color

Antes de empezar a describir los distintos modelos existentes para el tratamiento y procesamiento de las imágenes es necesario comprender los dos factores que motivan el uso del color dentro del procesamiento de imágenes, uno de esos factores manifiesta que el color es un descriptor que facilita la identificación de objetos y su extracción en una escena, en segundo lugar, los seres humanos somos capaces de discernir entre miles de tonalidades de color, es decir, lo que se compara con alrededor de 2 docenas de niveles de gris, por lo tanto, el objetivo de un modelo de color, conocido también como espacio de color o sistema de color, es el de facilitar la especificación de colores dentro de un modo estándar y detalla un sistema de coordenadas y el subespacio dentro de él donde cada color puede ser representado por un único punto [12]. A continuación se describe los modelos de color más comunes.

4.2.3.1. Modelo RGB

Como su nombre lo indica, las imágenes dentro del modelo RGB (Red, Green, Blue) están constituidas dentro de tres planos independientes, uno por cada color primario, de tal manera que cuando llegan a un monitor RGB, estas imágenes se combinan en la pantalla fosforescente para dar origen a una imagen en color compuesta; se resalta que la mayoría de los dispositivos que capturan las imágenes lo hacen utilizando el formato RGB, por lo que este tipo de modelo se hace importante dentro del procesamiento digital de imágenes; dentro de este modelo cada color presente aparece en sus componentes espectrales primarios: rojo, verde y azul [1]. En la figura 8 se aprecia el subespacio donde se representa los valores RGB y las demás variaciones en los colores a partir de los colores primarios, es decir, este modelo se basa en un sistema de coordenadas cartesianas que forma un cubo en el que los valores RGB están en las 3 esquinas, los valores de cian, magenta y amarillo están en las otras 3 esquinas, el negro está en el origen y el color blanco se encuentra en el vértice más alejado del origen, tal y como se observa en la figura 8 [12].

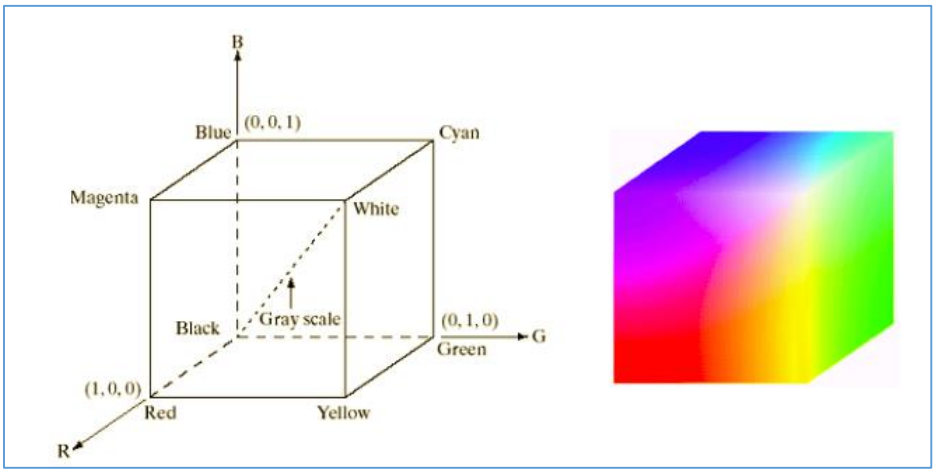


Figura 8. Diagrama esquemático del cubo RGB y el cubo a 24 bits **[12]**.

La escala de gris se extiende del negro al blanco en la línea marcada que une los puntos negro y blanco, tal y como se observa en la figura 8; los demás colores que corresponden al modelo son puntos sobre o dentro del cubo, y se definen por medio de vectores que se extienden desde el origen; las imágenes que se representan dentro de este modelo consisten en 3 imágenes componente que se adicionan de forma individual en forma de capas que se traslapan de tal manera que de esta combinación resulte una sola imagen a color, como se muestra en la figura 9 [12] [15].

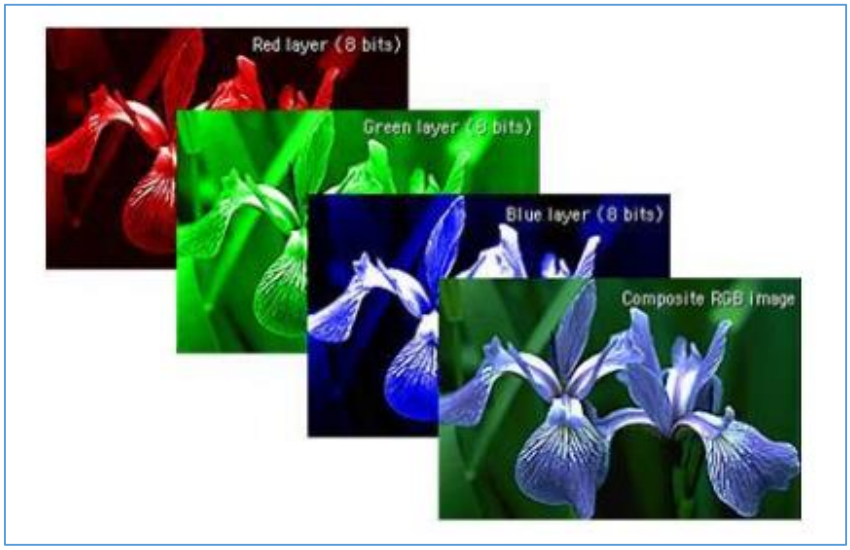


Figura 9. Adición de cada componente RGB para formar la imagen a color **[15]**.

4.2.3.2. Modelo CMY

Este modelo representa a los colores secundarios de la luz, o también conocidos como colores primarios de los pigmentos. A los colores CMY (Cian, magenta y amarillo) se los usa como filtros para sustraer colores de la luz blanca; dispositivos como impresoras y fotocopiadoras en color, necesariamente toman como entrada CMY o hacen uso de una conversión interna de RGB a CMY [1]. En la figura 9 se puede apreciar la representación de la distribución de los demás colores, con la única variante, respecto al modelo RGB, que donde estaba el color negro ahora existe el blanco y viceversa, caber recalcar que el sistema coordenado es el mismo que el modelo RGB, es decir, representa un sistema cartesiano.

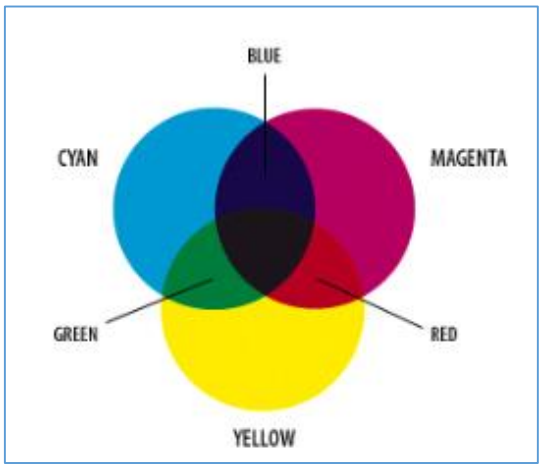


Figura 10. Modelo CMY **[16]**.

4.2.3.3. Modelo YIQ

El YIQ es una recodificación del modelo RGB por mostrar eficiencia en la transmisión y para la compatibilidad con los estándares de televisión en blanco y negro, es decir, el modelo YIQ se utiliza en las emisiones comerciales de televisión; cabe recalcar que la componente Y del sistema es la que brinda toda la información del video que necesita un sistema de televisión monocromático [16].

En la siguiente imagen (figura 11) se muestra la distribución de este modelo en el plano.

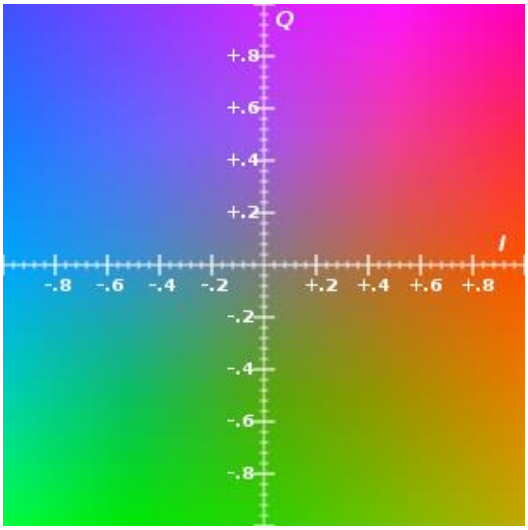


Figura 11. Modelo YIQ **[16]**.

El modelo YIQ fue diseñado para aprovechar la mayor sensibilidad del sistema visual humano a las variaciones de la saturación, de esta manera los estándares YIQ emplean más bits para representar la Y y menos para representar la I o Q; en el procesamiento de imágenes, este modelo presenta una gran ventaja frente a los demás, la luminancia (Y) y la información de color (I y Q) se encuentran desacopladas, lo que implicaría que la componente de luminancia de una imagen puede procesarse normalmente sin afectar a su contenido cromático [16].

4.2.3.4. Modelo HSI

Como se ha podido apreciar en cada uno de los modelos vistos (RGB y CMY), cambiar de un modelo a otro es fácil por lo tanto crear los colores también ya que ambos modelos usan coordenadas cartesianas, sin embargo, estos tipos de modelos de color no son convenientes para describir colores en términos prácticos para que el ser humano los pueda interpretar; esto significa que cuando la persona ve un objeto de color, lo describe por su tono, saturación y brillo (hue, saturation, intensity: tono, saturación e intensidad respectivamente), de ahí el nombre de este modelo conocido como *modelo de color HSI* (ver figura 12) que separa el componente de intensidad de la información de color en una imagen de color, como resultado, este modelo es una herramienta ideal para desarrollar algoritmos basados en descripciones de color naturales e intuitivas para los seres humanos [12].

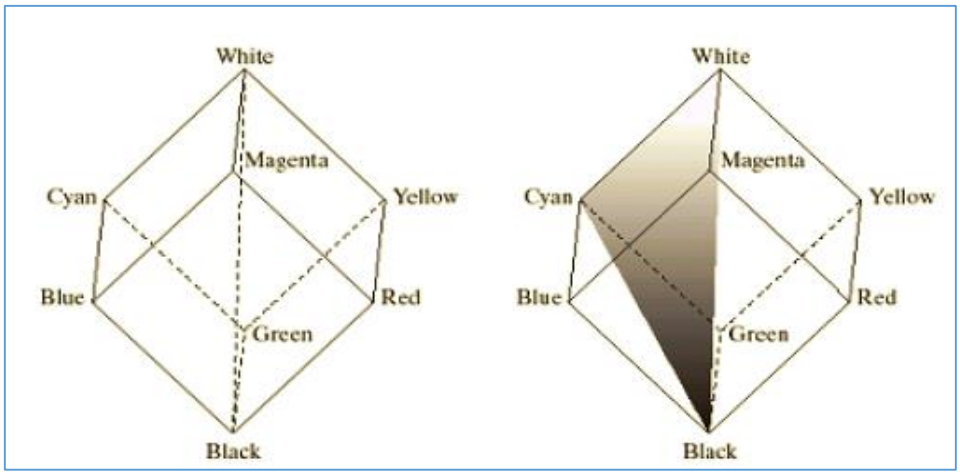


Figura 12. Deducción del modelo HSI a partir del cubo RGB **[12]**.

Para el desarrollo de este trabajo se usó, además de las imágenes binarias, imágenes RGB que corresponden al modelo del mismo nombre, para la realización de las respectivas pruebas del prototipo, sin embargo, como se mostrará en el apartado de resultados, éstas imágenes también se transforman a imágenes binarias dado que el procesamiento en este tipo de imágenes requiere de menos coste computacional.

4.2.4. Formatos de Imágenes

Básicamente, los formatos de las imágenes son archivos en los cuales se guarda información que conforma una imagen, cada formato es independiente, para elegir entre cualquiera de los formatos existentes es necesario conocer las posibilidades que ofrece con respecto a la gama de colores, a la compatibilidad, a la rapidez de carga, etc.; con respecto a la estructura, la mayoría posee una cabecera que indica al programa que lo solicite, las características de la imagen que almacenan; por ejemplo su color, tipo, resolución, etc., cada formato tiene una organización propia de su estructura [14].

Entre los formatos que más se destacan están: BMP, GIF, JPG, TIF y PNG; en el presente trabajo de investigación se hará uso del formato PNG, ya que en comparación con los demás, éste formato no tiene pérdida de información y permite que cada pixel tenga un nivel distinto de transparencia [14]. A continuación se describirá más detalladamente este tipo de formato.

4.2.4.1. PNG

El formato PNG (del inglés *Portable Network Graphics*, Gráficos de red portátiles o formato Ping) es un formato de archivos de gráficos de mapa de bits (una trama), se lo desarrolló como una alternativa gratuita al formato GIF para evitar la pérdida de información al comprimir los archivos y es un formato con creciente popularidad para desarrolladores web; PNG es una especificación de libre uso que no requiere pagar licencia de ningún tipo en aplicaciones que usen, creen o editen este tipo de imágenes [17].

El formato PNG permite almacenar imágenes en blanco y negro (profundidad de color de 16 bits por píxel) y en color real (profundidad de color de 48 bits por píxel), así como también imágenes indexadas, utilizando una paleta de 256 colores, además, soporta la transparencia de canal alfa, es decir, la posibilidad de definir 256 niveles de transparencia, mientras que por el contrario el formato GIF permite que se defina como transparente sólo un color de la paleta; como se había mencionado es un formato que se creó con el fin de evitar las pérdidas y lo hace ofreciendo una compresión sin pérdida de 5 a 25% mejor que la compresión GIF. A continuación en la tabla 1 se muestra la comparativa existente entre los distintos tipos de formatos [17].

Tabla 1. Comparación de los distintos formatos de imágenes **[17]**.

FORMATO	PROFUNDIDAD DE COLOR	MODOS DE COLOR	CANALES ALFA	COMPRESIÓN
BPM (.bmp)	1 bit: Blanco y negro. 4-8 bits: Escala de grises. 8 bits: Color indexado. 24 bits: Color RGB.	• RGB • Color indexado • Escala de grises • Mapa de bits.	No	Si: RLE en 4 y 8 bits.
TIF (.tif)	32 bits	• Mapa de bits • Color indexado • Escala de grises • RGB	Si	Si: (LZW)

GIF (.gif)	8 bits (256 colores)	• Mapa de bits • Color indexado • RGB	No	Si: (LZW)
JPEG (.jpg, .jpe)	24 bits	• Escala de grises • RGB • CMYK	No	Si: Con pérdidas.
PNG (.png)	24 bits	• RGB • Color indexado • Escala de grises • Mapa de bits	Si	Si: Sin pérdidas.

4.2.5. Histograma de una imagen digital

Dada una imagen es posible contar el número de pixeles que corresponden a cada tono en cada canal, a la representación gráfica de esta característica se le conoce como Histograma, es decir, es la representación objetiva de la imagen en cuanto a luminosidad y exposición se refiere; para comprender de mejor manera que es el histograma de una imagen y cuál es el procedimiento que se sigue para encontrarlo, se parte de la imagen mostrada en la figura 13 [18].

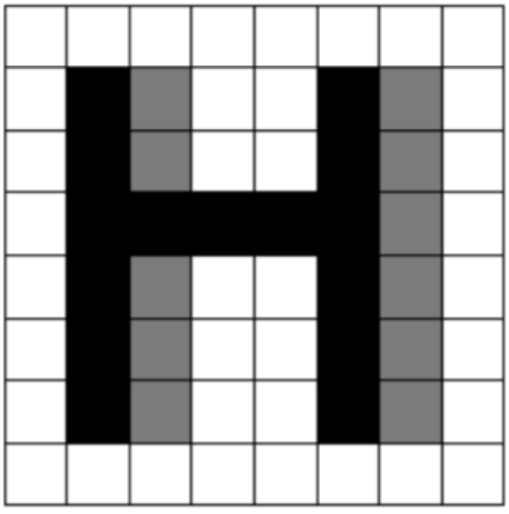


Figura 13. Imagen de 8x8 pixeles **[18]**.

La figura 13 muestra una imagen muy simple con un tamaño de 8x8 pixeles claramente identificados, solo son posibles 4 niveles de gris puesto que se van a usar 2 bits para codificar el brillo de cada pixel, por lo tanto los niveles de gris se enumeran del 0 al 3 tal y como se muestra en la tabla 2, correspondiendo un brillo mayor a los valores más altos.

Tabla 2. Identificación de los niveles de gris posibles de la figura 13.

Nivel de gris	Brillo
0	Negro
1	Gris oscuro
2	Gris claro
3	Blanco

Por lo tanto la gráfica que aparece en la figura 14 es el histograma que corresponde a la figura 13.

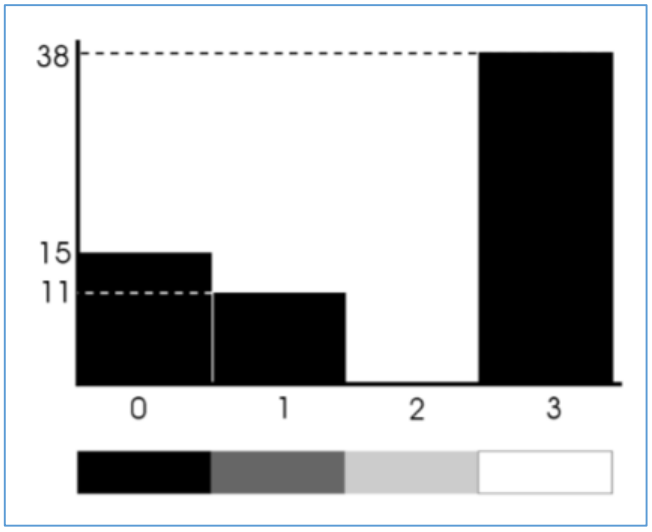


Figura 14. Histograma de la figura 13 **[18]**.

Los números que están en el eje horizontal representan los niveles de gris que pueden aparecer en la imagen, a la izquierda se encuentra el valor más oscuro (negro) mientras que en el extremo derecho está el más claro (blanco), el resto de niveles se distribuyen uniformemente; la altura de cada barra representa en cambio el número de pixeles de la imagen que presenta ese nivel de gris en concreto, por lo tanto se deduce que la imagen tiene 15 pixeles completamente negros, 11 tonos gris oscuro y 38 pixeles completamente

blancos, todos estos valores deberán sumar un total de 64, que es el número total de pixeles que tiene la imagen [18].

Para obtener el histograma de cualquier imagen en particular, existe hoy en día varias herramientas y programas que facilitan la obtención del mismo, como por ejemplo Gimp que es comúnmente utilizado por su sencillo manejo.

4.3. DISPOSITIVO KINECT XBOX 360

4.3.1. Breve historia

Kinect es un dispositivo que utiliza la tecnología de la cámara de profundidad desarrollada por la empresa israelí PrimeSense, de esta manera Microsoft en el año 2009 anunció públicamente el lanzamiento de este dispositivo bajo el nombre de Project Natal gracias a que Alex Kipman, director de Microsoft, decidiera ponerle el nombre de la ciudad brasileña Natal como un homenaje a su país de origen y porque la palabra natal significa «de o en relación al nacimiento», lo que refleja la opinión de Microsoft en el proyecto como «el nacimiento de la próxima generación de entretenimiento en el hogar». [19] [20].

En noviembre de 2010 Industrias Adafruit ofreció una recompensa para aquella persona que lograra desarrollar un controlador de código abierto para Kinect y fue el español Héctor Martin quien usando ingeniería inversa logró tal objetivo, dicho controlador permite el uso de la cámara RGB y de las funciones de profundidad. Hoy en día Kinect cuenta con gran aceptación y popularidad comercial, no solo en el ámbito de los videojuegos sino también siendo objeto de numerosos estudios aplicados a proyectos que ayudan al desarrollo de distintos ámbitos de la sociedad [19].

4.3.2. Arquitectura interna

La tecnología Kinect fue creada por Microsoft para la consola Xbox 630 con el fin de interactuar con el videojuego sin la necesidad de usar algún tipo de control remoto, es decir, basta que el usuario se coloque delante del mencionado dispositivo para que éste funcione detectando los movimientos, viendo de tal manera la escena en tres dimensiones en tiempo real, todo esto Kinect lo logra gracias al uso de dos sensores: una cámara de

color y un sensor de profundidad. Así mismo, para la localización de fuentes acústicas, y filtrado de ruidos, dicho dispositivo cuenta con un arreglo de 4 micrófonos [19] [21].
En la figura 15 se puede observar las partes de las que está formado Kinect, tal y como se mencionó en el párrafo anterior.

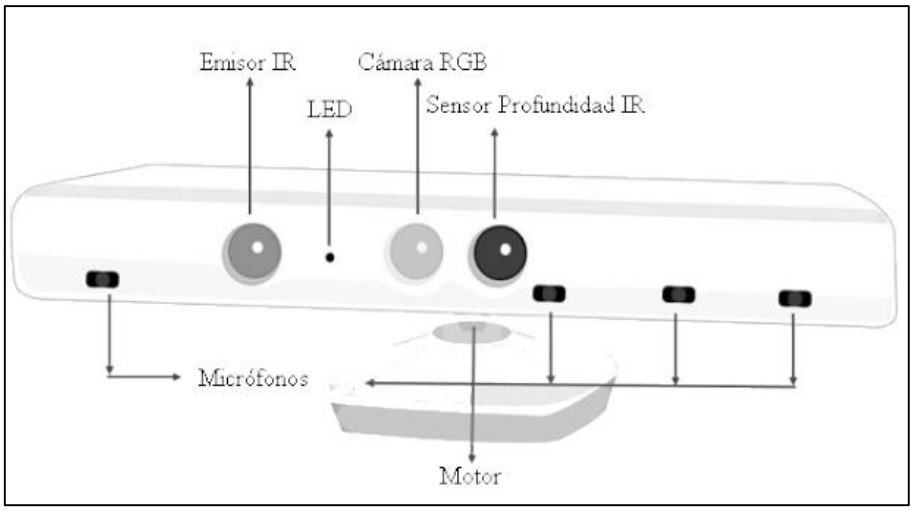


Figura 15. Componentes del sensor Kinect **[22]**.

Los dos sensores principales están situados en la parte frontal, tal y como se aprecia en la figura 15, los objetos que se encuentran dispuestos a los costados (Emisor IR y Sensor Profundidad IR - Receptor IR) conforman, como su nombre lo indica, el sensor de profundidad que genera los datos de profundidad, mientras que la lente que se encuentra en el centro forma parte de la cámara a color y posee una resolución de 640x480 [19].
A continuación se describe detalladamente, las características que sobresalen de cada uno de los componentes de Kinect [23]:

- ✓ **CÁMARA RGB:** Almacena los datos de los tres canales que envían los sensores, lo que permite la captura de imágenes a color y el envío de datos a un frecuencia de 30 fps (frames por segundo).

- ✓ **MICRÓFONO DE MÚLTIPLES MATRICES:** Arreglo de cuatro micrófonos que permite la localización de la fuente acústica y el descarte del ruido ambiental, dando la posibilidad también de grabar audio.

- ✓ **LUZ LED:** Indica si el dispositivo Kinect está listo para usarse, si funciona normalmente está en luz verde, pero si existe algún cambio brusco o si el sensor esta aun adquiriendo datos esta luz pasara a ser roja.

- ✓ **MOTOR:** Permite que Kinect pueda moverse en sentido vertical, hacia arriba o hacia abajo aproximadamente con un ángulo de 30°, con la finalidad de calibrar cada espacio concreto por lo que la altura máxima está recomendada a uno o dos metros.

- ✓ **SENSOR 3D DE PROFUNDIDAD:** Resulta de la combinación de un proyector de infrarrojos láser y un sensor de imagen CMOS (sensor de profundidad); el que se encarga de emitir los haces de luz infrarrojas es el proyector de infrarrojos laser mientras que el sensor de imagen CMOS se encarga de captar dichos haces de luz reflejados hacia el sensor de tal manera que sean convertidos en información de profundidad midiendo la distancia entre un objeto y el sensor.

- ✓ **SENSOR DE IMAGEN CMOS PARA RGB:** Capta las coordenadas de los ejes X e Y, es decir, se emplea para la captura de la resolución espacial, como entrada se usa RGB (Rojo, Verde, Azul) para proporcionar color a las imágenes que se capturan con el sensor de profundidad. En la figura 16 se puede apreciar claramente este componente.

Figura 16. Sensores que conforman Kinect **[23]**.

En la figura 17 se muestra un diagrama de los componentes de hardware del procesador de imagen de Kinect, ahí se puede apreciar claramente el chip de PrimeSense PS1080, el mismo que se encarga de generar y sincronizar las imágenes tanto de profundidad como de color, además a través del sistema de proyección de patrones de puntos denominado LightCoding, el chip logra ejecutar los algoritmos necesarios para adquirir imágenes, es resumen este chip permite reconstruir una captura de movimiento 3D de la escena que esté ubicada al frente de Kinect ya que captura su entorno en tres dimensiones y transforma esas imágenes sincronizadas en 3D [21] [23].

Cabe resaltar que los dos sensores tanto el proyector de infrarrojos como el sensor de profundidad trabajan en conjunto con el Chip PrimeSense PS1080.

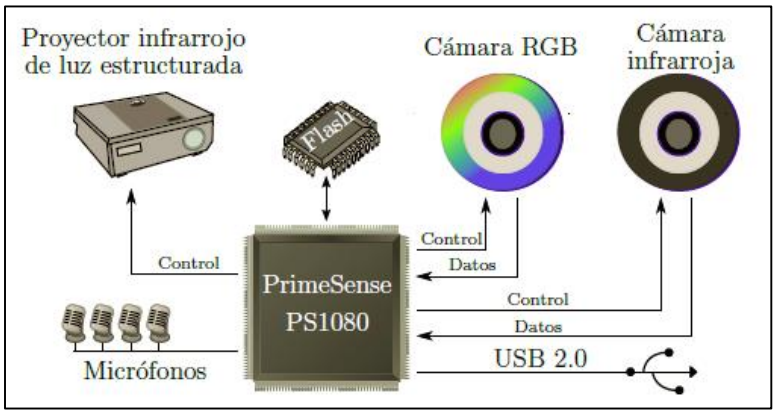


Figura 17. Hardware del dispositivo Kinect **[21]**.

Tal y como se mencionó anteriormente, el sensor de profundidad se compone por el emisor IR y el receptor IR, de tal manera que los datos de profundidad se obtienen en base al tiempo que tarda en emitir el láser, rebotar en un objeto y llegar hasta el receptor, es decir, el emisor IR emite rayos infrarrojos y el sensor de profundidad lee esos rayos reflectados y los convierte en información de la distancia entre el sensor y el objeto, gracias a este proceso es posible la adquisición de la imagen de profundidad [19] [21].

El arreglo de 4 micrófonos también forma parte importante dentro de la arquitectura de Kinect, ya que captura la información de audio, la graba y determina la dirección de donde proviene la fuente de audio, mientras que el motor permite una inclinación aproximada de ±27° que se suma a la visión vertical de la cámara [21].

Aunque no esté a simple vista, dentro de la estructura de hardware de Kinect también se encuentran los siguientes componentes:

- ✓ **ACELERÓMETRO:** Brinda estabilidad a las imágenes cuando el sensor se mueve.
- ✓ **MEMORIA RAM:** 512 Mb.
- ✓ **VENTILADOR:** Sirve para dar enfriamiento al dispositivo.

En la figura 18 se muestra las placas y circuitos que resaltan dentro del hardware del dispositivo Kinect.

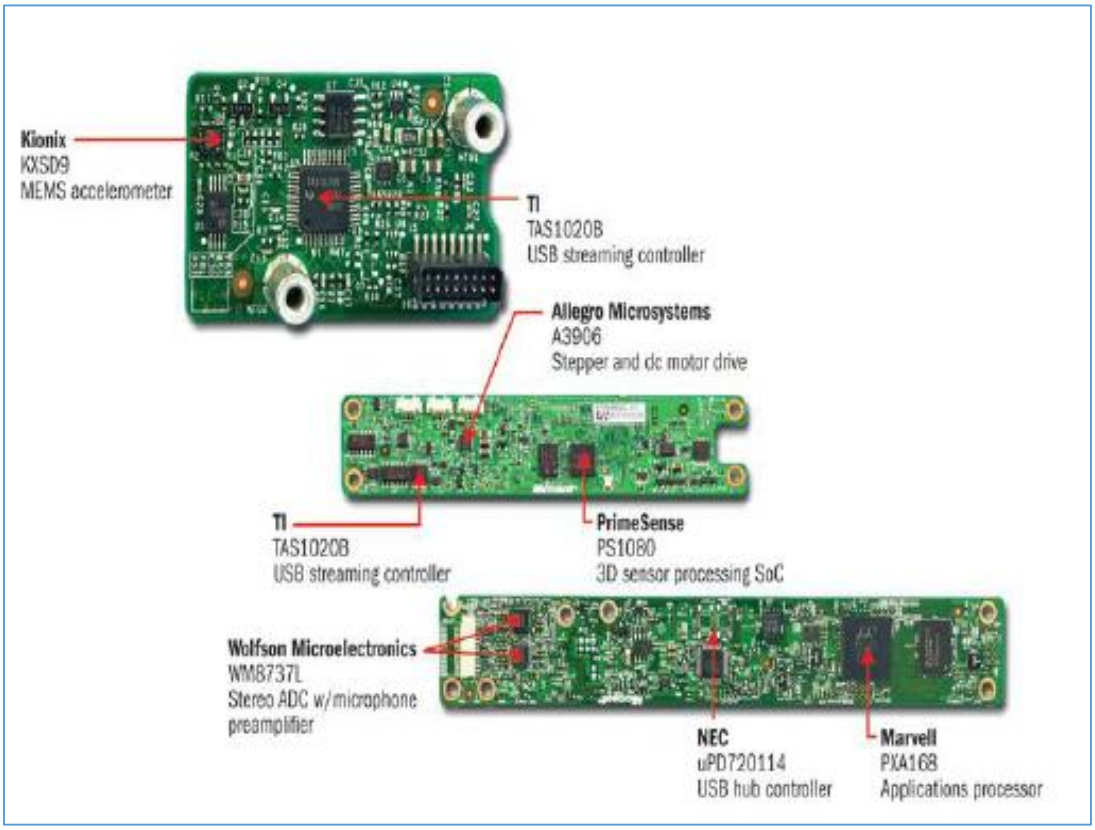


Figura 18. Circuitos principales de Kinect **[23]**.

4.3.3. Principales características

Las características de Kinect son [19] :

- Campo de visión horizontal: 57°
- Campo de visión vertical: 43°

- Rango de inclinación física: +/- 27°
- Rango de profundidad del sensor: 1.2 – 3.5 metros
- Resolución profundidad: 320x240 a 16 bits de profundidad a 30fps
- Resolución color: 640x480 32-bit de color a 30fps
- Audio: Audio de 16-bit a 16 kHz
- Resolución de objetos en profundidad: Distingue objetos separados a partir de 1 cm.

En cuanto a las conexiones, Kinect dispone de un conector especial, el mismo que cuenta con una conexión USB 2.0 más el cable de alimentación, tal y como se muestra en la figura 19, cabe resaltar que este conector es indispensable si se requiere utilizar Kinect para aplicaciones en la PC, ya que este dispositivo requiere de una alimentación superior a la que proporciona un USB 2.0, y al hacer uso de este cable se puede trabajar con Kinect sin problemas [24].

Figura 19. Cable extra para la conexión a PC **[24]**.

4.3.4. Funcionamiento de Kinect

Kinect es capaz de trabajar reconociendo lo siguiente: Reconocimiento de imágenes RGB, imágenes en 3D, esqueleto humano, y reconocimiento de audio.

- ✓ **Reconocimiento de imágenes RGB**

Kinect logra el reconocimiento de imágenes RGB en tiempo real y con esto permite añadir color a las imágenes capturadas con el sensor de 3D, gracias a que sus creadores han incorporado nuevos efectos y funciones a este dispositivo [23].

Cada imagen está formada por un conjunto de pixeles que a su vez se componen de cuatro valores a representar: rojo, verde, azul y el componente de transparencia, que para este caso será vacío, así cada pixel tiene un valor entre 0 y 254 que corresponden a un byte, entonces el sensor codifica las imágenes que adquiere en un vector de bytes, donde cada byte representa un componente de cada pixel. En la figura 20 se muestra como se distribuye la organización de los pixeles que va de arriba hacia abajo y de izquierda a derecha [23].

Figura 20. Disposición de los pixeles RGB en el array de bytes **[23]**.

- ✓ **Reconocimiento de imágenes en 3D**

Empleando una serie de filtros, y un sistema de patrones, Kinect es capaz de determinar e identificar si lo que tiene en su campo de visión es o no una persona, añadiendo a su reconocimiento la capacidad de identificar sus extremidades (superiores o inferiores), o cabeza, diferenciándolo de cualquier otro objeto que se encuentre dentro del campo de visión de Kinect (figura 20); como se mencionaba en apartados anteriores toda esta información la captura gracias al proyector de infrarrojos y al sensor de imágenes CMOS [23].

Figura 21. Reconocimiento de imágenes en 3D **[23]**.

- ✓ **Reconocimiento del esqueleto humano**

Una vez que se capturan las imágenes reconociendo al humano y por ende las partes del mismo, Kinect se encarga de convertir dicha información en puntos de articulaciones (Joints) que sirven de referencia para que el chip PrimeSense grafique el esqueleto humano, para ello Kinect almacena en su sistema más de 200 posiciones frecuentes del ser humano, y genera únicamente uno basándose en la experiencia. El reconocimiento del esqueleto se aprecia en la figura 22.

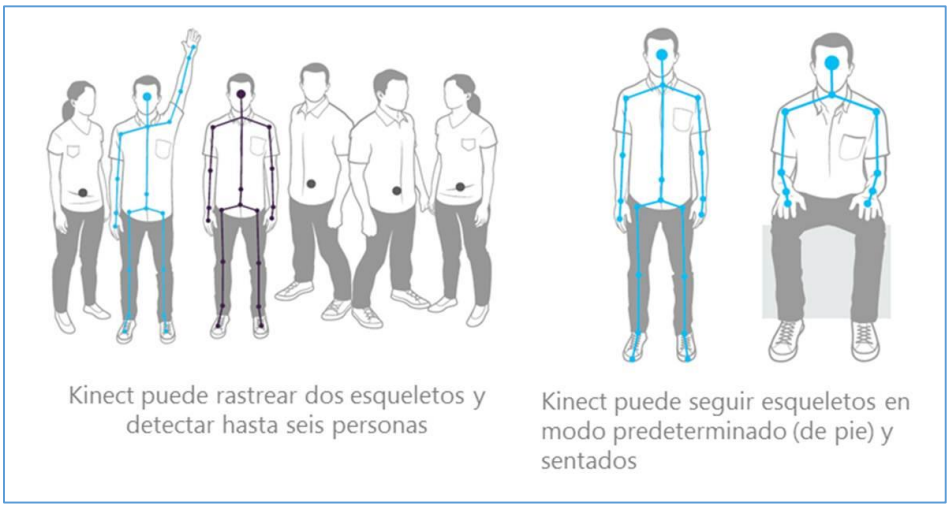


Figura 22. Reconocimiento del esqueleto humano con Kinect **[25]**.

En la figura 22 se aprecia en color azul, como Kinect identifica el esqueleto en las diferentes posiciones que el ser humano pueda estar. Son varios los puntos de articulaciones que Kinect logra identificar a lo largo de todo el cuerpo humano, (brazos, manos, muñecas, rodillas, cadera, entre otros), ver figura 23.

Todo esto se logra debido a la función Skeleton Tracking que permite detectar al ser humano incluso cuando esté en movimiento, dicha función tiene almacenado los datos de la imagen a color y los datos de la cámara de profundidad lo que admite distinguir entre objetos y personas asociando a éstas extremidades, articulaciones o incluso gestos [23].

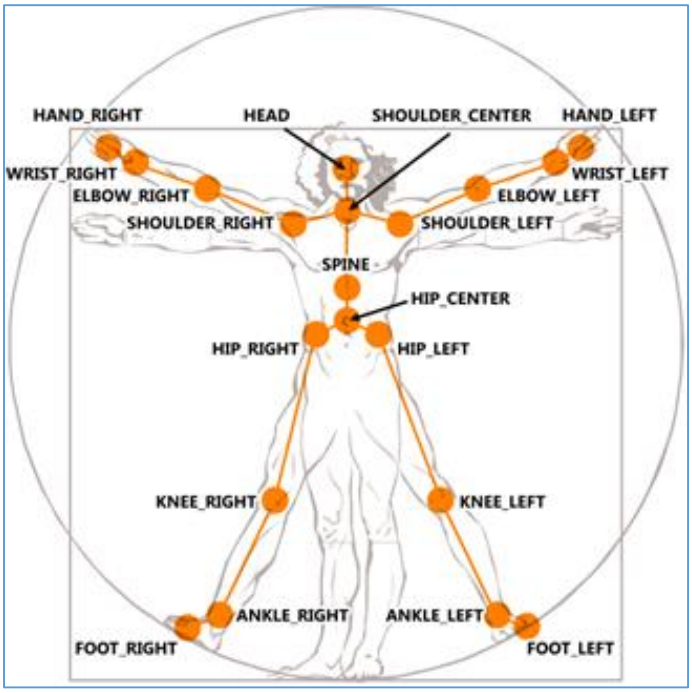


Figura 23. Puntos (Joints) que identifica Kinect **[26]**.

La función Skeleton Tracking consta de un SkeletonFrame que contiene los datos de cada esqueleto dentro de la función SkeletonData. A continuación en la figura 24 se muestra el esquema de la función Skeleton Tracking.

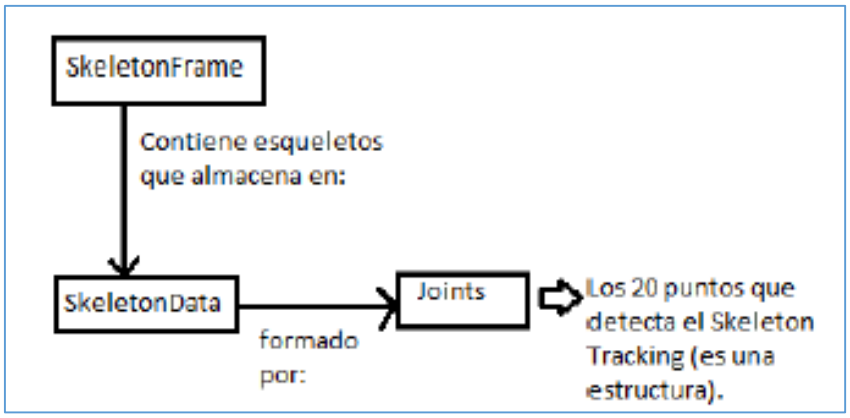


Figura 24. Esquema de la función Skeleton Tracking **[23]**.

Los pasos que se sigue para que Kinect logre detectar al humano es:

a) El sensor lanza una serie de puntos.
b) Kinect crea el mapa de profundidad a partir de los puntos detectados.
c) Encuentra el suelo y separa los objetos del fondo para localizar el contorno del humano.
d) Clasificación de las partes correspondientes al ser humano
e) Identifica las articulaciones.
f) Finalmente simula el esqueleto humano.

Cabe recalcar que la posición de cada articulación viene determinada en tres dimensiones (coordenadas X, Y e Z) cuya distancia se expresa en metros.

Como se puede ver en la figura 25, se trata de un sistema de coordenadas que coloca el conjunto de sensores en el punto de origen; el eje X indica la posición horizontal, el eje Y describe la posición vertical y el eje Z denota la distancia desde Kinect hasta la determinada articulación [23].

Figura 25. Eje de coordenadas para detectar el esqueleto **[23]**.

- ✓ **Reconocimiento de audio**

Los micrófonos en Kinect están distribuidos de tal manera que logren anular el ruido y se logre reconocer eficazmente la voz, uno está a la izquierda y tres restantes se ubican a la derecha del dispositivo, se utiliza un sistema de software especializado para determinar exactamente de dónde proviene el sonido y de esta manera crear una especie de burbuja alrededor del usuario [23].

En la figura 26 se muestra las características que se destacan para el reconocimiento de audio con Kinect.

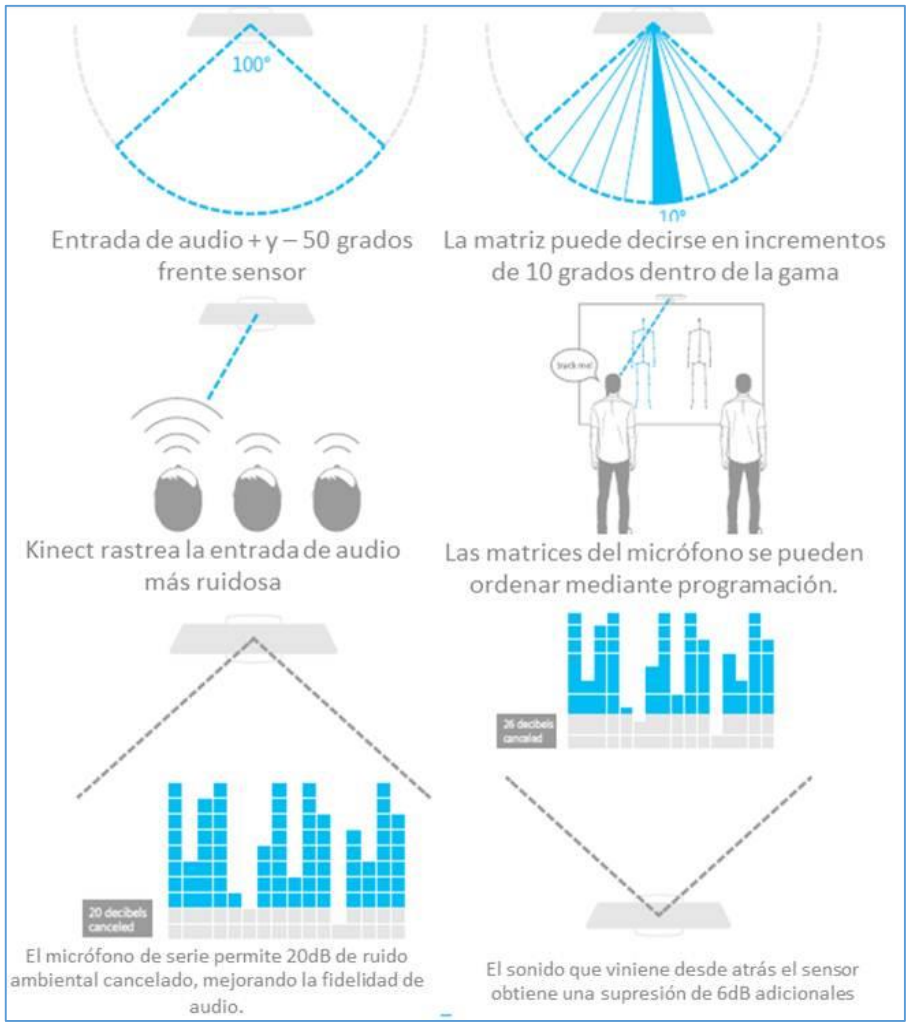


Figura 26. Características para el reconocimiento de audio con Kinect **[25]**.

4.3.5. Kinect Xbox 360 frente a otros dispositivos

A medida que se desarrollan nuevas tecnologías, todo evoluciona, y con ello las empresas también lanzan al mercado mejoras de productos o nuevos dispositivos, tal es el caso de Microsoft que hoy en día presenta la nueva versión de Kinect, denominado Kinect 2.0 que presenta mejoras en la cámara principal y Asus que también presenta lo que parece ser un claro competidor de Kinect, el Asus Xtion Pro Live, a continuación se describe detalladamente cada uno de estos dispositivos.

4.3.5.1. Kinect 2.0

Son muchas las ventajas que la nueva versión de Kinect presenta frente a su predecesor, una de las que resalta mayormente es la cámara principal que incorpora tecnología time-of-flight **(**TOF**)** permitiendo de esa manera la captura de imágenes con mayor detalle y

resolución además presenta una gran precisión. Al poseer tecnología TOF, la cámara de profundidad brinda la posibilidad de reproducir una escena con tres veces más fidelidad que el primer Kinect. En la figura 27 se muestra la nueva versión de Kinect [27].

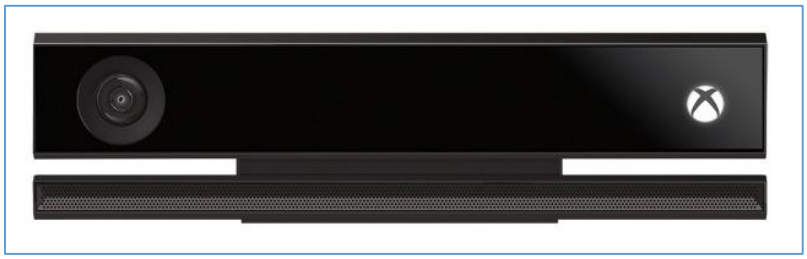


Figura 27. Dispositivo Kinect versión 2.0 **[27]**.

Otra de las ventajas al incorporar tecnología TOF es el aumento del campo de visión un 60% más grande lo que trae consigo el reconocimiento, seguimiento y distinción de todos los movimientos de 6 personas al mismo tiempo ya que se posee un espacio mayor de registro y una distancia menor entre el dispositivo y el usuario [27].

El sensor de infrarrojos también presenta grandes variaciones ya que logra reconocer objetos y personas en condiciones de muy poca luz, la precisión es tal que puede reconocer gente y registrar cuerpos incluso sin ninguna luz visible para el ojo humano. En ambientes con poca luz es capaz de distinguir con total precisión cada uno de los dedos de la mano dentro de una distancia aproximada de cuatro metros [27].

El procesador de la nueva versión de Kinect es capaz de leer y recoger información con una velocidad de datos de 2Gbps, gracias a todas estas mejoras Microsoft Research ha tenido que perfeccionar también el software, el mismo que tiene que ser capaz de interpretar todo lo que el dispositivo capta para poderlo emplear correctamente [27].

A continuación se resumen las principales características que se destaca de esta nueva versión de Kinect 2.0 de Microsoft frente a su predecesor Kinect Xbox 360.

Principales características [27]**:**

- Mayor campo de visión: 70º en horizontal (antes 57º) y 60 en vertical (antes 43º); esto permite detectar hasta 6 personas simultáneamente.

- Mayor resolución: 1920 x 1080 Full HD (antes 640 x 480), lo que permite detectar con mucha más precisión todo el entorno.
- Mejor rango de profundidad del sensor: El rango de actuación pasa a ser de 0,5 a 4,5 metros.
- USB 3.0: Al aumentar la velocidad de la comunicación con el ordenador los datos fluyen más rápido y esto disminuye la latencia del sensor. Pasa de 90ms a 60ms.
- Mejora de la captación de sonidos: Esta versión de Kinect viene dotada de una gran mejora en cuanto al reconocimiento de voz y la captación de sonidos, se ha mejorado la eliminación del ruido ambiental y esto permite captar con más detalle las instrucciones vocales.
- Captación de movimiento a oscuras: Ahora Kinect 2.0 es capaz de reconocer y captar los movimientos aunque en sala no exista iluminación.
- Motor de inclinación: No posee este motor por lo que el campo de visión es mayor.

4.3.5.2. Asus Xtion Pro Live

A mediados del año 2011, la empresa tecnológica Asus sacó al mercado lo que parece ser un claro competidor de Kinect denominado Xtion Pro Live, este dispositivo se origina gracias al acuerdo que se dio entre la compañía PrimeSense y la empresa Asus, según manifiestan los creadores, el Xtion Pro Live es compatible tanto con Windows como con Linux añadiendo a su hardware una cámara VGA y un arreglo de dos micrófonos [19]. En la figura 28 se muestra el nuevo Asus Xtion Pro Live.

Figura 28. Asus Xtion Pro Live **[19]**.

Principales características [28]**:**

- Consumo de energía: Inferior a 2.5 W
- Distancia de uso: Entre 0.8 m y 3.5 m
- Campo de visión: 58° H, 45° V, 70° D (Horizontal, Vertical, Diagonal)
- Sensor RGB y profundidad
- Profundidad del tamaño de la imagen:
    - VGA (640x480) : 30 fps
    - QVGA (320x240): 60 fps
- Resolución: SXGA (1280*1024)
- Plataforma: Intel X86 & AMD
- SO compatibles:
    - Win 32/64 : XP , Win7
    - Linux Ubuntu 10.10: X86,64/32 bit
    - Android (bajo petición)
- Interfaz: USB2.0
- Software: Kit de desarrollo de software (OPEN NI, SDK)
- Lenguaje de programación:
    - C++/C# (Windows)
    - C++(Linux)
    - Java
- Ambiente de operación: Interiores
- Dimensiones: 18 x 3.5 x 5 cm.

A continuación se muestra una tabla que resume y compara cada una de las principales características de los mencionados dispositivos.

Tabla 3. Tabla comparativa de las principales características de los mencionados dispositivos.

Características	Kinect Xbox 360	Kinect 2.0	Asus Xtion Pro Live
Campo de visión	57.5° horizontal, 43.5° vertical	70° horizontal, 60° vertical	58° horizontal, 45° vertical.
Interfaz	USB 2.0	USB 3.0	USB 2.0
Cámara de color	640x480 @30fps	1920x1080 @30fps	640x480 @30fps

Cámara de profundidad	320x240	512x424	320x240
Distancia de uso	0.8m – 4.0m	0.5m – 4.5m	0.8m -3.5m
Motor de inclinación	Si	No	Si
Arreglo de micrófonos	4 micrófonos - 48 Hz	4 micrófonos – 48KHz	2 micrófonos
Sistema Operativo	Win 7, Win 8	Win 8	Win 32/64 : XP , Win 7, Linux Ubuntu 10.10: X86,64/32 bit

En el presente trabajo se utilizó el dispositivo Kinect Xbox 360 debido al costo en comparación con su competencia, además los nuevos dispositivos (Kinect 2.0 y Asus Xtion Pro Live) actualmente se encuentran en el mercado únicamente pocos ejemplares ya que son relativamente nuevos y se los comercia a desarrolladores exclusivamente.

4.3.6. Controladores

4.3.6.1. OpenNI y NITE

OpenNI es un controlador que se liberó gracias al trabajo que se dio entre la empresa PrimeSense y OpenNI, estas empresas se unen con el fin de lanzar al mercado los drivers propios para Kinect, los mismos que son de código abierto y funciona perfectamente bien con el SDK propio de Microsoft, incluso con el dispositivo Xtion Pro Live de Asus [19]. El framework OpenNI (open Natural Interaction) es multilenguaje y multiplataforma, es decir, trabaja perfectamente en Windows, Linux, Ubuntu o Mac OS, el término interacción natural hace referencia al empleo de alguno de los sentidos que pueden ser la visión, el oído, es decir, la interacción humano-computador se puede dar, por ejemplo, mediante: reconocimiento de voz para la ejecución de comandos, reconocimiento de gestos con las manos, tracking de las partes del cuerpo, entre otros [29].

El objetivo de este controlador es facilitar la comunicación con los sensores de audio, video y profundidad de Kinect o Asus Xtion, al poseer una arquitectura en capas, tal y como se muestra en la figura 28, aquí la capa de aplicación (capa superior) representa el

software desarrollado para la interacción natural, la capa media (OpenNI Interfaces) facilita la comunicación entre la aplicación y los sensores, finalmente en la capa inferior (Hardware Device) se recopila la información del entorno (movimientos, sonido, etc.) a través de los sensores de Kinect [30].

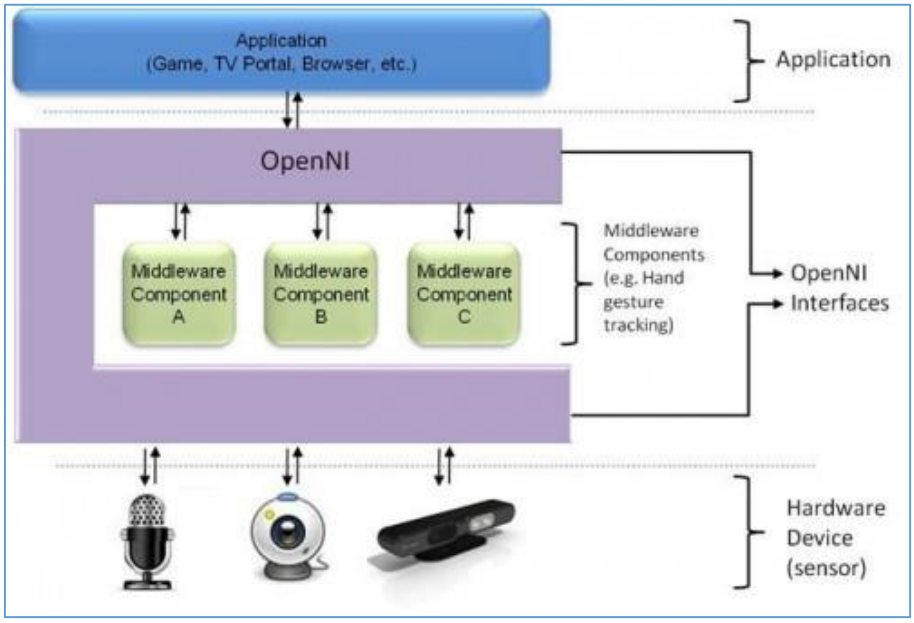


Figura 29. Arquitectura en capas de OpenNI **[29]**.

Una vez que se accede a los sensores, es necesario saber interpretar toda esa información, para ello, OpenNI cuenta con algoritmos de audio y visión que provee el middleware. Algunos de los componentes que posee se listan a continuación [30]:

- ✓ **Cuerpo (Full Body Analysis):** Recibe la información del sensor y crea información relacionada con el cuerpo, ya sea articulaciones, orientación, centro de masa etc.
- ✓ **Manos (Hand Point Analysis):** Componente del software que retorna la localización específica de las manos.
- ✓ **Gestos (Gesture Detection Middleware):** Identifica gestos predefinidos como un saludo a la cámara mediante las manos.
- ✓ **Escena (Scene Analizer middleware):** Componente que analiza la escena de la imagen y genera información como la separación entre el fondo y el frente, reconocimiento de figuras, las coordenadas del piso, etc.

4.3.6.2. Microsoft Kinect SDK

Al igual que OpenNI, Microsoft, como es lógico, también cuenta con un software para el manejo de Kinect, el Software de Desarrollo de Kinect (SDK) para Windows, este software incluye drivers, interfaces de programación de aplicaciones (APIs) para flujos de datos no procesados del sensor, interfaces naturales de usuario (NUI), archivos de instalación y materiales de referencia, el SDK ofrece a los desarrolladores la capacidad de diseñar aplicaciones con distintos lenguajes de programación entre ellos: C++, C# o Visual Basic a través de Microsoft Visual Studio [31].

Son muchas las características que más sobresalen del SDK, los desarrolladores tienen la capacidad de acceder a flujos de datos no procesados tanto del sensor de profundidad como del sensor de la cámara a color y de los cuatro micrófonos. Con el SDK se tiene la capacidad de detectar el esqueleto de una o dos personas en movimiento dentro del campo de visión de Kinect, otra particularidad de este software es la capacidad avanzada de audio**,** es decir, se cuenta con la supresión sofisticada de ruido y cancelación de eco, formación de haz para identificar la fuente actual de sonido e integración con el API de reconocimiento de voz de Windows [31].

El SDK se instala de forma rápida en Windows, no requiere de una configuración compleja y el tamaño total de instalación es de menos de 100 MB, además cuenta con una extensa documentación técnica donde se explica paso a paso la mayoría de muestras que se incluyen en el SDK [31].

El Kit de Desarrollo de Software proporciona herramientas que los desarrolladores pueden usar y aprovechar al máximo, a continuación se lista todo lo que el SDK incluye [23]:

- ✓ **Hardware de Kinect:** Constituido por los elementos que conforman el hardware: Kinect, cable USB y adaptador USB.
- ✓ **Controladores Kinect:** Los controladores lo conforman el conjunto de micrófonos, controles de audio y video, y aquellas funciones de enumeración que permiten el uso de más de un Kinect para una aplicación.
- ✓ **Skeleton tracking:** Implementa esta función para dos personas que estén dentro del rango de visión de Kinect.

- ✓ **Cámara de profundidad**: Se la usa para calcular la distancia existente entre el Kinect y el objeto.
- ✓ **Procesamiento de audio:** Útil para cuando se requiera usar los micrófonos de Kinect.
- ✓ **APIs:** Interfaz de Programación de aplicaciones, son un conjunto de rutinas, protocolos y herramientas que permiten el desarrollo de aplicaciones de software, el SDK implementa una serie de APIs para el uso de los desarrolladores.
- ✓ **Ejemplos de código fuente:** Al momento de instalar el SDK, los desarrolladores cuentan con ejemplos para el óptimo uso de las funcionales que ofrece Kinect.

En la figura 30 se muestra como sucede la interacción entre el hardware y el software haciendo usando el SDK.

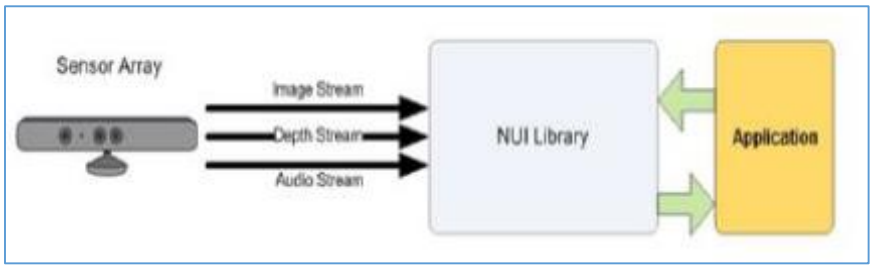


Figura 30. Interacción Hardware-Software con la aplicación **[23]**.

❖ Arquitectura del SDK de Kinect

A continuación en la figura 31 se aprecia cada uno de los componentes de la arquitectura de Kinect, los números corresponden a lo siguiente [23]:

1. Hardware de Kinect.
2. Corresponde a los drivers de Kinect, estos brindan soporte al sistema de micrófonos como un dispositivo Kernel-mode al que se puede acceder mediante las APIs, streaming de datos e imágenes de profundidad, y funciones de enumeración para permitir a una aplicación el manejo de más de un Kinect.
3. Por medio de las NUI API se recopilan los datos capturados por los sensores de imagen y además controlan el dispositivo, representa el núcleo principal de

Kinect para la API de Windows y da la facilidad de manejar los datos y brindar soporte a la función Skeleton Tracking.

4. Kinect Audio DMO amplía el soporte de los micrófonos para encontrar exactamente la formación de la fuente acústica.
5. APIs estándares de Windows.

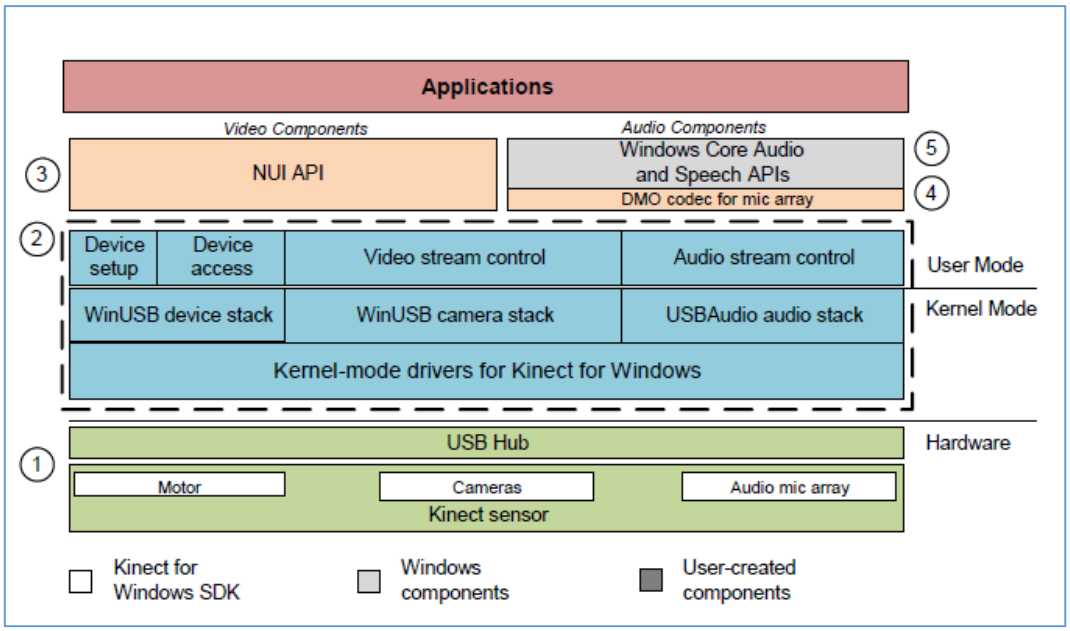


Figura 31. Arquitectura del SDK de Kinect **[32]**.

4.4. TÉCNICAS DE EXTRACCIÓN DE CARACTERÍSTICAS

Hoy en día los problemas que se relacionan con la detección y clasificación de objetos son de gran importancia dentro de la comunidad científica, debido a que se requiere la adquisición de imágenes para tal propósito, se cuenta con distintos tipos de cámaras, ya sean cámaras convencionales o cámaras dotadas de sensores infrarrojos, el fin es lograr procesar toda la información capturada por tales dispositivos en un tiempo prudencial, para ello existen diversos métodos que permiten la detección de objetos y el análisis de las imágenes, su posterior clasificación y finalmente determinar la existencia o no del objeto buscado, algunos de los métodos son: detector de bordes Canny, métodos basados en la computación de la orientación y magnitud de los gradientes de las imágenes, métodos que se basan en la diferencia de la intensidad de los histogramas o en histogramas de gradientes orientados (HOG), entre otros [33].

Se detalla a continuación brevemente el funcionamiento de cada uno de ellos [33] :

- Detector de bordes Canny: Es un método que se encarga de extraer los bordes de los objetos en las imágenes mediante la selección de aquellas regiones que poseas una alta derivada espacial, el hecho de tener en cuenta únicamente los bordes de los elementos en una imagen reduce ampliamente los datos a tratar, filtrando información no útil en la imagen, conservando características esenciales.
- Histograma de las diferencias en intensidad: Se basa en la elaboración de un vector de características que almacena información de las frecuencias relativas de las diferencias en intensidad calculadas entre pixeles vecinos a lo largo de cuatro orientaciones en una imagen en escala de grises.
- Histograma de Gradientes Orientados: Se basa en la división de la imagen en subbloques distribuidos a lo largo y ancho de dicha imagen con un cierto solape entre ellos, aquí cada bloque se subdivide en subbloques llamados también celdas y sobre estas se calcula el histograma de los gradientes orientados para finalmente almacenarlo en un vector de características de la imagen.

En este trabajo se utilizó el método HOG, dado su robustez frente a diferentes condiciones de iluminación, pequeñas variaciones en el contorno de la imagen, diferentes fondos y escalas, sin embargo, según bibliografía se conoce que los métodos mayormente usados son: El método detector de bordes Canny y el método HOG; es por ello que a continuación para mejor comprensión, se describen estos métodos más detalladamente.

4.4.1. Detector de bordes Canny

Los problemas relacionados con la detección de bordes dentro del procesamiento de imágenes, son de vital importancia para el reconocimiento de objetos, es por ello que el algoritmo de Canny se usa para la detección de bordes existentes en una imagen, según varios autores, este algoritmo está considerado como uno de los mejores métodos para la detección de contornos al hacer uso de máscaras de convolución y de la primera derivada [34].

El fundador de este algoritmo, propuso 3 criterios en los que se basa la ejecución del método de Canny, estos son:

- **Detección:** Evitar la eliminación de bordes importantes para no procesar falsos bordes.
- **Localización:** Establece que la distancia entre la posición real y la localización del borde debe ser la más mínima posible.
- **Respuesta:** Que integre las respuestas múltiples que pertenecen a un único borde.

Tal y como se había mencionado, el algoritmo de Canny hace uso de la primera derivada, esta derivada toma el valor de cero en todas aquellas regiones donde la intensidad no varía, y tiene un valor constante en toda la transición de intensidad, para la ejecución del algoritmo de Canny se deben seguir los pasos que se listan a continuación [34]:

- ✓ **Obtención del gradiente:** Se calcula la magnitud y orientación del vector gradiente en cada píxel, cabe recalcar que dentro del tratamiento de imágenes, el gradiente del pixel de una imagen es un vector que indica la dirección en la cual se origina un mayor cambio en el color o la intensidad de la imagen, por ejemplo, en imágenes en escala de grises el gradiente se dirige a pixeles de menor valor, es decir, de pixeles blancos a pixeles negros, mientras que el modulo indica la magnitud de este cambio.

- ✓ **Supresión no máxima:** Se logra el adelgazamiento del ancho de los bordes obtenidos con el gradiente con el fin de lograr bordes de un pixel de ancho.

- ✓ **Histéresis de umbral:** Finalmente en este paso, se aplica una función de histéresis basada en dos umbrales, con esto se intenta reducir la posibilidad de aparición de contornos que sean falsos.

Una vez que se sigue todos estos pasos, se tiene que cerrar los contornos en la imagen que pudiesen haber quedado abiertos por problemas de ruido. A continuación en la figura 32 se muestra el resultado que se obtiene en una imagen al aplicar el detector de bordes Canny.

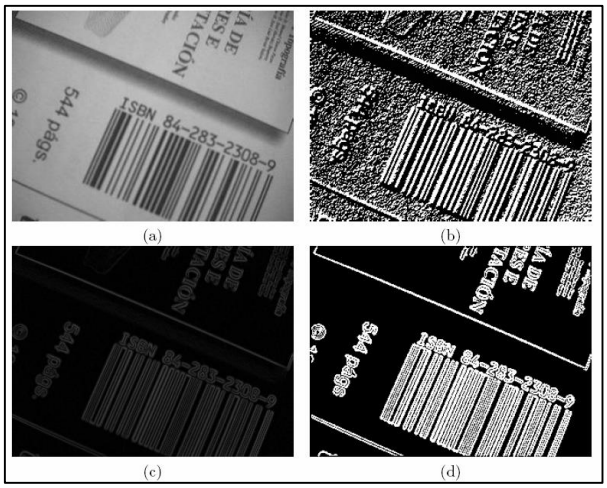


Figura 32. Resultado de aplicar el detector de bordes Canny: (a) imagen original; (b) orientación; (c) supresión no máxima; (d) histéresis de umbral **[34]**.

4.4.2. Histograma de gradientes orientados (HOG)

El histograma de Gradientes Orientados (del inglés *Histogram of Oriented Gradients* - HOG) es un método que consiste en dividir la imagen en subbloques que se distribuyen a lo largo y ancho de la misma imagen con un cierto grado de solapamiento entre ellos, a partir de ahí se procede a dividir en celdas y sobre estos últimos se calcula la magnitud y orientación de los gradientes en cada pixel, se calcula entonces el histograma de los gradientes orientados sobre cada uno de estos bloques, y finalmente se almacena en el vector de características de la imagen; por ejemplo: Dado el rango de valores de pixel [0, 255], se realiza una división en ocho clases del mismo tamaño: [0 ,32), [32, 64), [64, 96), … [224, 255]) y almacena en cada clase la frecuencia de pixeles con un valor comprendido entre ese subrango, es decir, el número de pixeles en la imagen cuyo valor esta entre los valores de inicio y de fin de cada subrango; según varios autores, este método presenta mayor confiabilidad y robustez dado las condiciones de iluminación, cambios en el contorno de la imagen o distintos fondos y escalas [33] [35].

Existen tres etapas que se siguen para la realización de este método, se enumeran a continuación [36]:

- Calcular los vectores de los gradientes.
- Calcular los histogramas sobre las orientaciones de los gradientes.
- Normalizar los gradientes.

En lugar de reducir toda la imagen bruscamente, HOG lo hace de forma iterativa y para obtener más información de la imagen, se hace uso de un Descriptor HOG, que como se había mencionado anteriormente, a una determinada imagen se la divide en cierto número de subimágenes del mismo tamaño para en lo posterior agruparlas en bloques con un mismo número de celdas, el solapamiento que debe existir entre dichos bloques es tal que el avance de bloques horizontalmente se realiza eliminando la columna de celdas de la izquierda y añadiendo la columna de la derecha y a su vez, verticalmente, eliminando la fila de celdas de arriba y añadiendo la fila de celdas de abajo, de esta forma los gradientes no se calculan uniformemente sobre un mallado denso lo que implicaría un coste en tiempo de computación demasiado grande ya que se tendría que calcular un HOG por cada celda, sino que gracias a la división en subbloques y celdas el coste computacional se agilizaría considerablemente [33] [35].

El procedimiento para el cálculo de los descriptores HOG de una imagen determinada se muestra en la figura 33.

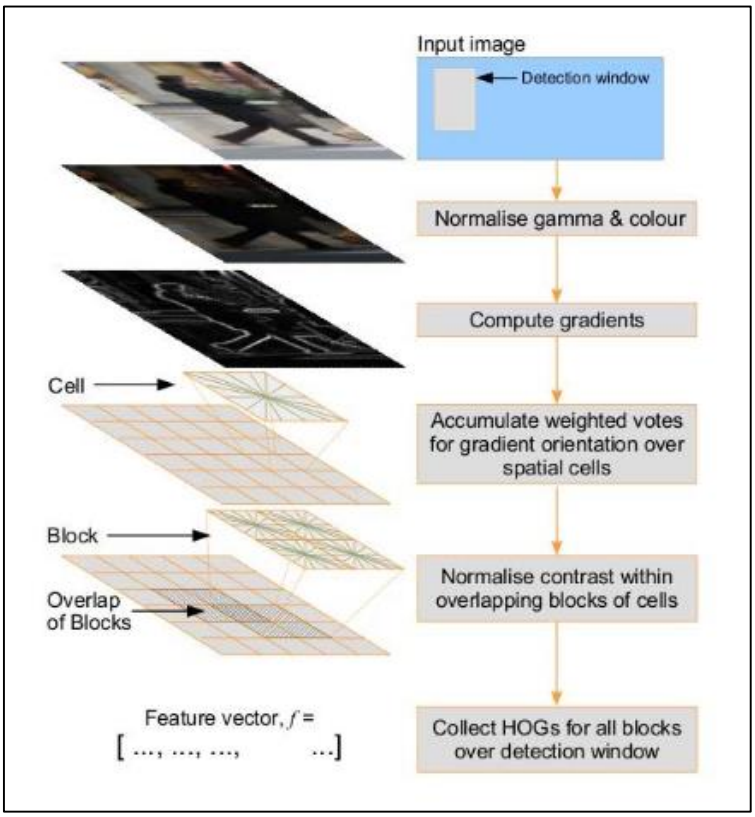
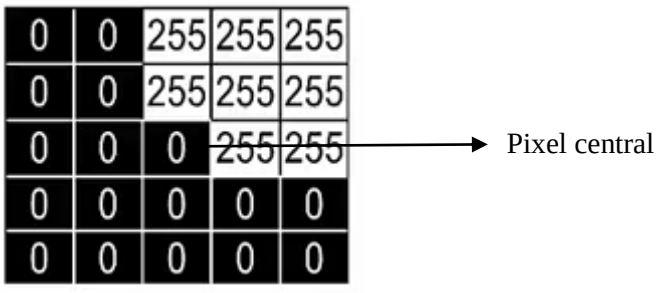


Figura 33. Proceso de extracción de características **[33]**.

Primero se detecta y captura una imagen a color para su transformación a escala de grises, seguidamente se calculan los gradientes sobre toda la imagen y se la divide en bloques solapados cierta área, posteriormente tal y como se observa, cada bloque se divide en subbloques (celdas) y para cada celda calculamos los histogramas de los gradientes orientados, finalmente aplicando una ventana gaussiana se almacena dicha información en el vector de características de la imagen, repitiendo el proceso para todos los bloques de la imagen [33].

Como se mencionó el gradiente es el cambio direccional en la intensidad de la imagen que se define claramente por dos valores, la dirección donde el cambio de la imagen es máxima y la magnitud del cambio en la dirección de máxima variación. A continuación se describirá detalladamente los pasos necesarios para el cálculo del descriptor HOG.

- Cálculo del gradiente: Basado en la diferencia de intensidad de los pixeles adyacentes en la dirección horizontal y vertical. Por ejemplo se tiene una imagen en donde los pixeles están distribuidos se la siguiente forma:



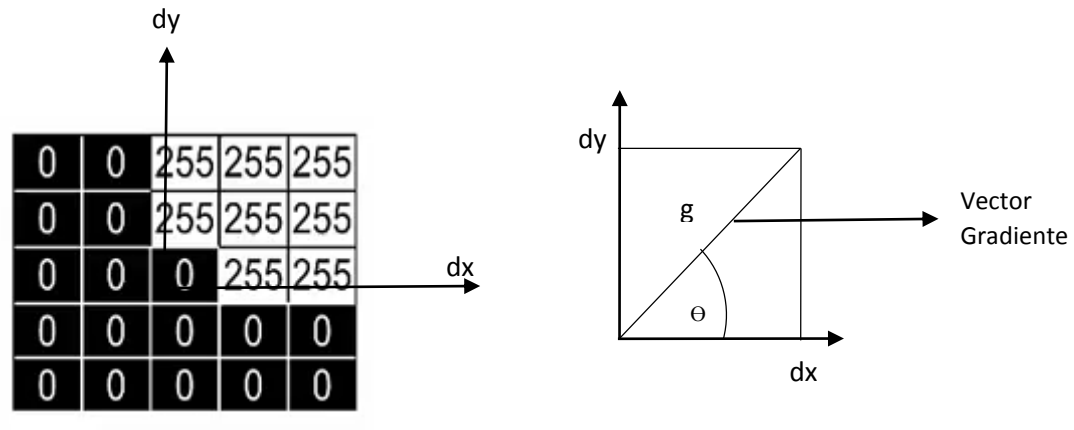
Basado en la siguiente fórmula se procede a calcular cada uno de los cambios de intensidad en ambas dirección tal y como se explica a continuación.

Por ejemplo para el pixel central se tiene:

$$\text{dirección horizontal:} \quad dx = I(x+1, y) - I(x-1, y) = 255$$

$$\text{dirección vertical:} \quad dy = I(x, y+1) - I(x, y-1) = 255$$

Con estas fórmulas se calcula dichos cambios para uno de los pixeles de la imagen en cuestión. A partir de las diferencias en horizontal y vertical se procede a calcular la orientación y la magnitud global del gradiente.



A partir del vector gradiente, se tiene:

$$Orientacion\ del\ gradiente: \quad \theta(x,y) = arctan\frac{dy}{dx}$$

$$Magnitud\ del\ gradiente: \quad g(x,y) = \sqrt{dx^2 + dy^2}$$

Una vez que se haya encontrado los cambios de intensidad y la magnitud y orientación de cada pixel de la imagen, se procede a calcular el descriptor HOG que es el descriptor global de toda la imagen que captura la forma del objeto.

- Cálculo del descriptor HOG: Para ello en primera instancia se divide a la imagen en celdas de tamaño fijo, tal y como se observa en la figura 34.

Figura 34. División de la imagen en celdas de tamaño fijo.

A continuación se calcula el histograma de las orientaciones de los gradientes en cada una de las celdas (figura 35).

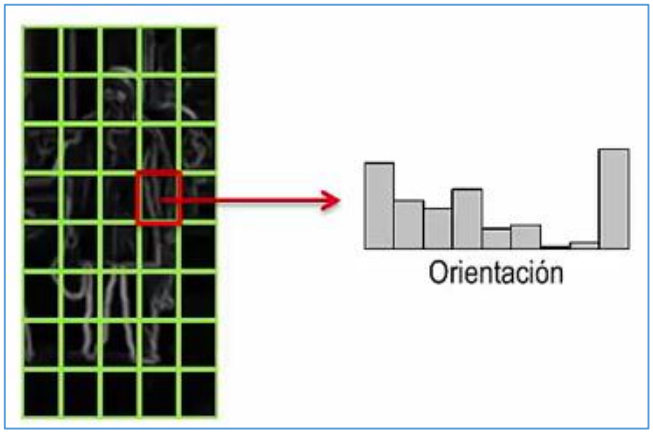


Figura 35. Histograma de orientaciones de gradientes en cada una de las celdas.

Obteniendo lo que se observa en la figura 36, para almacenar un arreglo con la respresentacion global de la imagen en forma de vector de características.

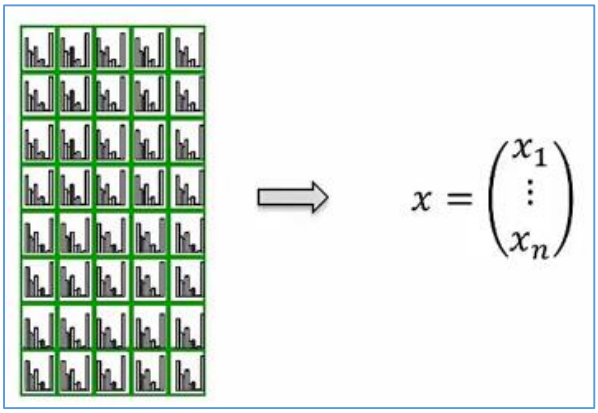


Figura 36. Histograma de cada una de las celdas de la imagen.

- <u>Cálculo del histograma de orientaciones en una celda.</u>

Para ello se procede primeramente a dividir cada una de las celdas en un tamaño fijo, por lo general este varía entre 6 y 8 pixeles tanto en ancho como en alto (figura 37), cabe recalcar que estos valores pueden variar.

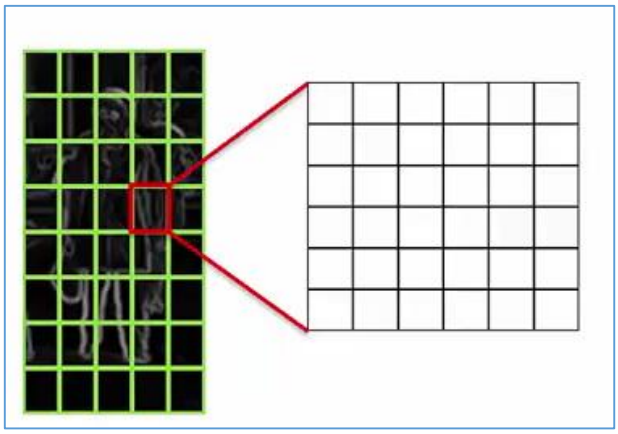


Figura 37. División en un tamaño fijo de cada una de las celdas.

A continuación se divide el rango de orientaciones en un número de intervalos fijos, la primera consideración acepta al rango de orientaciones en sí mismo ya que se puede considerar la orientación del gradiente con signo y con esto el rango de orientaciones estaría en el rango de 0-360° o bien se lo puede considerar sin signo con lo que estaría en el rango de 0-180°, tal y como se observa en la figura 38.

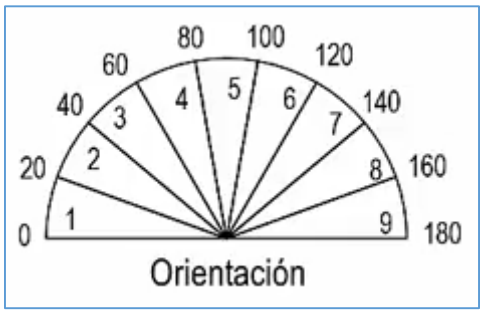


Figura 38. Diagrama de orientaciones e intervalos para los gradientes.

Con esta orientación (0-180°) dos gradientes con la misma dirección pero sentidos inversos se consideran equivalentes y quedan asignados al mismo intervalo.

Otro parámetro que hay q tener en cuenta es determinar en cuántos intervalos se divide el rango de orientaciones, que como se puede observar en la figura 37, corresponde a nueve intervalos que suele ser un valor común, con un rango de 20° por intervalo.

Realizado esto, se asigna cada pixel de la celda a un intervalo en función de la orientación del gradiente, calculados previamente en los pasos anteriores, posteriormente se acumula

la magnitud del gradiente de todos los pixeles asignados a un intervalo, obteniendo lo que se muestra en la figura 39, aquí se observa que a cada intervalo le corresponde valores de la magnitud de los gradientes.

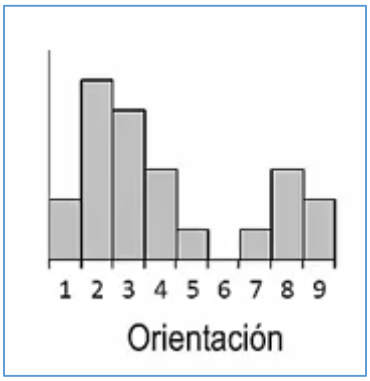


Figura 39. Acumulación de la magnitud del gradiente.

Cuando se tiene cambios de iluminación en las imágenes, la intensidad de los gradientes va a cambiar, por lo tanto las variaciones se reflejarán en los valores del histograma, es por ello que surge la necesidad de normalizar dichos valores con el objetivo de conseguir que la magnitud global del gradiente sea similar en todas las imágenes, entonces es necesario una normalización local adaptada a cada una de las zonas de la imagen, de esta forma se introduce el concepto de bloque.

Un bloque es simplemente la agrupación de las celdas en espacios de bxb celdas, una configuración habitual que se usa dentro del descriptor HOG es el empleo de bloques con tamaños de celdas de 2 celdas en horizontal y 2 celdas en vertical, como se muestra en la figura 40.

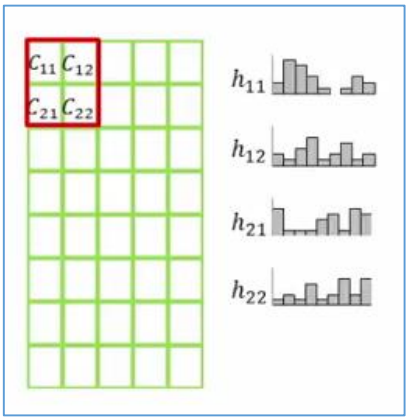


Figura 40. Histograma de cada una de las celdas de un bloque determinado.

Entonces para cada bloque le corresponde un histograma de cada una de las celdas, dichos histogramas se concatenan para obtener el vector con la representación del problema (Figura 41).

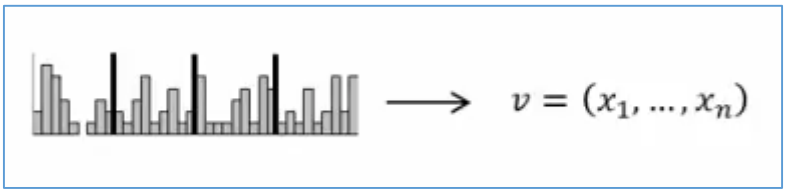


Figura 41. Histogramas concatenados para la obtención del vector de características.

En la práctica, los bloques se definen de tal manera que exista cierto solapamiento entre ellos, con el objetivo de encontrar un descriptor más robusto ante deformaciones y variaciones en la forma del objeto; habitualmente los bloques se colocan con una separación de una celda tanto en horizontal como en vertical y la representación final del descriptor HOG se obtiene concatenando la representación normalizada de todos estos bloques solapados, tal y como se observa en la figura 42.

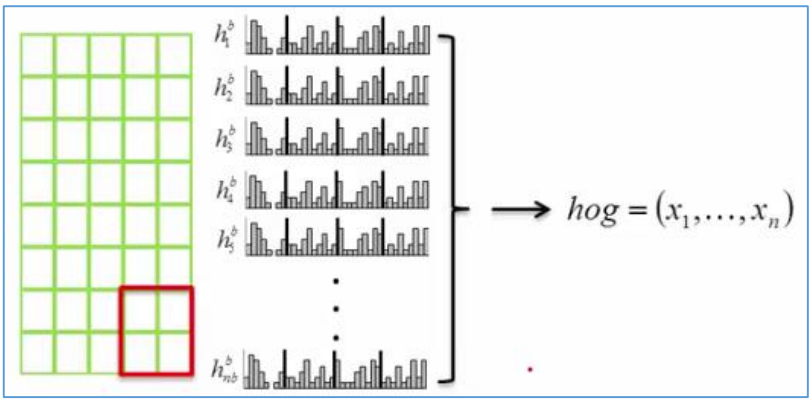


Figura 42. Representación final del descriptor HOG.

De esta forma en la figura 43 (a) se observa que cada celda contribuye a la descripción de varios bloques, tantos como celdas se tenga en cada uno de los bloques, sin embargo pese a que dicha celda tendrá el mismo histograma en todos los bloques, cada uno de estos aplicará una normalización diferente, lo que hace que la contribución del histograma de la celda a los distintos bloques sea distinto como se muestra en la figura 43 (b).

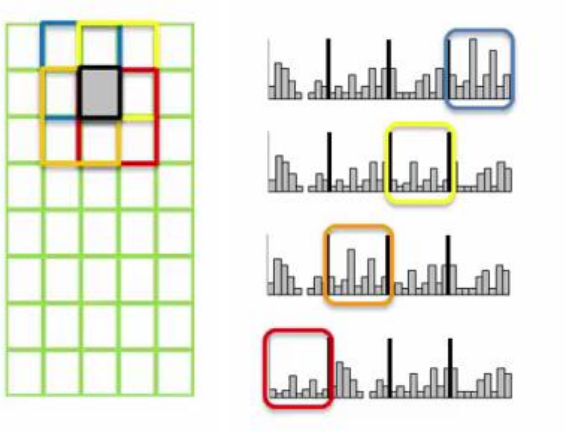


Figura 43. a) Contribución de una celda en cada uno de los bloques, b) Histogramas normalizados de la celda que aporta a cada bloque.

- Parámetros finales del descriptor HOG:
  - Tamaño de celda
  - Signo del gradiente
  - N° de intervalos del histograma de orientaciones
  - N° de celdas en cada bloque.

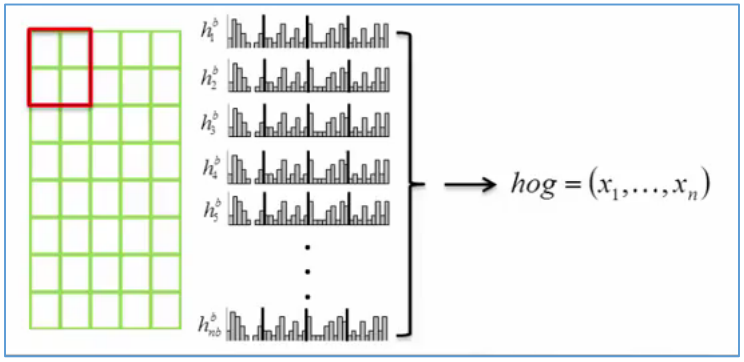


Figura 44. Parámetros finales para el descriptor HOG.

Todos estos parámetros ayudan a determinar la dimensión *n* del descriptor final, este número final de dimensiones se puede calcular con la expresión:

$$n = n°bloques\ x\ n°\frac{celdas}{bloque}\ x\ n°\ intervalo\_histograma$$

De donde el número de bloques se encuentra aplicando lo siguiente:

$$n° \; bloques = n° \; celdas - n° \frac{celdas}{bloque} + 1$$

Realizado esto ya se conoce exactamente la dimensión del descriptor, es decir, el número de componentes que contendrá el mismo.

4.5. APRENDIZAJE AUTOMÁTICO

El aprendizaje automático o también conocido como Machine Learning por sus siglas en inglés, es la rama de la inteligencia artificial, que mediante la ejecución de técnicas, permite dotar a las computadoras de la capacidad de "aprender", es decir, se trata de crear programas capaces de generalizar comportamientos y reconocer patrones a partir de información provista en forma de ejemplos, de esta manera, al aprendizaje automático se lo conoce como un proceso de inducción del conocimiento, aquí se obtiene un enunciado general a partir de casos particulares, este método de aprendizaje se lo usa comúnmente para el reconocimiento de rostros, clasificación de correos (spam), reconocimiento de voz, entre otros, en resumen, Machine Learning permite: extraer información valiosa que admite comprender una determinada situación o problema; procesar, clasificar, interpretar, analizar y evaluar información disponible; predecir escenarios y situaciones; generar aprendizaje para implementar mejoras y automatizar procesos [37] [38].

Como se había mencionado, el aprendizaje automático, posee técnicas que posibilitan la creación de los distintos algoritmos necesarios para el aprendizaje, por ello, Machine Learning se divide en dos áreas principales: aprendizaje supervisado y no supervisado, el primero se basa en hacer predicciones a futuro fundamentándose en comportamientos o características que se observan en datos que están previamente almacenados (base de datos), es decir permite buscar patrones en datos en la base de datos existente relacionando todo los campos con el campo objetivo, mientras que el segundo (aprendizaje no supervisado) hace uso de datos que no están etiquetados, el fin que se persigue con este método de aprendizaje, es explorar los datos para encontrar alguna estructura o manera de organizarlos [38].

Cabe recalcar que para el desarrollo de este trabajo de investigación se hace uso del primer método de aprendizaje, el aprendizaje supervisado, es por ello, que a continuación

se hará énfasis en una de las técnicas mayormente usada dentro de este método, las Máquinas de Soporte Vectorial.

En la figura 45 se muestra una clasificación de los tipos de aprendizaje inductivo, que como se había mencionado, este tipo de aprendizaje se basa en que el sistema pueda adquirir los conocimientos suficientes para realizar la tarea para la cual fue diseñado a partir de ejemplos reales y concretos, cabe resaltar que en la práctica no existen sistemas inductivos puros, ya que los aspectos generales se suman a la solución previos a ser experimentados deductivamente, para de esa forma generalizar el desarrollo de manera inductiva [39].

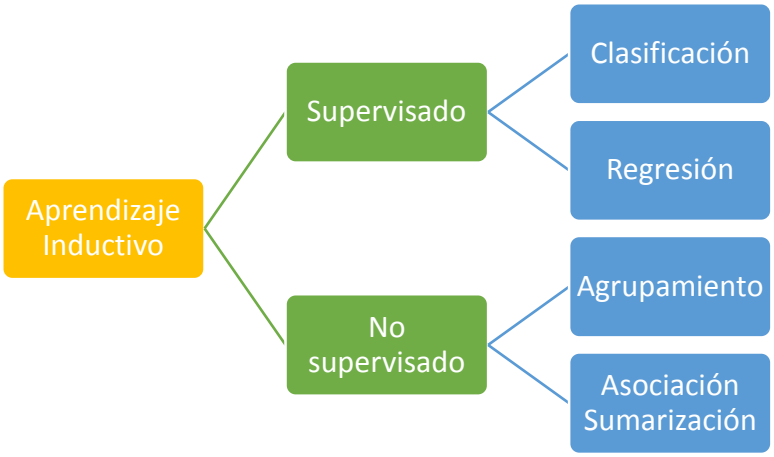


Figura 45. Clasificación de métodos de aprendizaje inductivo.

Los algoritmos que siguen modelos predictivos son los llamados de aprendizaje supervisado, para cada muestra evaluada por la función se obtiene la salida, conocida como etiqueta, que sirve para evaluar la capacidad de acierto de la predicción realizada, si la etiqueta es discreta, el modelo será de clasificación, mientras que si es continua se realiza una tarea de regresión [39].

Las máquinas de soporte vectorial son algoritmos predictivos y existen ambos tipos, tanto SVM de clasificación como SVM de regresión. La toolbox LibSVM dispuesta para Matlab permite trabajar con los dos algoritmos, pero en este caso, dado que se empleará etiquetas discretas sólo se utilizará el modelo de clasificación.

Los algoritmos que siguen modelos descriptivos son conocidos por aprendizaje no supervisado, ya que no tienen un atributo salida. Los datos de entrada suelen ser variables aleatorias sobre las que se construye un modelo de densidad. Se dividen generalmente en: agrupamiento en donde los datos se agrupan según su similaridad; sumarización el mismo que busca hallar una descripción para el conjunto de datos y asociación cuyo objetivo es encontrar patrones de asociación entre los atributos de los datos de entrada [39].

4.5.1. Aprendizaje Supervisado

Tal y como se había descrito en párrafos anteriores, el aprendizaje supervisado hace uso de un algoritmo que origina una función estableciendo una correspondencia entre las entradas y salidas deseadas del sistema, el uso de este algoritmo se da principalmente en problemas relacionados con la clasificación, aquí el sistema de aprendizaje intenta etiquetar (clasificar) una serie de vectores por medio de varias categorías que se conocen comúnmente como *clases*, la médula del conocimiento del sistema en cuestión se encuentra constituida por ejemplos de etiquetados anteriores [40].

Los pasos que se suelen seguir para la resolución de un problema mediante el aprendizaje supervisado, se describen a continuación [39]:

- ✓ Determinar los objetos de entrenamiento: Fijar el tipo de dato a emplear.
- ✓ Adquirir un conjunto de datos de entrenamiento, ya sea de forma manual, mediante un sensor, ejemplos reales, entre otros.
- ✓ Establecer el formato de los datos de entrada, es decir, partiendo de los ejemplos de entrenamiento del paso anterior, encontrar las características a estudiar para el desarrollo de una función que acople dichas características al entorno de programación.
- ✓ Decidir la función que resolverá el problemas: elegir si se emplean arboles de decisión, redes neuronales, máquinas de soporte vectorial, etc.

Entre los ejemplos de sistemas de aprendizaje supervisado que se destacan, se encuentran [39]:

- Redes Neuronales
- Máquinas de soporte vectorial

- Clasificador bayesiano ingenuo
- Modelo de los k-vecinos más próximos
- Arboles de decisión
- Funciones de base radial.

A continuación se detallará todos los aspectos importantes de las máquinas de soporte vectorial, ya que como se había mencionado, se usa esta técnica para la clasificación.

4.5.1.1. Máquinas de Soporte Vectorial

Las máquinas de soporte vectorial (SVM por su nombre en inglés *Support Vector Machine*) son una técnica o método de aprendizaje supervisado, que ha demostrado en los últimos años, tener un gran desempeño, superando a las máquinas de aprendizaje tradicional como las redes neuronales, una SVM aprende la superficie de dos clases distintas, en donde a partir de muestras de entrada se reconoce patrones que permiten resolver problemas de clasificación, en este caso se hará uso de este algoritmo para el reconocimiento de manos, con la premisa de que se identifican únicamente dos clases, una clase asociada a las manos y otra a todo aquello que no lo sea [35] [41].

En la figura 45 se puede interpretar el problema de clasificación de dos clases, es decir, partiendo de un conjunto de muestras iniciales tanto positivas como negativas con *n* características cada uno, se coloca cada una de las muestras como un punto en el espacio de *n* dimensiones, entonces el objetivo es hallar una línea, conocida como *hiperplano,* que logre separar los puntos de ambas clases en dos grupos, tal y como se observa en la siguiente figura [35].

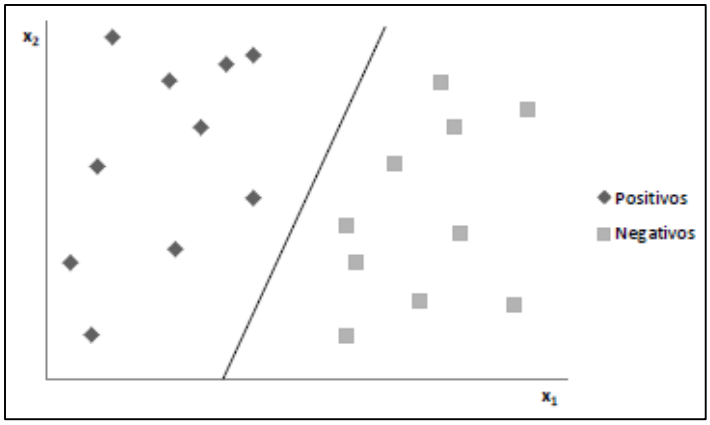


Figura 46. Hiperplano de separación de dos clases **[35]**.

Los datos que se mapean con SVM se lo realiza por medio de un *Kernel*, a un espacio de características dimensionalmente mayor, por ejemplo: si los puntos de entrada están en $\Re^2$ entonces son mapeados por la SVM a $\Re^3$, ahí encuentra un hiperplano que los separe y busca la máxima separación entre las clases, dicha distancia está marcada como margen en la figura 47; cabe recalcar que un Kernel es una función matemática que se emplea en las Máquinas de Soporte Vectorial que permite convertir problemas de clasificación no-lineal en el espacio dimensional original, a un problema de clasificación lineal en un espacio dimensional mayor, los kernels más comunes son: la función lineal, polinomial, RBF (Radial Basis Function), ERBF (Exponential Radial Basis Function), entre otros [39] [40].

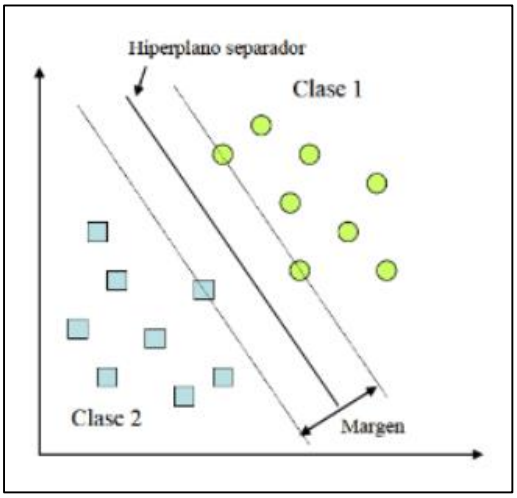


Figura 47. Separación de datos mediante SVM **[39]**.

La forma más fácil para resolver un problema mediante SVM es que los datos que se tienen como entrada sean linealmente separables, pero en algunas ocasiones esto no siempre sucede y puede que las entradas sean linealmente no separables o que a su vez exista cierto nivel de ruido. Dadas estas situaciones se distinguen distintos 2 tipos de SVM que se pueden y se enumeran a continuación, estas SVM corresponden problemas de clasificación binaria.

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

- **Máquinas de Soporte Vectorial para clasificación binaria**

❖ **SVM lineal con margen máximo.**

Las máquinas de soporte vectorial conforman hiperplanos que separan los datos de entrada en dos subgrupos que tienen etiqueta propia en el caso de ser linealmente separable, es decir, 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, tal y como se observa en la figura 48 [42].

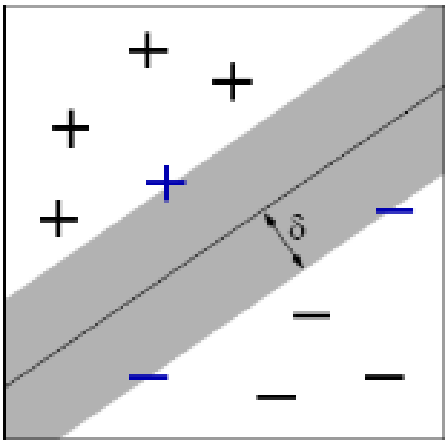


Figura 48. SVM linealmente separable **[42]**.

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

$$h(x) = \omega^T x + b = 0$$

Donde ω y $x \in \Re^d$, siendo d la dimensión del espacio de entrada.

La resolución para dicho caso sería suponer que se tiene un conjunto de n datos separables linealmente:

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \, donde \, x_1 \in \Re^d \, e \, y_1 \in \{-1,1\}$$

Se cumplirá, según el lado en el que esté respecto del hiperplano:

$$\omega^T x_i + b \ > 0, \qquad para\ y_i = 1, i = 1, \dots, n$$

$$\omega^T x_i + b \ < \ 0, \qquad para\ y_i = -1, i = 1, \dots, n$$

Las ecuaciones definen las dos clases presentes en este problema que al no estar mezcladas permiten hallar con sencillez a nivel matemático el hiperplano que las separa con margen máximo. Las expresiones antes halladas se las puede reducir a una sola:

$$y_i\ (\omega^T x_i + b)\ > 0, para\ i = 1, \dots, n$$

Para resolver el problema, se considera que los vectores soporte (con relleno en color negro en la figura 49) es decir, los puntos más cercanos al hiperplano, cumplen:

$$h(x_i) = 1, para\ y_i = 1$$

$$h(x_i) = -1, para\ y_i = -1$$

En donde al ser una aproximación sencilla, los vectores soporte son fácilmente identificables a nivel gráfico. Se resalta que no puede haber datos del conjunto de aprendizaje dentro del margen, por lo que la ecuación $y_i\ (\omega^T x_i + b)\ > 0$, queda así:

$$y_i\ (\omega^T x_i + b)\ \geq 1, i = 1, \dots n$$

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

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

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

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

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:

Maximizar:

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

Sujeto a:

$$y_i\,(\omega^T x_i + b)\ \geq 1, i = 1, \ldots n$$

Que es la condición que indica que ningún vector de entrenamiento debe quedar dentro del margen que separa a las dos clases. En la figura 49 se muestra todas estas situaciones con sus elementos característicos, es decir, las dos clases, el hiperplano y el margen máximo.

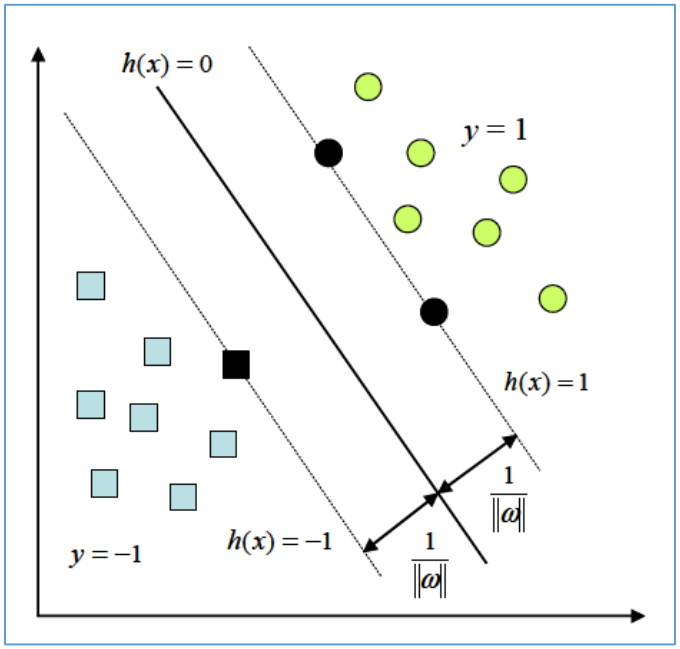


Figura 49. SVM con margen máximo (en negro los vectores de soporte) **[39]**.

Entonces se puede decir que la ecuación del hiperplano sólo depende de los vectores soporte, de ahí surge el nombre del algoritmo, máquinas de soporte vectorial.

❖ SVM para la clasificación no lineal

Existen situaciones donde es posible que los datos de entrada no sean linealmente separables, como se puede apreciar en la figura 50.

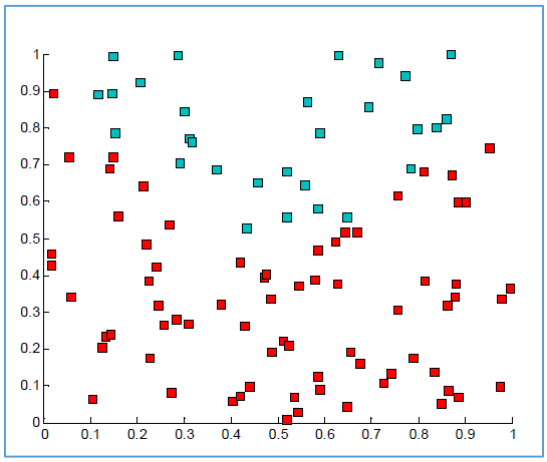


Figura 50. Conjunto de datos linealmente no separables **[39]**.

A continuación se describen dos situaciones en la que se puede dar solución a este tipo de problemas:

a) El primero requiere del uso de una función Kernel para la transformación de las variables de entrada ya que los datos pueden ser separables con margen máximo pero en un espacio de dimensionalidad mayor, ver figura 51.

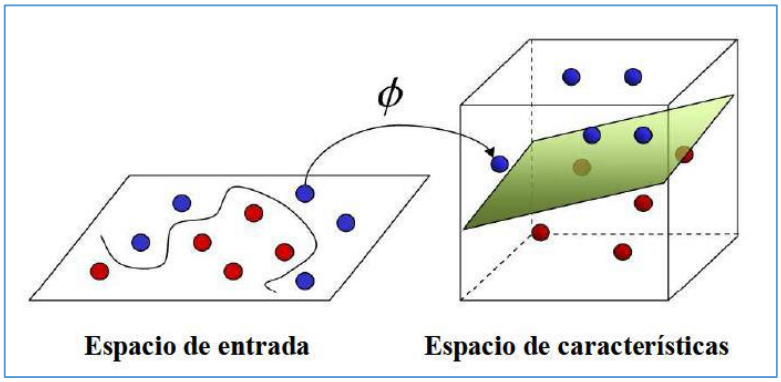


Figura 51. Transformación de datos de entrada a un espacio de dimensión mayor **[39]**.

En la figura 52 se puede observar gráficamente como la función Kernel realizaría la separación y el traslado de los datos al espacio de características [42].

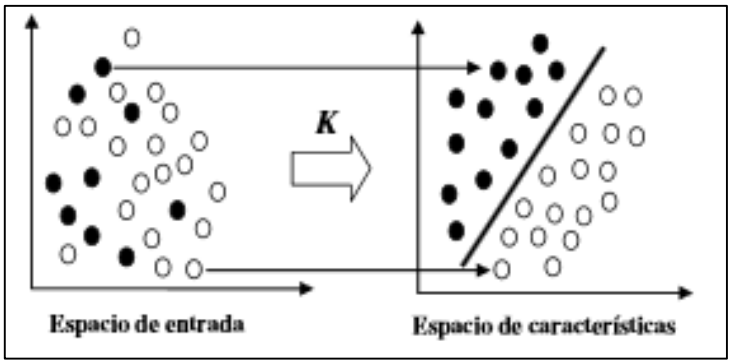


Figura 52. SVM no lineal inducida por una función Kernel **[42]**.

b) La segunda situación de las SVM se denomina como Margen Blando *o "Soft Margin"* por sus siglas en inglés, aquí no es viable encontrar una transformación de los datos, ya sea, en el espacio de entrada o en el espacio de características, es decir, no se puede separar linealmente los datos; trata aquellos casos donde los datos de entrada son erróneos, presentan ruido o solape entre las distintas clases de los datos de entrenamiento que comprometa el hiperplano clasificador [42].

- **Máquinas de Soporte Vectorial para clasificación multiclase**

Las máquinas de soporte vectorial (SVM) se usan comúnmente para problemas de tipo binario, pero puede darse el caso de que el problema a tratar implique datos de entrada de más de dos clases, como es el caso de este trabajo de investigación, aquí se tiene una clasificación multiclase dado que cada letra a reconocer representan una categoría distinta y para realizar dicha clasificación es necesario resolver tantos problemas binarios como clases haya que identificar. En cada uno de ellos se considera una clase positiva y el resto negativas.

Existen dos métodos para resolver problemas de clasificación multiclase, entre ellos [43]:

- **Clasificación 1-v-r *(del inglés one-versus-rest):*** En cada uno de los problemas se considera una clase positiva y las demás negativas, por lo que se tiene que hallar tantos hiperplanos como clases existan.

- **Clasificaion 1-v-1** ***(del inglés one-versus-one):*** En este método en cambio, para cada problema se toman dos clases de las K totales y se compara cada clase con cada una de las restantes, lo que supone realizar K(K-1)/2 clasificaciones; de estos dos métodos, el primer enfoque es el más habitual.

4.6. MATLAB

Matlab (del inglés Matrix Laboratory) es un software dotado de un lenguaje de alto desempeño ya que integra el cálculo, la visualización y la programación en un ambiente intuitivo y fácil de usar, se caracteriza porque las soluciones a los distintos problemas computacionales se los resuelven a través de arreglos de matrices, es ampliamente usado por Ingenieros de varias ramas por su versatilidad y capacidad de resolver problemas complicados numéricos [14] [44].

Al ser un software versátil, se lo usa ampliamente en [14]:

- Desarrollo de algoritmos
- Cálculos numéricos
- Análisis de datos, exploración y visualización
- Gracias a Simulink, permite el modelado, simulación y prueba de prototipos.
- Graficación de datos con fines científicos o de ingeniería.
- Desarrollo aplicaciones que necesitan de la creación de una interfaz gráfica de usuario (GUI).

Gracias al uso de los llamados *toolboxes*, Matlab permite aplicar la teoría aprendida al extender el ambiente para la resolución de problemas específicos, ya que posee toolboxes para áreas como: Procesamiento Digital de señales, Sistemas de Control, Redes Neuronales, Lógica Difusa, Procesamiento de imágenes, entre otras [14].

4.6.1. Interfaz Gráfica de Usuario.

La interfaz gráfica de usuario (GUI por el acrónimo en inglés Graphical User Interfase), es básicamente un entorno visual que permite la interacción entre el computador y el usuario; entre los programas que facilitan la creación de una GUI se destaca Matlab, que pese a no ser opensource, es muy potente y resalta entre los programadores.

Para realizar una interfaz gráfica de usuario desde Matlab, basta con escribir "*guide*" en la ventana de comandos y se desplegara una ventana como se muestra en la figura 53 en donde al seleccionar *Blank GUI (Default)* se nos abrirá un nuevo formulario en el cual ya podremos empezar a diseñar un determinado programa.

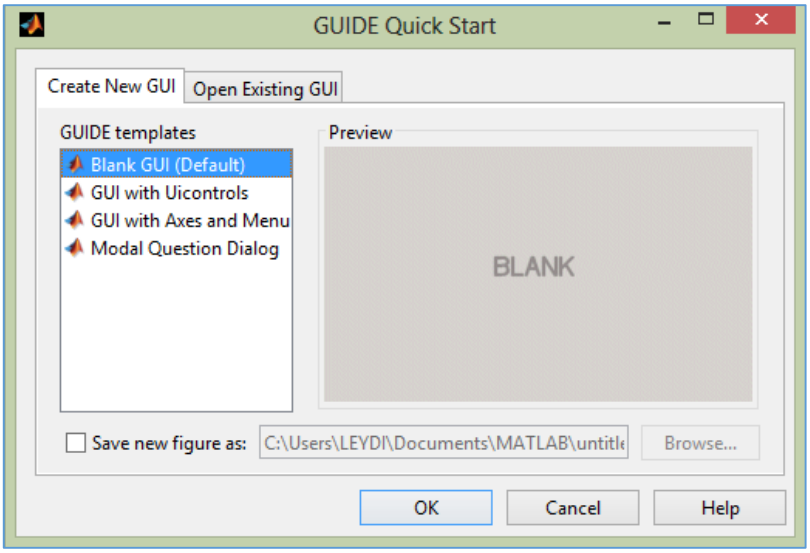


Figura 53. Ventana de inicio de GUI.

4.6.2. LIBSVM

Como se había mencionado Matlab nos ofrece la posibilidad de trabajar con toolboxes y librerías que simplifican las tareas del programador, dicho esto, a continuación se hace hincapié en la librería que se usó para cumplir con cada uno de los objetivos propuestos en el presente trabajo de investigación.

La librería LibSVM es un conjunto de funciones en diferentes lenguajes de programación que logran desarrollar distintos algoritmos de máquinas de soporte vectorial; fue realizada por Chih-Chung Chang and Chih-Jen Lin y resuelve problemas SVM de clasificación (C-SVM, nu-SVM), de regresión (épsilon-SVR, nu-SVR) y de estimación-distribución (SVM de una clase) tanto de clases binarias como de multiclase [39].

En este proyecto únicamente se emplearon los códigos dispuestos para Matlab y de entre las funciones disponibles se hizo uso de svmtrain y svmpredict.

4.6.2.1. SVMTRAIN

Es la función encargada de entrenar la SVM, se le introducen una serie de datos de entrada y genera un modelo en forma de variable tipo estructura que caracteriza nuestra SVM, la librería LibSVM permite clasificar datos con etiquetas binarias (-1, 1), o problemas con mayor número de clases; en caso de emplear más de dos clases en el modelo de la SVM las etiquetas tendrán que ser positivas empezando por la número 1 (2, 3, 4…) [39].

La llamada a la función se realiza con la siguiente sintaxis:

```
modelo = svmtrain(training_label_vector, training_instance_matrix,
'libsvm_options')
```

De donde [39]:

- Modelo: Es el resultado del entrenamiento. Una variable tipo estructura que define matemáticamente el modelo sobre el que se clasifican los futuros datos de entrada. Un ejemplo del modelo sería:

```
Parameters: [5x1 double]
nr_class: 26
totalSV: 2112
rho: 325x1 double
Label: 26x1 double
ProbA: []
ProbB: []
nSV: 26x1 double
sv_coef: 2112x25 double
SVs: 2112x20736 double
```

  Parameters: Es el vector que guarda el tipo de parámetros que definimos a la hora de ejecutar la opción. Serían los correspondientes al tipo de función SVM usada, el Kernel o las variables.

  Los datos que se muestran en el ejemplo, son los que se obtienen mientras se ejecutó esa función en Matlab, como se observa existe un total de 26 clases (nr_class: 26), siendo 26 etiquetas que corresponden a cada una de las letras del abecedario (Label: 26x1 double); en total se generaron 2112 vectores de soporte (totalSV: 2112), dentro de la variable nSV vienen detallados cuantos para cada clase, mientras que SVs y sv_coef definen los vectores soporte calculados.

- training_label_vector: es el vector de dos columnas de datos de entrenamiento; son datos que deben ser conocidos ser conocidos ya que tenemos que conocer su clase para poder entrenar a la máquina.
- training_instance_matriz: Es la columna de etiquetas del vector de datos de entrenamiento. Debe tener una longitud igual a la del vector training_label_vector y definen la clase del vector de su correspondiente fila. Las etiquetas pueden ser binarias (-1, 1) o positivas para problemas con un número mayor de clases (1, 2, 3…), como es el caso de este trabajo de investigación.
- libsvm options: Son las opciones de caracterización del modelo empleadas por la librería LibSVM.

4.6.2.2. SVMPREDICT

La función svmpredict es la encargada de ejecutar la clasificación basada en SVM, necesita como punto de partida modelos creados a través de la función svmtrain y datos de entrada en el sistema (vectores de posición y etiqueta) y en función de estos inputs clasifica dichos datos.

La llamada a la función se realiza con la siguiente sintaxis:

```
[predicted_label, accuracy, decision_values/prob_estimates] =
svmpredict(testing_label_vector, testing_instance_matrix, model,
'libsvm_options')
```

De donde las salidas de la función son [39]:

- predicted_label: Es un vector columna de etiquetas correspondientes a los datos de entrada, las calcula en función del modelo que se le suministra comparando en qué punto del modelo está el vector de entrada y clasificándolo.
- accuracy: Muestra la fiabilidad total de la clasificación. Como uno de los datos de entrada es la etiqueta de los datos, se compara la etiqueta de entrada con la obtenida de la clasificación y calcula el porcentaje de acierto.
- decisión_values/pro_estimates: Si se selecciona esta variable obtenemos la probabilidad de estimación de los vectores.

Las entradas a la función svmpredict son [39]:

- model: Modelo creado con la función svmtrain. Debe estar en formato variable de la estructura de Matlab.
- testing_label_vector: Vector columna con las etiquetas de los datos de entrada estimadas, que resulta útil para calcular la fiabilidad de la clasificación.
- testing_instance_matrix: Vector de dos columnas que contiene los datos de entrada a la función. Estos datos son de posición en X e Y y son los que contrasta contra el modelo para clasificarlos.
- libsvm options: Son las opciones de caracterización del modelo empleadas por la librería LibSVM.

4.6.3. Función extractHOGFeatures

Otra función importante para el desarrollo del presente trabajo de investigación es la función extractHOGFeatures para encontrar las características del Histograma de Gradientes Orientados (HOG).

Su sintaxis puede expresarse de las 4 formas siguientes:

```
features = extractHOGFeatures(I)
[features,validPoints] = extractHOGFeatures(I,points)
[___, visualization] = extractHOGFeatures(I,___)
[___] = extractHOGFeatures(___,Name,Value)
```

De donde [44]:

- features: extractHOGFeatures(I) retorna las características del descriptor HOG de una imagen de entrada a color o escala de grises (I). las características se devuelven en un vector 1xN donde N es la longitud característica de HOG; las características devueltas codifican información de forma local de regiones dentro de una imagen. Puede utilizar esta información para muchas tareas, incluyendo la clasificación, detección y seguimiento.
- [features,validPoints] = extractHOGFeatures(I,points) retorna las características HOG extraídas alrededor de puntos específicos localizados. La función también retorna validPoints que contiene las ubicaciones de los puntos de entrada cuya

región circundante está totalmente contenida dentro de la información I. La información escala asociada con los puntos se ignora.

- [___, visualization] = extractHOGFeatures(I,___) opcionalmente devuelve una visualización característica HOG, utilizando cualquiera de las sintaxis anteriores. Se puede observar esta visualización usando plot(visualization).
- [___] = extractHOGFeatures(___,Name,Value) utiliza las opciones adicionales especificadas por uno o más nombres, el valor del par de argumentos , utilizando cualquiera de las sintaxis anteriores .

5. MATERIALES Y MÉTODOS

El desarrollo del presente trabajo de investigación fue producto de una serie de pasos que permitieron realizar: la delimitación del tema a desarrollar, el planteamiento de los objetivos, recolección de información, dar solución a los problemas que se presentaron, para de esa manera cumplir cada uno de los objetivos propuestos.

Para determinar el tema de investigación, se hizo uso del método deductivo, es decir, que partiendo de conocimientos generales se determinó las posibles soluciones (casos particulares), en cada una de las fases existentes para el desarrollo del tema; según antecedentes previos se conoce que el lenguaje de señas es extenso y puede variar incluso por cada persona que haga uso de este lenguaje adaptando, creando o modificando señas para su mejor entendimiento ante las demás personas, valiéndose de esta premisa y con el fin de que todas las letras del abecedario sean traducidas, se estableció una pequeña modificación en cuanto a las letras ”j” y “z”, ya que estas representan señas dinámicas, debido a que se hizo uso de técnicas como las máquinas de soporte vectorial para la clasificación y lo hacen sólo para las letras estáticas del abecedario, se decidió establecer una seña para las mencionadas letras y de esa manera completar un total de 26 letras que corresponden al abecedario del lenguaje de señas ecuatoriano para su posterior traducción.

Dentro del hardware que se empleó, se encuentra:

- Computador portátil con CPU (Intel ® Core ™ i5-3317U) 1.70GHz, 6GB RAM, 500 GB de disco duro y sistema operativo Windows 8.
- Dispositivo Kinect Xbox 360.
- Adaptador de Kinect para conectarlo a la PC.

El código fue implementado en Matlab 2014a porque se requirió el uso de las máquinas de soporte vectorial (SVM) para la clasificación y Matlab brinda la posibilidad de trabajar con la Liberia LIBSVM propia para este tipo de problemas, además la interfaz gráfica de usuario fue implementada en este mismo software para mostrar la respectiva traducción de cada una de las letras del abecedario de señas ecuatoriano.

El proceso llevado a cabo para conseguir la traducción de cada una de las señas (26 letras) a través de la mano dominante del usuario, que para este caso se consideró que la persona es diestra, se basa en el procesamiento de una secuencia de imágenes (frames) obtenidas a partir del sensor de profundidad de Kinect. Este proceso se divide en cuatro fases principales:

- Detección de la mano derecha.
- Obtención de las características descriptivas de la misma.
- Clasificación.
- Interpretación de la información obtenida en las etapas anteriores.

En la figura 54 se muestra el esquema que se siguió para dar solución a los objetivos planteados, desglosando cada una de las fases antes mencionadas.

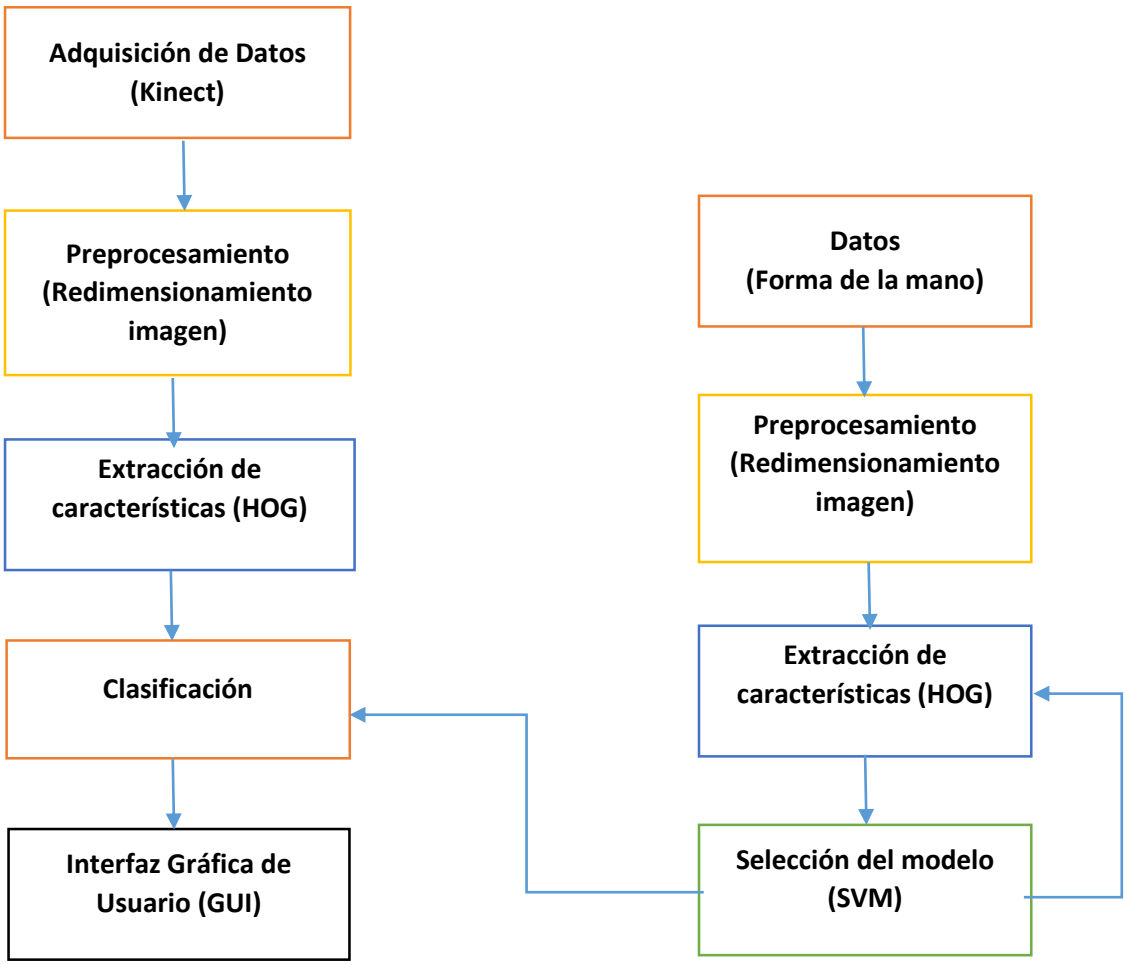


Figura 54. Diagrama del procedimiento seguido para el reconocimiento y traducción.

En la parte izquierda del diagrama de la figura 54, en el primer recuadro se señala la adquisición de los frames para la correspondiente base de datos a través del sensor Kinect, posterior a ello, se procesa cada una de las imágenes tomadas, segmentando el fondo y adecuando el tamaño de las imágenes, en una primera instancia el área capturada de cada una de las imágenes corresponde a un tamaño de 100x100 pixeles, pero mejores resultados se obtuvieron acortando el área a un tamaño de 64x64 pixeles que pertenece únicamente a la zona de la mano del usuario; seguido a este bloque se encuentra la etapa para la extracción de las características que en este caso se lo hace mediante el descriptor HOG, explicado detalladamente en la revisión de literatura, obtenidas las características se entrena una máquina de soporte vectorial adoptando cada una de las características adquiridas en el paso anterior para una posterior clasificación.

En la parte derecha del mismo diagrama se muestra en cambio el proceso que se sigue cuando se van adquiriendo las imágenes del sensor de Kinect en tiempo real para su posterior comparación con los patrones almacenados, a diferencia de la etapa anterior aquí ya se selecciona un modelo previamente entrenado para su posterior clasificación e interpretación de cada una de las señas realizadas por el usuario mostrando el resultado de la traducción a través de una GUI hecha en Matlab.

5.1. Instalación del paquete de compatibilidad de Kinect para Windows

Previo al desglose de cada una de las fases desarrolladas, es importante la instalación de Kinect y sus respectivos paquetes para que funcione correctamente al momento de manipularlo dentro de Matlab, para ello se instaló su paquete respectivo *(Kinect for Windows Sensor)* al escribir en la ventana de comandos de Matlab lo siguiente:

```
>> supportPackageInstaller
```

Con lo que nos apareció una imagen como la que se muestra en la figura 55, en donde se seleccionó la primera opción *(Install from Internet*) por recomendación de la página de Mathworks.

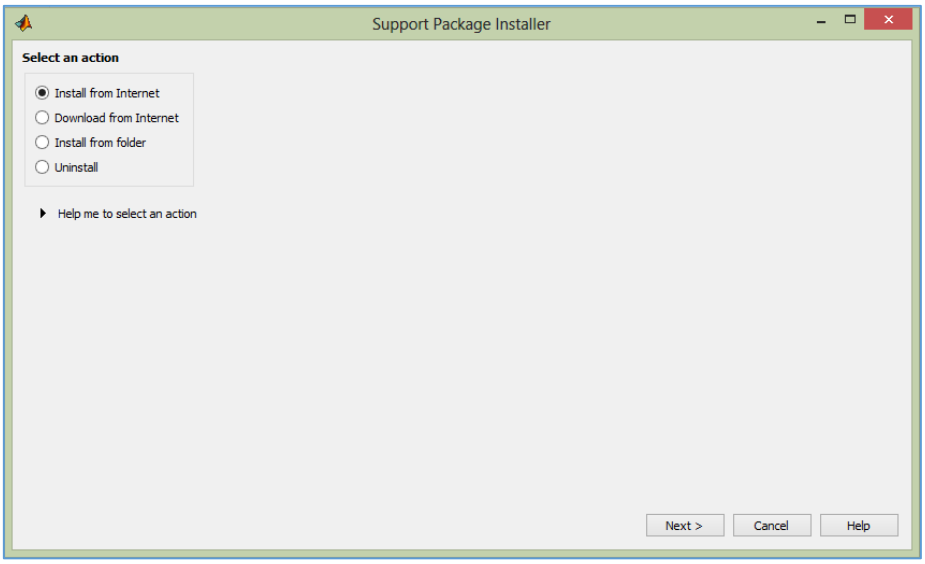


Figura 55. Instalación del paquete de Kinect for Windows Sensor (paso 1).

Seguido a esto, se selecciona el paquete requerido que para nuestro caso es el denominado *Kinect for Windows Sensor*, tal y como se observa la figura 56.

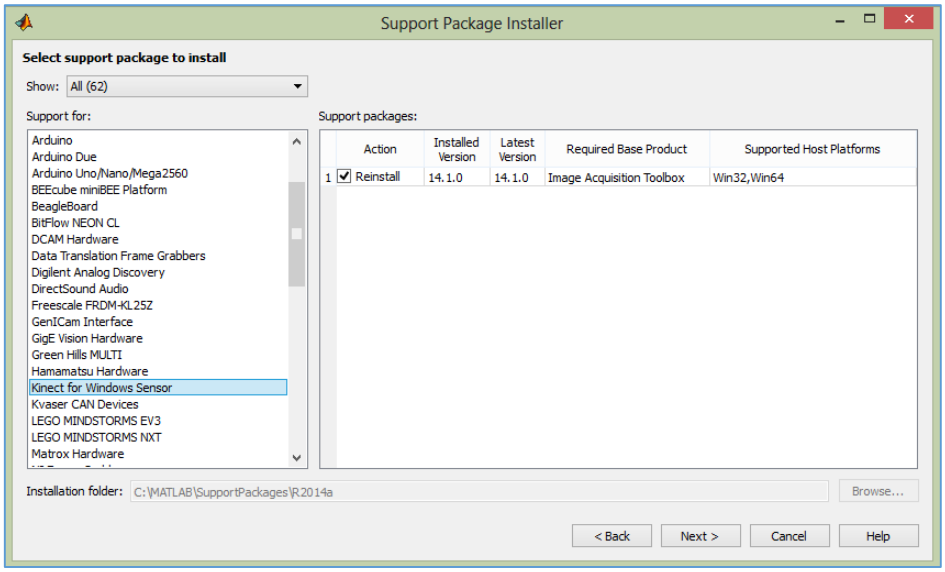


Figura 56. Instalación del paquete de Kinect for Windows Sensor (paso 2).

A continuación, Matlab nos pedirá logearnos para poder descargar e instalar el paquete, lo hacemos y aceptamos los términos como en cualquier otro programa, finalmente se podrá instalar el paquete. (Ver figura 57)

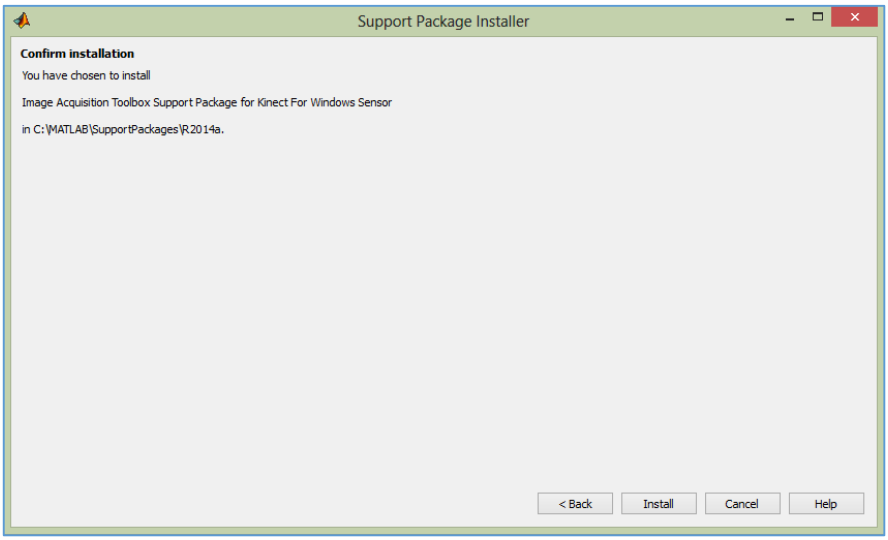


Figura 57. Instalación del paquete de Kinect for Windows Sensor (paso 3).

Las figuras 55, 56, y 57 muestran cómo se instaló el paquete necesario para que el dispositivo Kinect sea usado adecuadamente con Matlab, cabe resaltar que este paso fue esencial previo al desarrollo del código implementado en el mencionado software para el diseño del traductor de las 26 letras del abecedario de señas ecuatoriano. En la figura 58 se muestra la comprobación de que Kinect está instalado correctamente en Matlab a través del comando *imaqhwinfo.*

```
>> imaqhwinfo

ans =

    InstalledAdaptors: {'kinect'}
        MATLABVersion: '8.3 (R2014a)'
          ToolboxName: 'Image Acquisition Toolbox'
       ToolboxVersion: '4.7 (R2014a)'
```

Figura 58. Dispositivo Kinect instalado en Matlab.

Una vez realizado esto, se procedió con la ejecución de cada una de las fases, a continuación se detalla cada una de ellas.

5.2. Fase 1: Detección de la mano

Dentro de la fase uno se encuentra la *detección* de la mano, para ello, en primera instancia se empieza recolectando los datos (frames) tanto de 100x 100 pixeles como de 64x64 pixeles, con el sensor de profundidad de Kinect, con el fin de que el procesamiento de las imágenes sea el menor posible, es decir, el coste computacional sea mínimo, facilitando la segmentación de las imágenes capturadas de la mano derecha del usuario, sin embargo, también se realizaron pruebas capturando frames de la cámara de color de Kinect, en un fondo negro con una tamaño de imágenes de 64x64 pixeles, obteniendo mejores resultados.

De esta manera en Matlab, primeramente se identifica de entre todos los puntos de las articulaciones que Kinect es capaz de reconocer, aquella que corresponde a la mano derecha del usuario, que como se había mencionado pertenece a una persona diestra, capturando imágenes con un tamaño de 100x100 pixeles, que abarca parte del antebrazo y la mano del usuario; luego se realizaron pruebas reduciendo el área de captura a un tamaño de 64x64 pixeles, que por el contrario abarca únicamente la mano del ejecutante de la seña.

5.3. Fase 2: Obtención de características

En el momento en que la detección es satisfactoria, es decir, se dispone de una región, se prosigue con la fase dos, la *obtención de características*, a partir del Histograma de Gradientes Orientados (HOG) aplicando esta técnica para obtener los gradientes de cada una de las imágenes tomadas en la fase uno.

El objetivo de esta fase es diferenciar los pixeles de la imagen que pertenecen al contorno del objeto de estudio, que en este caso corresponde a la mano, del resto de pixeles de la imagen, posteriormente a partir de esto se obtiene una estructura que describen puntos y características relevantes de la mano.

Primeramente se procesó cada una de las imágenes almacenadas en la base de datos, para ello se adecua la imagen de tal forma que el tamaño de todas ellas sea el mismo, específicamente de 100x100 pixeles y de 64x64 pixeles como se mencionó en la fase uno, y se las convierte en imágenes binarias, calculando un nivel de umbral global de tal manera que se logre transformar una imagen en intensidad a una imagen binaria.

Se convierte cada una de las imágenes de la base de datos a imágenes binarias con el fin de mejorar el rendimiento y simplificar cálculos posteriores, sobre estas imágenes se procedió a extraer las características descriptivas, es decir, las características del histograma de gradientes orientados (HOG) con la función "*extractHOGFeatures*" de Matlab, dicha función permite el cálculo de los gradientes de intensidad con el fin de identificar los puntos de la imagen donde hay una mayor variación de intensidad, por lo general esta variación se maximiza localmente en los pixeles donde aparece la transición entre la superficie de dos objetos, estos puntos coincidirán con puntos de borde o frontera del objeto de interés (mano), entonces el gradiente se calcula tanto en la dirección horizontal como en la vertical, se trabajó con un tamaño de celdas de 2x2 pixeles, en las imágenes tanto las de 100x100 pixeles como las de 64x 64 pixeles.

5.4. Fase 3: Clasificación

En la fase tres la de *clasificación*, a partir de los datos entrenados, que resultan de la fase dos, de la mano derecha, tanto positivas (aparecen manos), como negativas (no aparecen manos) haciendo un total de dos clases para cada una de las letras, se entrena un detector usando la técnica de aprendizaje denominada Maquinas de Soporte Vectorial (SVM), para crear un modelo de clasificación de las clases antes mencionadas, en esta fase se hace uso de la librería LIBSVM que se encuentra disponible para Matlab, previamente explicada en la revisión de literatura.

5.5. Fase 4: Interpretación de los datos

Finalmente para la fase de *interpretación* de los datos se implementa una interfaz gráfica de usuario que captura los frames del sensor de profundidad en tiempo real para su respectiva traducción, dicha interfaz es lo más sencilla y amigable posible, con el fin de que los recursos computacionales sean mínimos, y el usuario logre manejarla con facilidad; posee tres botones para su manejo: Iniciar, Detener y Salir, con el primero se da inicio al sensor Kinect para que empiece a tomar los datos en tiempo real, con el segundo se detiene la captura de los mismos, y con el último se muestra una ventana que nos pregunta si queremos salir realmente de la interfaz; cabe resaltar que como se hicieron varias pruebas con las distintas cámaras que el sensor posee, los cuadros que se muestren en la interfaz aumentan al trabajar reduciendo el área de captura, es decir, aparece un cuadro más en dicha interfaz cuando las imágenes corresponden a una área de 64x64 pixeles, tal y como se mostrará en el apartado de resultados.

6. RESULTADOS

Corresponden al desarrollo práctico del tema de investigación, para ello, tal y como se ha mencionado en la metodología, el proceso del presente trabajo se ha dividido en cuatro fases, las mismas que ayudaron a cumplir a cabalidad con cada uno de los objetivos propuestos.

6.1. Fase 1: Detección de la mano

Mediante Matlab se logró detectar e identificar claramente la mano derecha del usuario, que como se supone es una persona diestra, en la figura 59 mediante el rectángulo amarillo se muestra una de las capturas de las imágenes que corresponde a un tamaño de 100x100 pixeles; mientras que en la figura 60 mediante el rectángulo verde se muestra en cambio la que corresponde a un tamaño de 64x64, en ambos casos, las imágenes fueron capturadas a una distancia mínima de 1m, y en fondos que no son blancos, se observa claramente como el área se reduce significativamente en la figura 60, dejando para el posterior procesamiento imágenes que corresponden únicamente a la zona de la mano, lo que no sucede en las primeras pruebas (figura 59) donde se procesa también parte del antebrazo del usuario.

Figura 59. Detección de la mano derecha (100x100 pixeles).

Figura 60. Detección de la mano derecha (tamaño de 64x64 pixeles).

En la figura 61 se observa las imágenes que se capturan tanto de la cámara de color como del sensor de profundidad, con el fin de constatar que efectivamente se reconoce la mano derecha en tiempo real.

Figura 61. Seguimiento de la mano derecha. Imagen de la cámara de color (izq.). Imagen del sensor de profundidad (der.)

En la base de datos que se formó, se tiene por cada letra 100 imágenes haciendo un total de 2600 imágenes de 100x100 pixeles, que corresponden a cada una de las letras del abecedario, además se resalta el hecho de que para esta fase ya se implementa una GUI para que se vayan capturando cada una de las imágenes necesarias para conformar la base de datos antes mencionada, dicha interfaz se puede observar en la figura 62.

Figura 62. GUI para la primera fase (detección de la mano derecha), imágenes de 100x100 pixeles.

Se resalta el hecho de que se crea una seña para la letra “j” y para la “z” (tabla 4) valiéndose de la premisa de que las personas que hacen uso del lenguaje dactilológico pueden adecuar las señas para su mejor entendimiento ante los demás, ya que estas señas corresponden a señas dinámicas de este lenguaje y las máquinas de soporte vectorial ayudan a la clasificación de las señas estáticas como las demás letras, esto se hace con el fin de no exceptuar ninguna letra para su traducción y al ser este únicamente un prototipo pudiendo ser mejorado en futuros trabajos e investigaciones.

Tabla 4. Señas para las letras "j" y "z" del abecedario.

Letra j	Letra z
	

Como resultado en la tabla 5 se muestra la base de datos para su posterior entrenamiento.

Tabla 5. Base de datos con imágenes de cada una de las señas del abecedario (100x100 pixeles).

Letra a	Letra b	Letra c
		
Letra d	**Letra e**	**Letra f**
		

Letra g 	**Letra h** 	**Letra i** 
Letra j 	**Letra k** 	**Letra l** 
Letra m 	**Letra n** 	**Letra o** 
Letra p 	**Letra q** 	**Letra r** 
Letra s 	**Letra t** 	**Letra u** 

Letra v	Letra w	Letra x
		
Letra y	**Letra z**	
		

En el caso donde las imágenes capturadas fueron de 64x64 pixeles se procedió de igual manera, con la única variación de que ya se habilitó la cámara de color de Kinect para la captura de las imágenes, tal y como se observa en la figura 63.

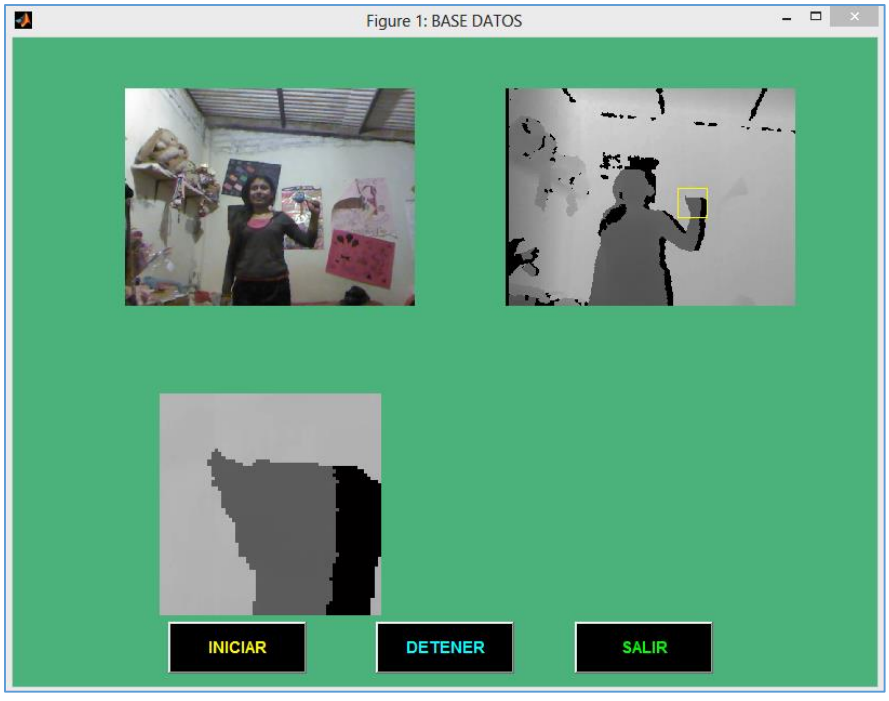


Figura 63. GUI para la primera fase (detección de la mano derecha), imágenes de 64x64 pixeles.

La base de datos que se formó con estas imágenes se reduce a la mitad de las obtenidas en la prueba anterior, es decir, se tiene un total de 1300 imágenes; en la tabla 6 se muestra como quedó estructurada esta base de datos.

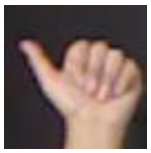
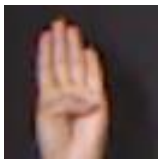
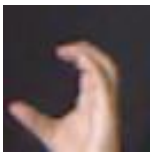
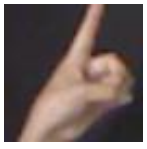
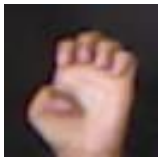
Tabla 6. Base de datos con imágenes de cada una de las señas del abecedario (64x64 pixeles).

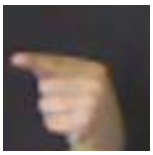
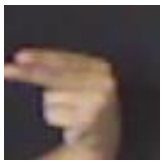
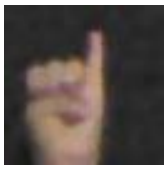
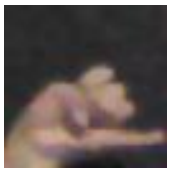
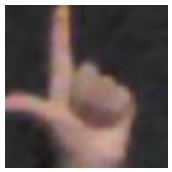
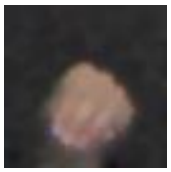
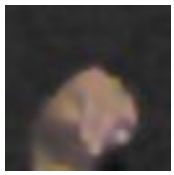
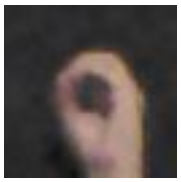
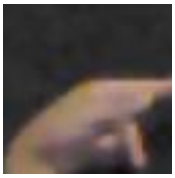
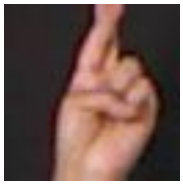
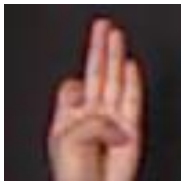
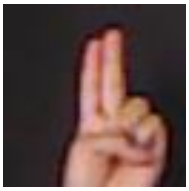
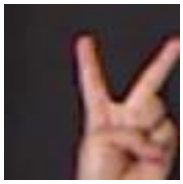
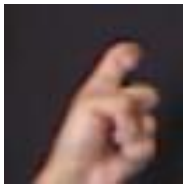
Letra a 	**Letra b** 	**Letra c** 
Letra d 	**Letra e** 	**Letra f** 
Letra g 	**Letra h** 	**Letra i** 
Letra j 	**Letra k** 	**Letra l** 
Letra m 	**Letra n** 	**Letra o** 
Letra p 	**Letra q** 	**Letra r** 

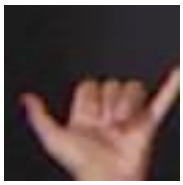
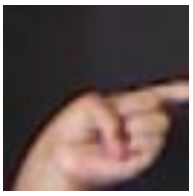
Letra s 	**Letra t** 	**Letra u** 
Letra v 	**Letra w** 	**Letra x** 
Letra y 	**Letra z** 	

En la tabla 7, por el contrario se muestra la base de datos formada con las imágenes provenientes de la cámara de color del sensor, al igual que en el caso anterior se tiene un total de 1300 imágenes.

Tabla 7. Base de datos con imágenes de cada una de las señas del abecedario tomadas con la cámara de color, (64x64 pixeles).

Letra a 	**Letra b** 	**Letra c** 
Letra d 	**Letra e** 	**Letra f** 

Letra g 	**Letra h** 	**Letra i** 
Letra j 	**Letra k** 	**Letra l** 
Letra m 	**Letra n** 	**Letra o** 
Letra p 	**Letra q** 	**Letra r** 
Letra s 	**Letra t** 	**Letra u** 
Letra v 	**Letra w** 	**Letra x** 

Letra y	Letra z	
		

Las imágenes que forman la base de datos, en los tres casos, se las capturó en un ambiente donde no se necesita de luz artificial para la iluminación y en fondos que no son blancos; al trabajar con los datos provenientes del sensor de profundidad de Kinect, en la primera prueba (100x100 pixeles – sensor de profundidad) se obtuvo una traducción equivalente al 76.92% ya que el prototipo no reconoce las siguientes letras: a, e, m, n, o y s; en la segunda prueba (64x64 pixeles – sensor de profundidad) se logró una traducción del 84.6% donde el prototipo no reconoce las siguientes letras: h, m, n y o; finalmente en la última prueba (64x64 pixeles – cámara de color) se logró una traducción que corresponde al 73.07% ya que las letras: a, g, h s, n, o y t no se reconocen con total exactitud; cabe resaltar que en esta última prueba influye en un gran porcentaje el ambiente donde se capturan cada una de las imágenes y la mucha o poca iluminación que exista en el lugar, es por ello que la captura de las mismas se las tomó en un fondo negro, lo que no sucede al trabajar con el sensor de profundidad, en este caso el fondo no debería ser blanco y la iluminación no afectaría en los resultados obtenidos previamente.

6.2. Fase 2: Obtención de características.

Aplicando la función extractHOGFeatures se obtuvo cada una de las características descriptivas de las imágenes almacenadas en la base de datos, para el posterior entrenamiento de la máquina de soporte vectorial., consiguiendo lo que se observa en la figura 67, en este caso se ha tomado muestras específicas que corresponden a las letras “y”, “w”, y “a” del alfabeto de señas ecuatoriano respectivamente.

Antes de ello se convirtió cada una de las imágenes de la base de datos en imágenes binarias para su mejor procesamiento, obteniendo lo que se observa en las figuras 64, 65 y 66, este proceso se aplicó en las tres pruebas realizadas con el objetivo de que el coste computacional sea el menor posible, especialmente cuando se trabaja con imágenes RGB.

Figura 64. Preprocesamiento de las imágenes almacenadas en la base de datos (letra “y”). Imagen original (izq.), Imagen binaria (der.), (100x100 pixeles).

Figura 65. Preprocesamiento de las imágenes almacenadas en la base de datos (letra “w”). Imagen original (izq.), Imagen binaria (der.), (64x64 pixeles).

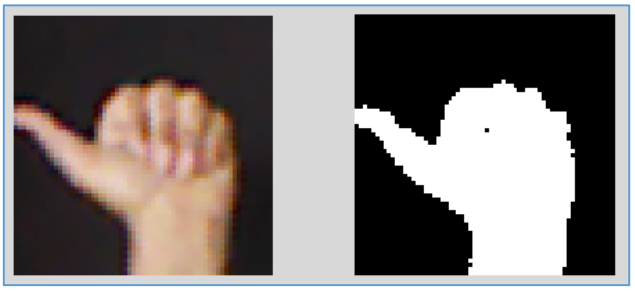


Figura 66. Preprocesamiento de las imágenes almacenadas en la base de datos (letra “a”). Imagen original (izq.), Imagen binaria (der.), (64x64 pixeles).

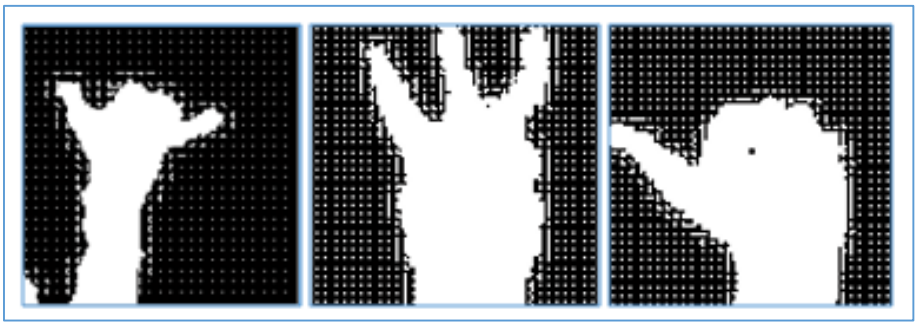


Figura 67. HOG para las letras "y", "w" y "a" de izquierda a derecha, respectivamente.

Para poder realizar el entrenamiento de los datos, fue necesario generar un arreglo que almacene las etiquetas correspondientes a cada una de las letras del abecedario (26 letras), es decir, a partir del conjunto de ejemplos de entrenamiento (muestras) se etiqueta las clases para entrenar una SVM capaz de construir un modelo que prediga la clase de la nueva muestra, entonces en el código implementado en Matlab se guardó tanto la variable de las características como de las etiquetas, el término etiqueta se refiere al número al que corresponde cada letra del abecedario, por ejemplo, se tiene un total de 26 letras en todo el abecedario y a cada una de ellas le corresponde un número dependiendo de la letra a la que se refiere, la letra "a" tiene como etiqueta el "1", la letra "c" al estar en la tercera posición tiene como etiqueta el "3", la letra "e" con una etiqueta de "5" y así sucesivamente.

Hecho esto se procedió a generar el modelo SVM que ayudó a la clasificación y posterior traducción de cada una de las señales denominada "*svmstruct*", para ello se entrena los datos que hasta el momento ya se generaron con la función "*svmtrain*" que trabaja conjuntamente con las variables previamente almacenadas y guardadas (características y etiquetas).

Entrenado el modelo, se realizó pruebas para la verificación del mismo, para ello se creó una tabla de ceros que ayudara a mostrar el resultado de todo el test, y con la ayuda de la función "*svmpredict*", como su nombre lo indica, se predice el posible resultado a generar cuando se capturen los datos a través del sensor de profundidad de Kinect, dicha función trabaja con el modelo previamente entrenado (svmstruct), la función *svmpredict* devuelve la clasificación exacta y la etiqueta predicha.

6.3. Fase 3: Clasificación

Una vez que el modelo fue entrenado y probado se procedió a generar una función que llame a todos los parámetros previamente creados, como las etiquetas predichas y las características descriptivas de la imagen que se almacenan en "*svmstruct"*, ya que mediante esta función se logró capturar los datos que el usuario emite en tiempo real para la comparación con el modelo previamente evaluado y de esta manera mostrar la traducción en una interfaz gráfica de usuario, cabe resaltar que en esta etapa también se llevó a cabo un tratamiento de las imágenes que se capturan en tiempo real como el redimensionamiento de la imagen y la segmentación del fondo de la misma con el fin de lograr comparar eficazmente los datos ya almacenados con los que se toman en ese momento.

Gracias a la función "*svmpredict*" se logra trabajar tanto con los datos que se toman en tiempo real como con los que ya se tienen almacenados, y mediante el comando "*insertText*" disponible en Matlab para colocar texto en imágenes, se logró mostrar en una tabla previamente creada, la traducción de cada una de las señas que se hacen representado a las letras del alfabeto de señas ecuatoriano, para lograr dicha traducción se trabajó con el comando "*char*" que devuelve cada una de las letras del abecedario al trabajar con los caracteres del código ascci.

6.4. Fase 4. Interpretación de los datos

En la figura 68 se observa la interfaz concluida dotada de las opciones que esta posee para su manejo.

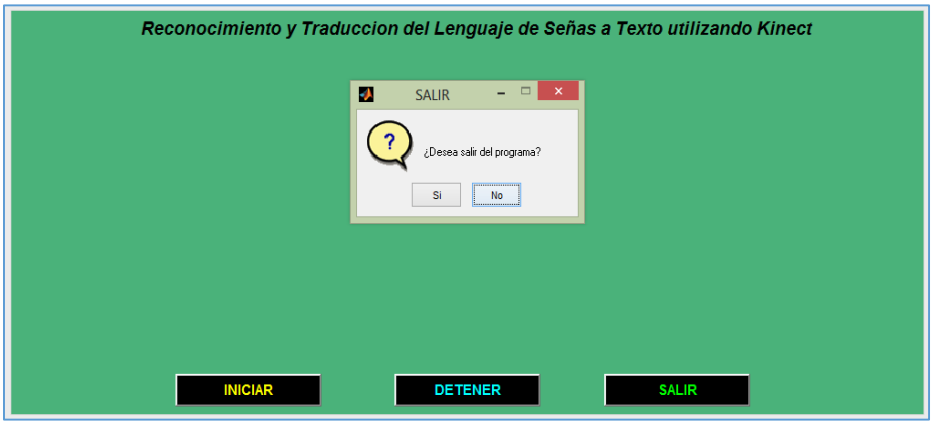


Figura 68. Interfaz Gráfica de Usuario (GUI) diseñada para el prototipo.

A continuación se muestra la traducción de cada una de las señas correspondientes a las letras del abecedario, estas imágenes corresponden a la primera prueba realizada.

Figura 69. Traducción de la letra "A".

Figura 70. Traducción de la letra "B".

Figura 71. Traducción de la letra "C".

Figura 72. Traducción de la letra "D".

Figura 73. Traducción de la letra "E".

Figura 74. Traducción de la letra "F".

Figura 75. Traducción de la letra "G".

Figura 76. Traducción de la letra "H".

Figura 77. Traducción de la letra "I".

Figura 78. Traducción de la letra "J".

Figura 79. Traducción de la letra "K".

Figura 80. Traducción de la letra "L".

Figura 81. Traducción de la letra "M".

Figura 82. Traducción de la letra "N".

Figura 83. Traducción de la letra "O".

Figura 84. Traducción de la letra "P".

Figura 85. Traducción de la letra "Q".

Figura 86. Traducción de la letra "R".

Figura 87. Traducción de la letra "S".

Figura 88. Traducción de la letra "T".

Figura 89. Traducción de la letra "U".

Figura 90. Traducción de la letra "V".

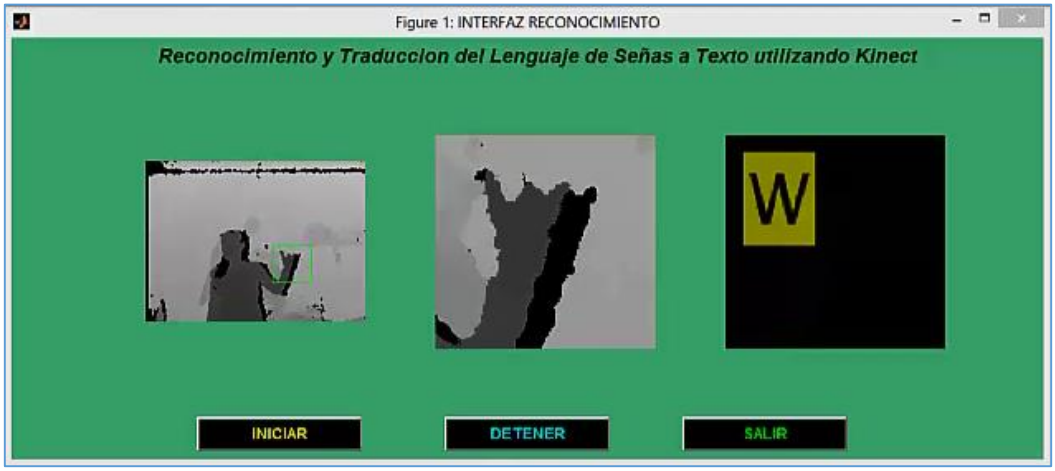


Figura 91. Traducción de la letra "W".

Figura 92. Traducción de la letra "X".

Figura 93. Traducción de la letra "Y".

Figura 94. Traducción de la letra "Z".

A continuación se muestra la traducción de cada una de las señas correspondientes a las letras del abecedario con las imágenes de 64x64 pixeles, es decir, la segunda prueba realizada.

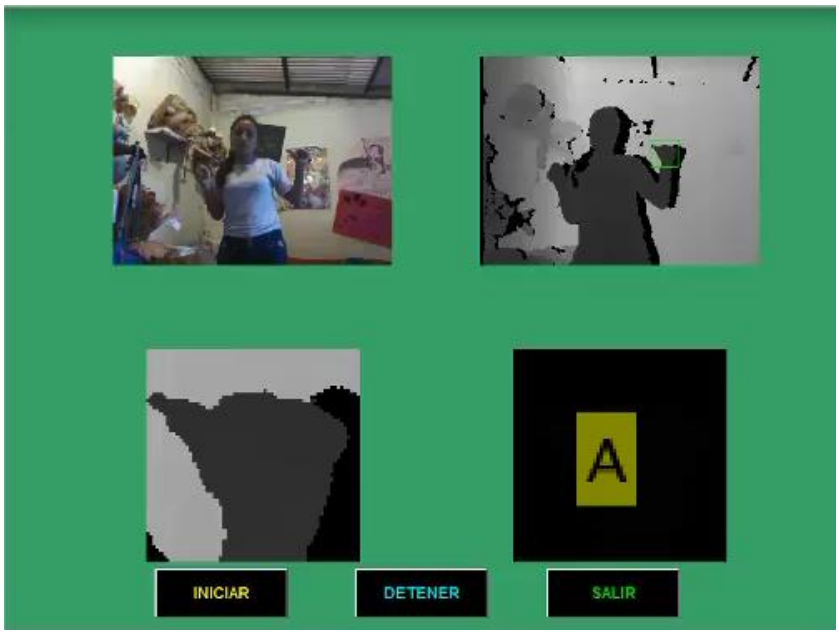


Figura 95. Traducción de la letra "A".

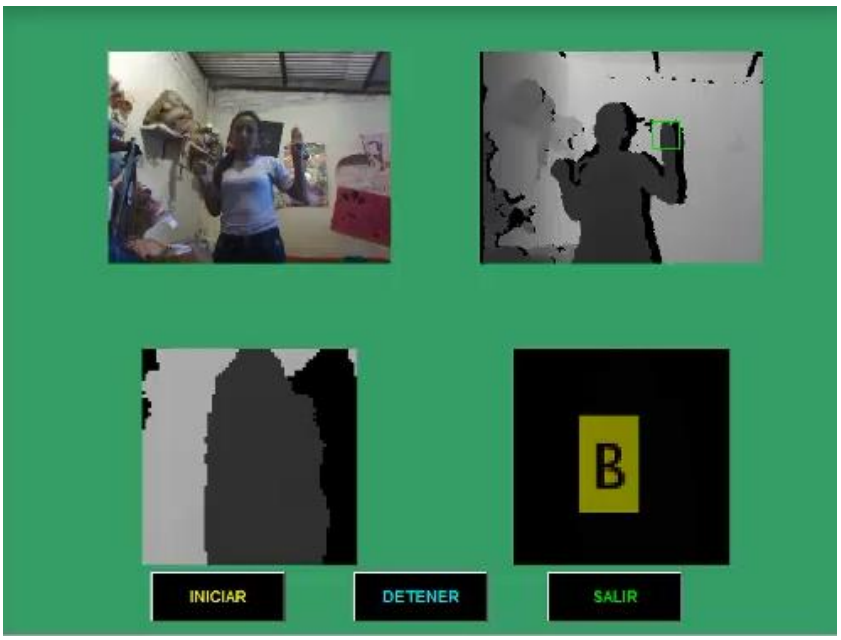


Figura 96. Traducción de la letra "B".

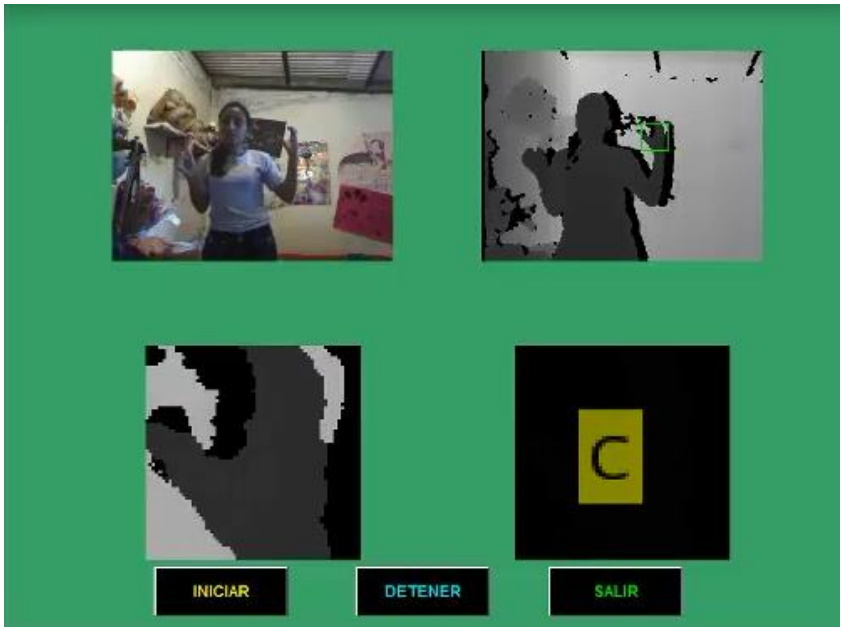


Figura 97. Traducción de la letra "C".

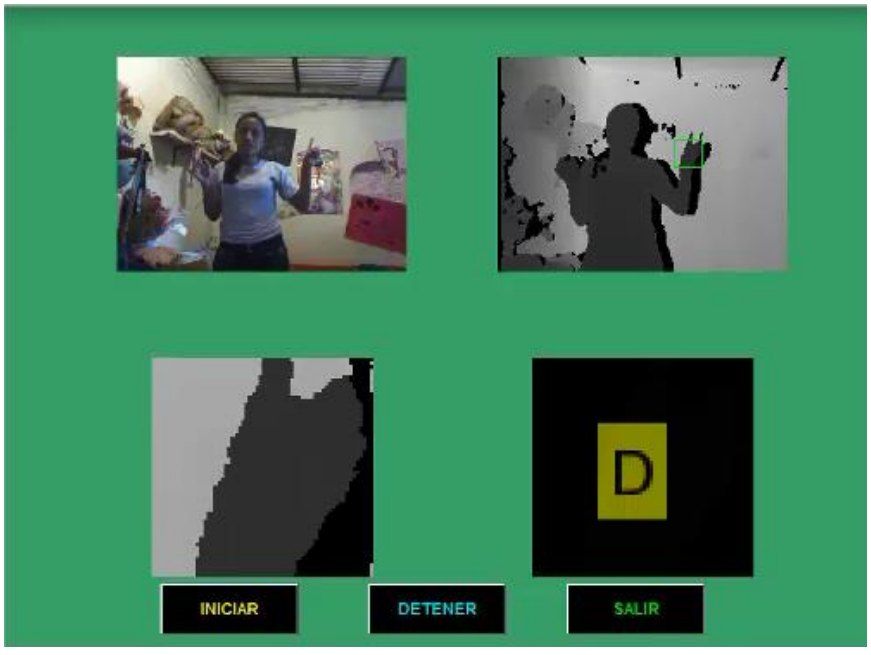


Figura 98. Traducción de la letra "D".

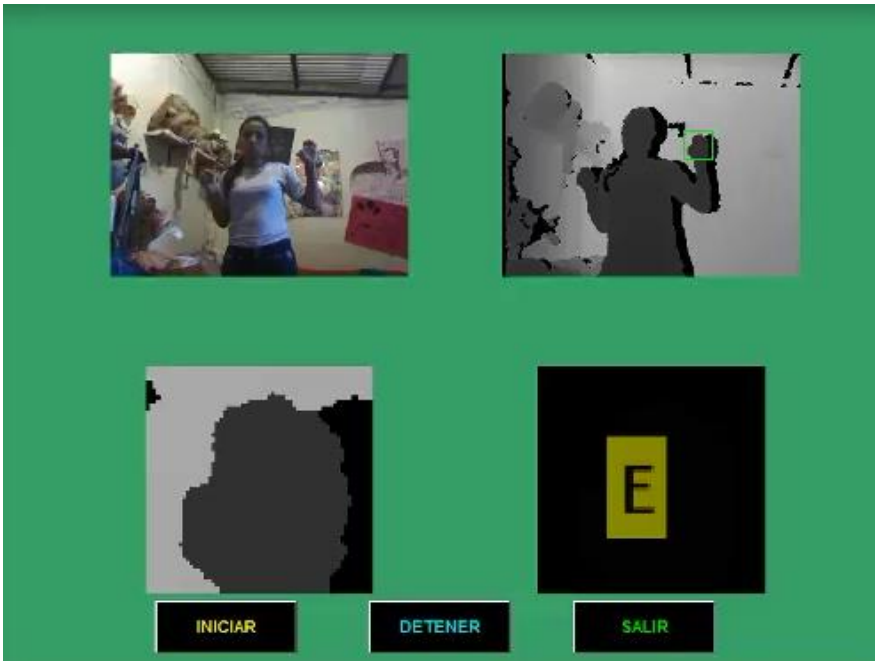


Figura 99. Traducción de la letra "E".

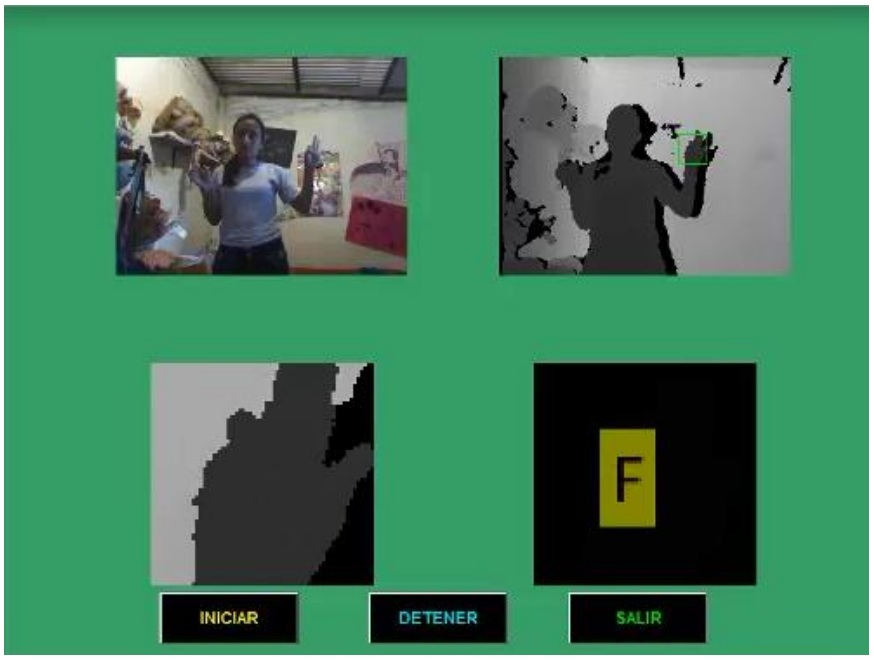


Figura 100. Traducción de la letra "F".

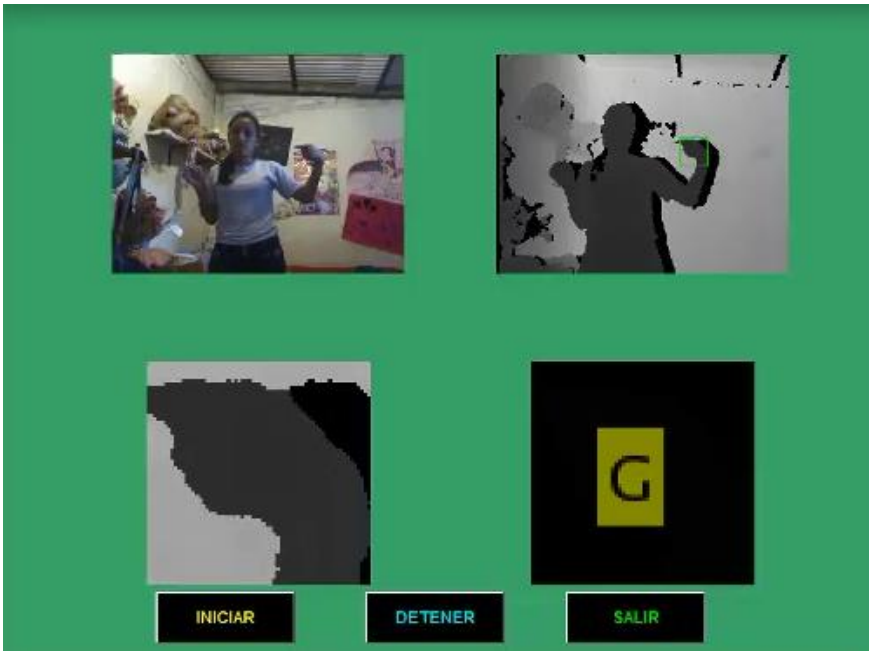


Figura 101. Traducción de la letra "G".

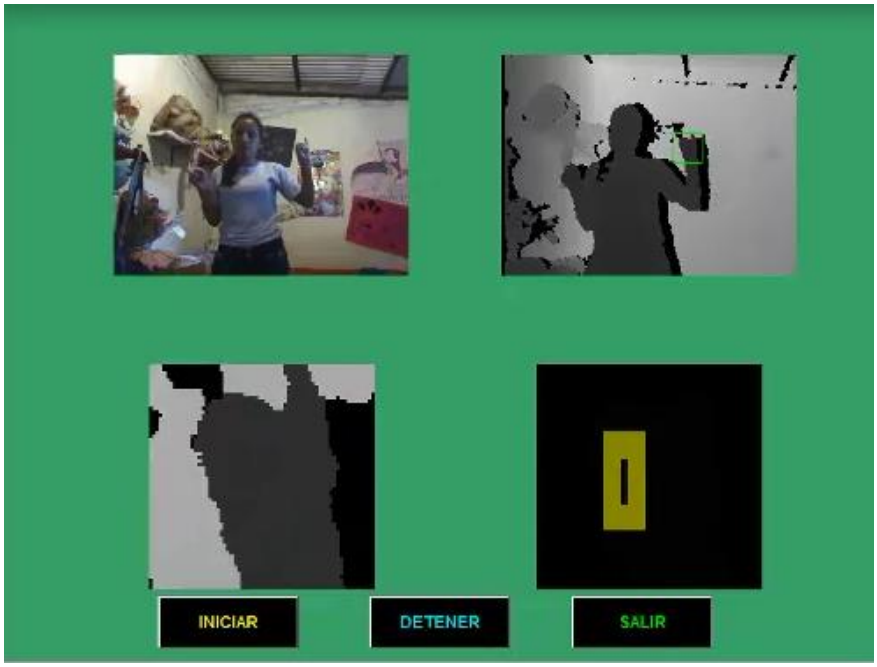


Figura 102. Traducción de la letra "I".

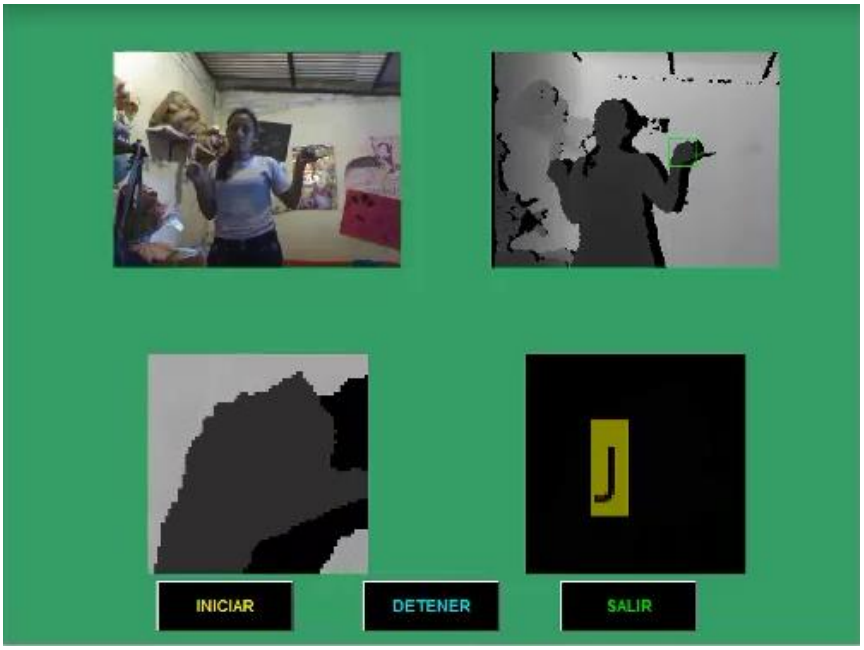


Figura 103. Traducción de la letra "J".

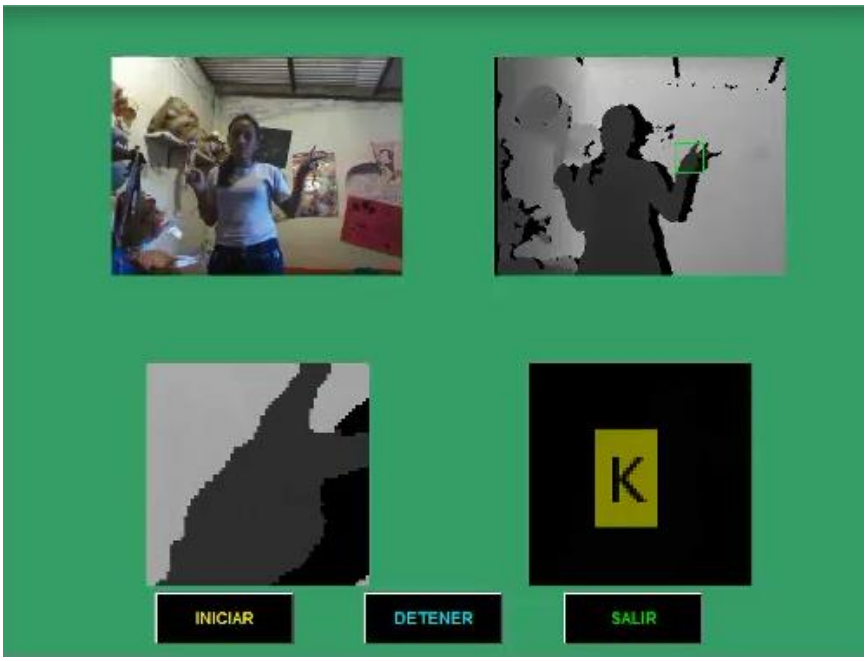


Figura 104. Traducción de la letra "K".

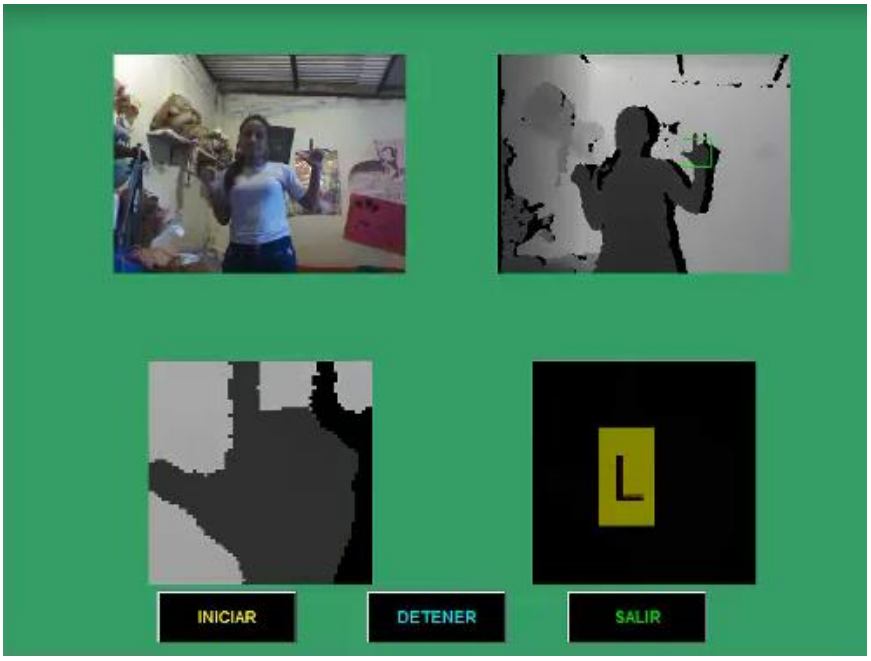


Figura 105. Traducción de la letra "L".

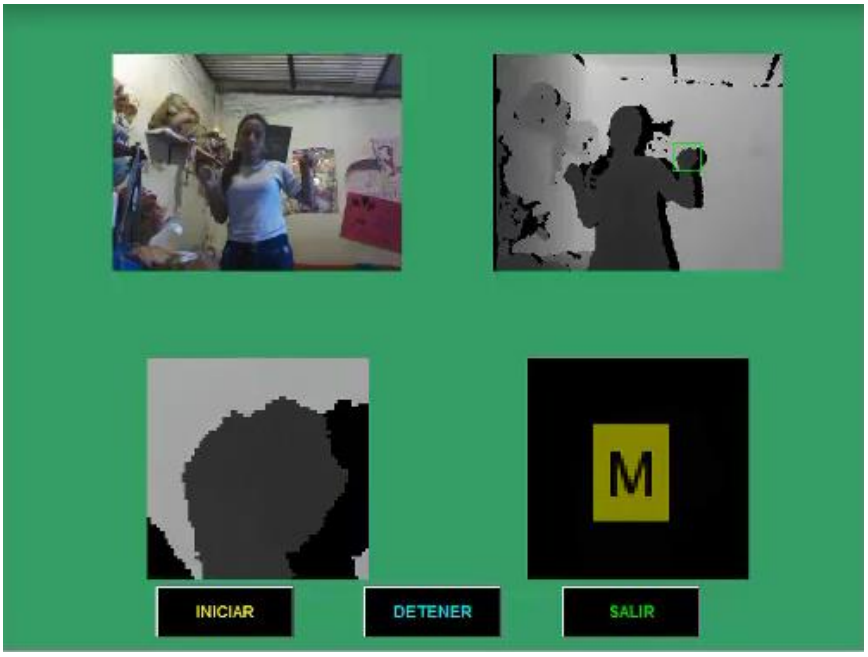


Figura 106. Traducción de la letra "M".

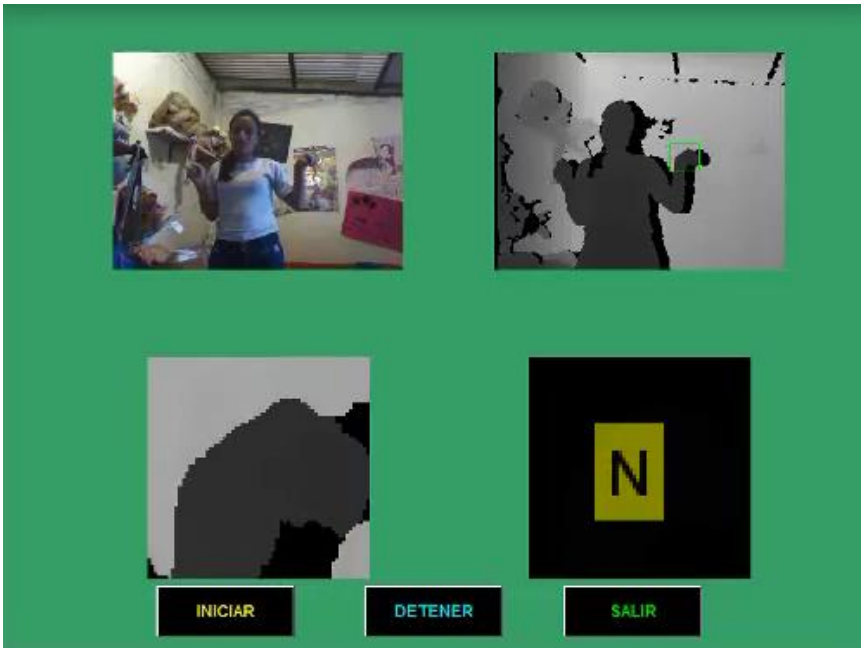


Figura 107. Traducción de la letra "N".

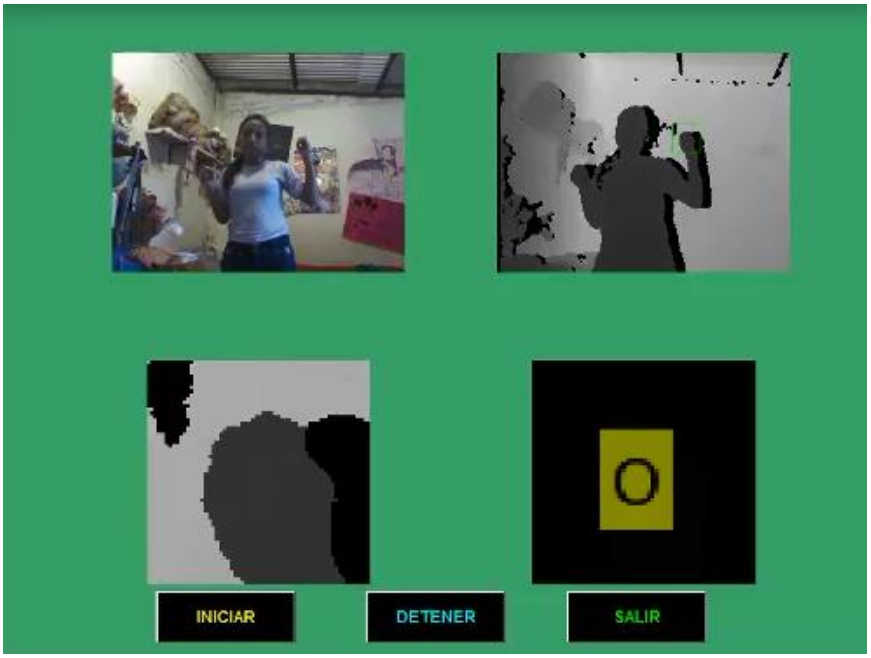


Figura 108. Traducción de la letra "O".

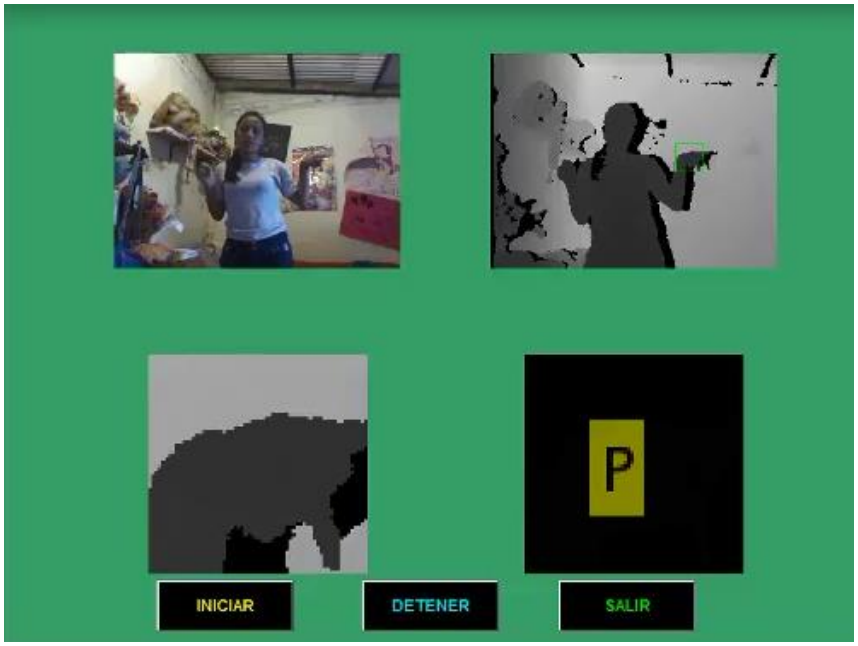


Figura 109. Traducción de la letra "P".

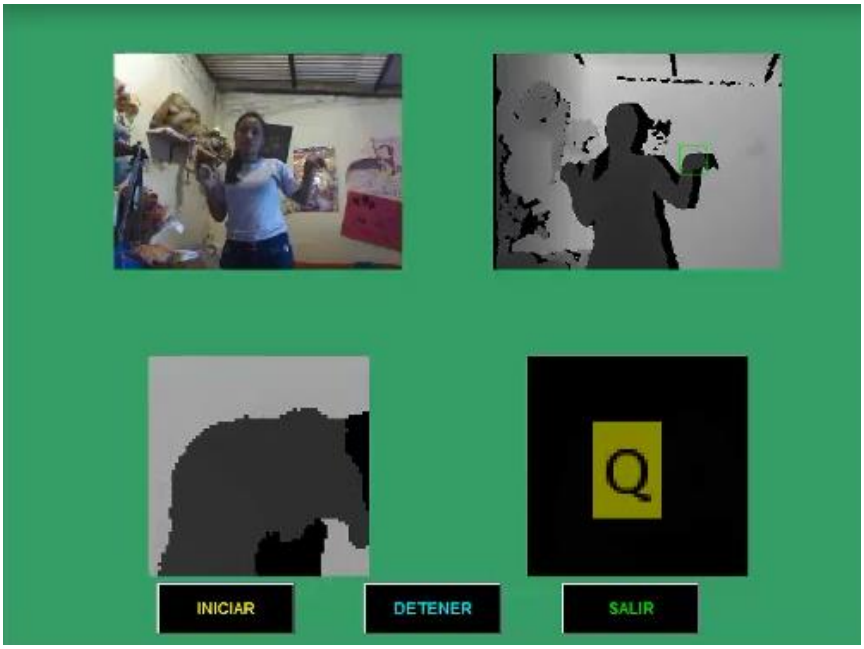


Figura 110. Traducción de la letra "Q".

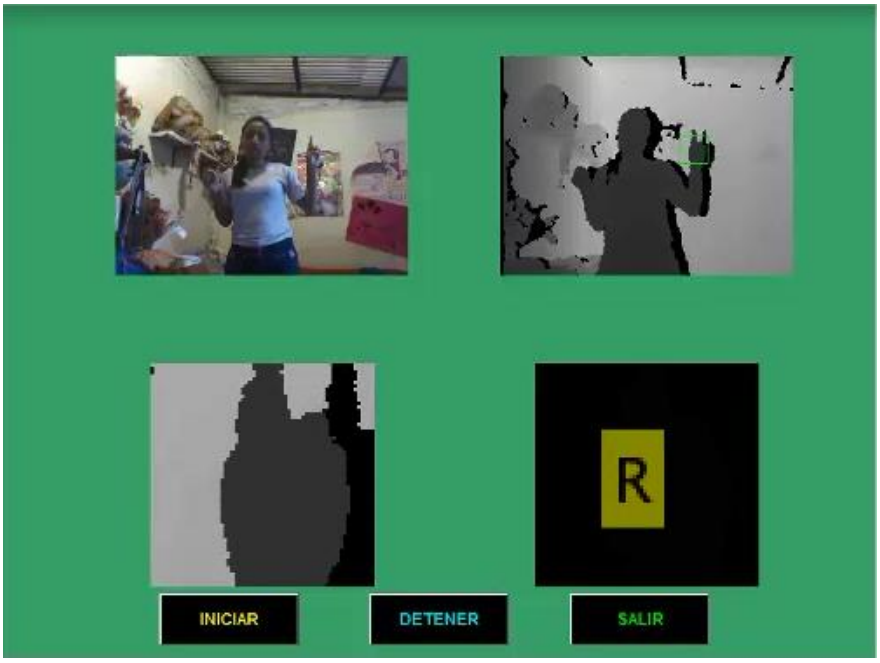


Figura 111. Traducción de la letra "R".

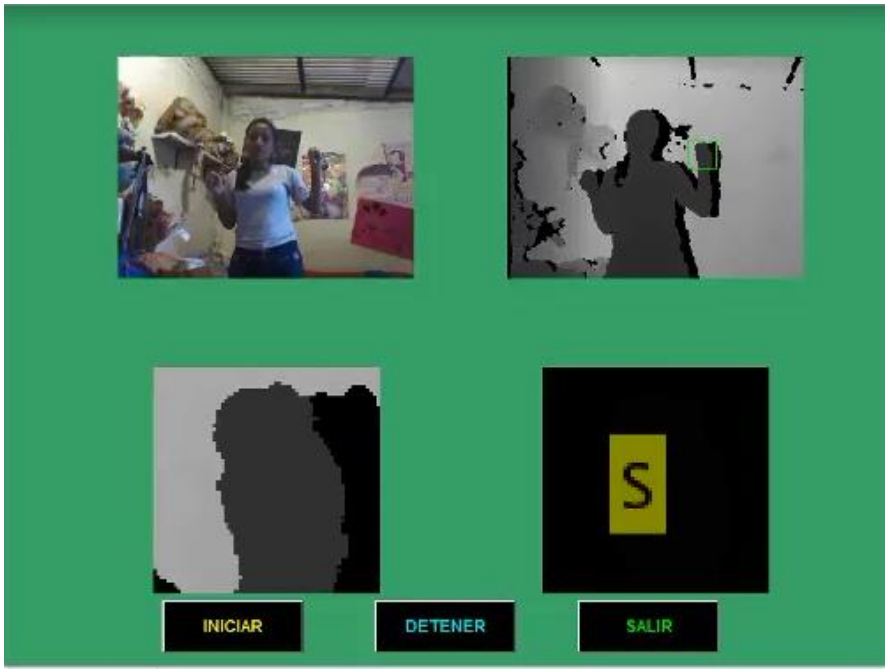


Figura 112. Traducción de la letra "S".

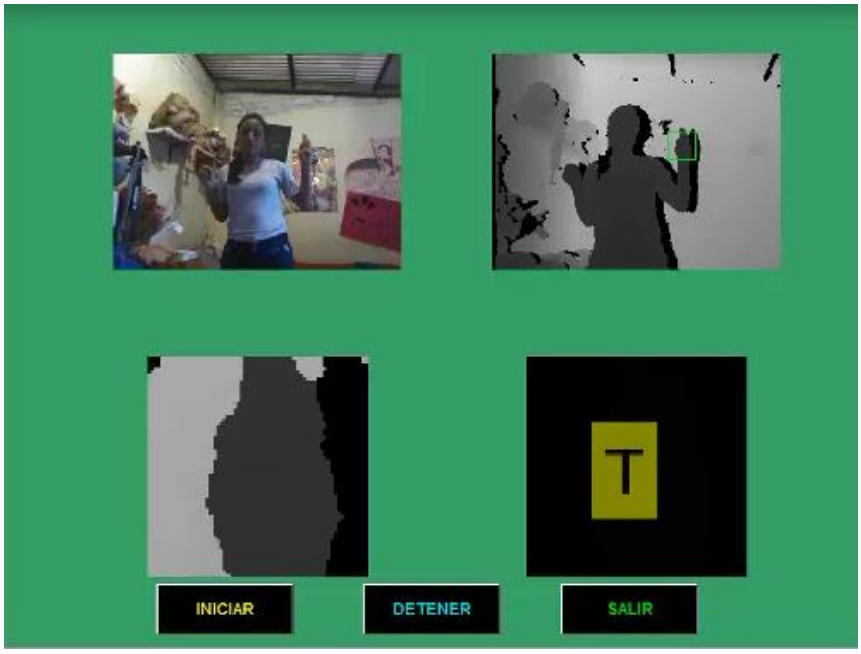


Figura 113. Traducción de la letra "T".

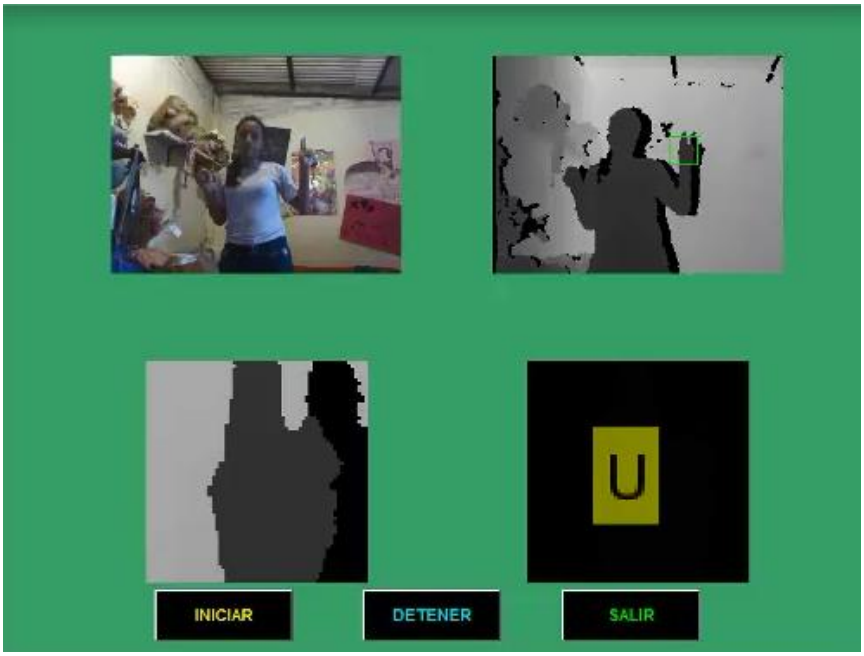


Figura 114. Traducción de la letra "U".

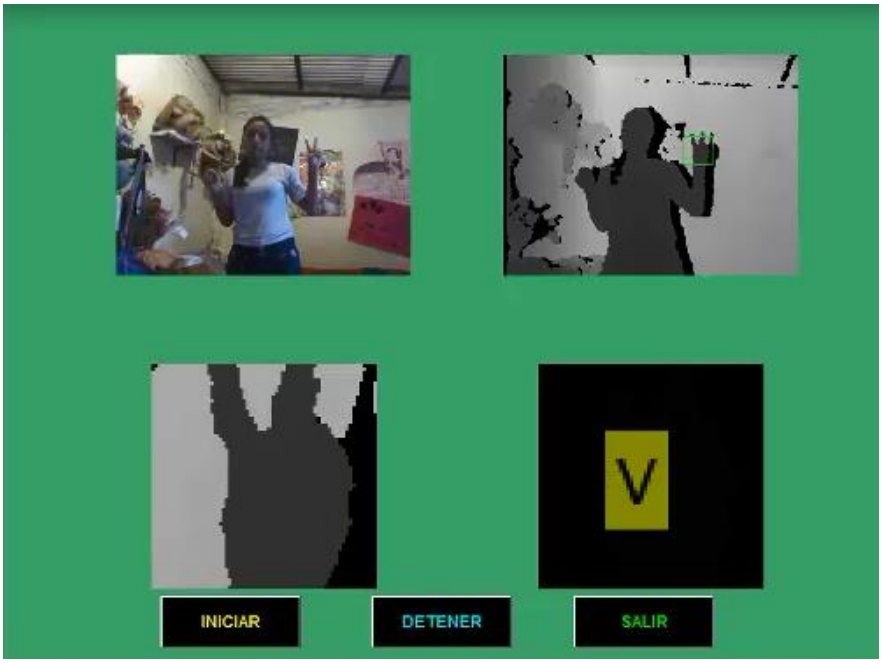


Figura 115. Traducción de la letra "V".

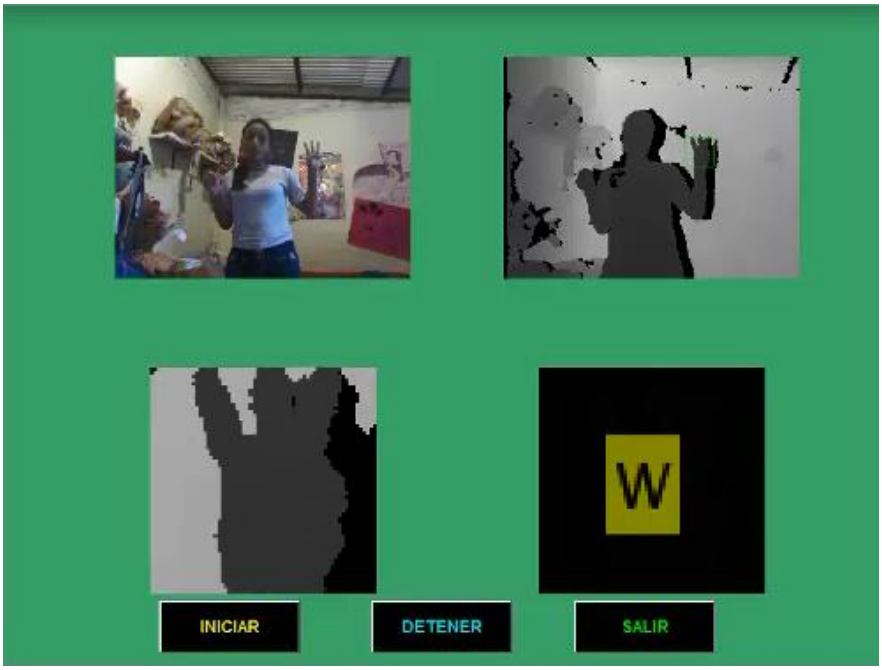


Figura 116. Traducción de la letra "W".

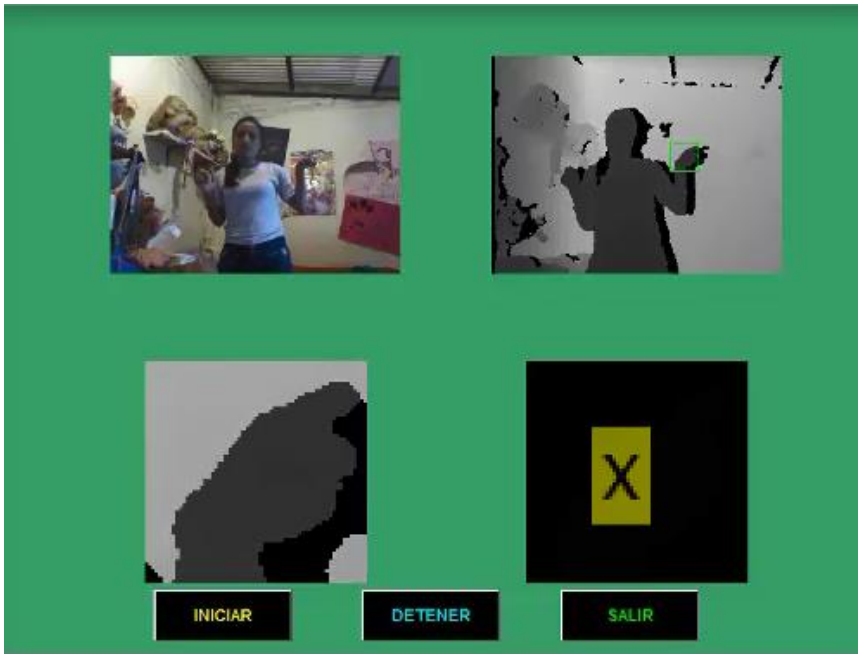


Figura 117. Traducción de la letra "X".

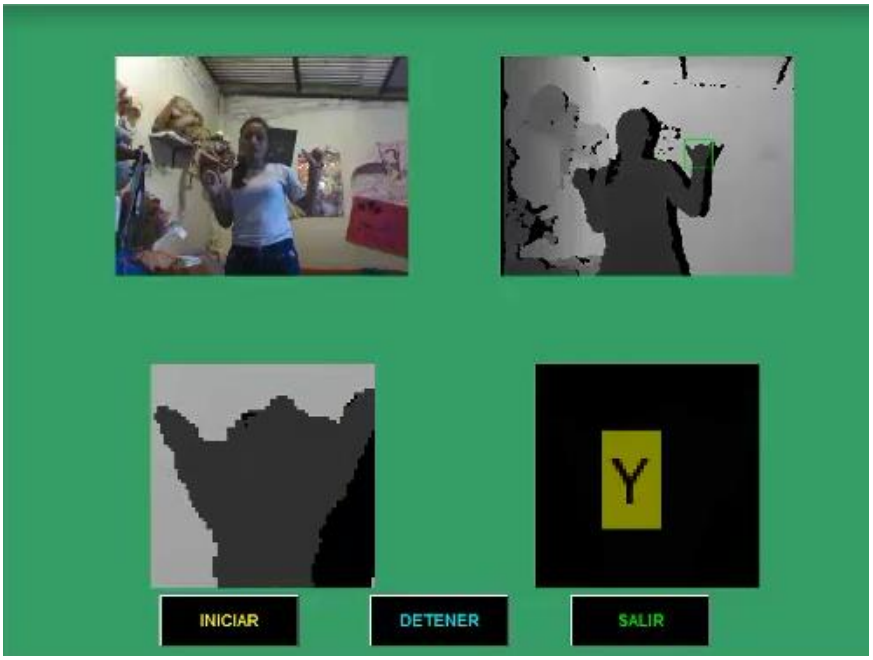


Figura 118. Traducción de la letra "Y".

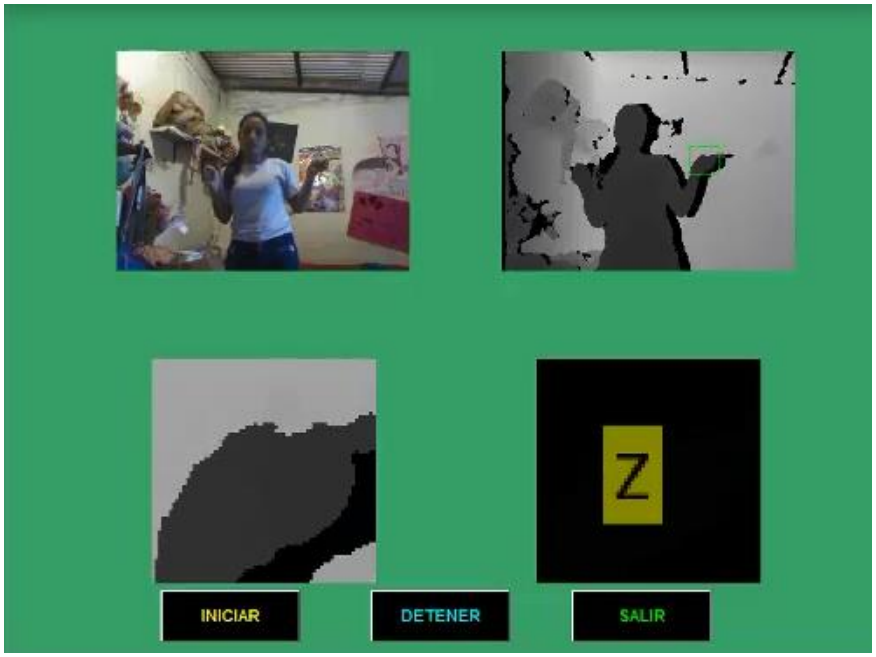


Figura 119. Traducción de la letra "Z".

Finalmente se muestra la traducción de cada una de las señas correspondientes a las letras del abecedario con las imágenes de 64x64 pixeles capturadas con la cámara de color de Kinect, es decir, la tercera prueba realizada.

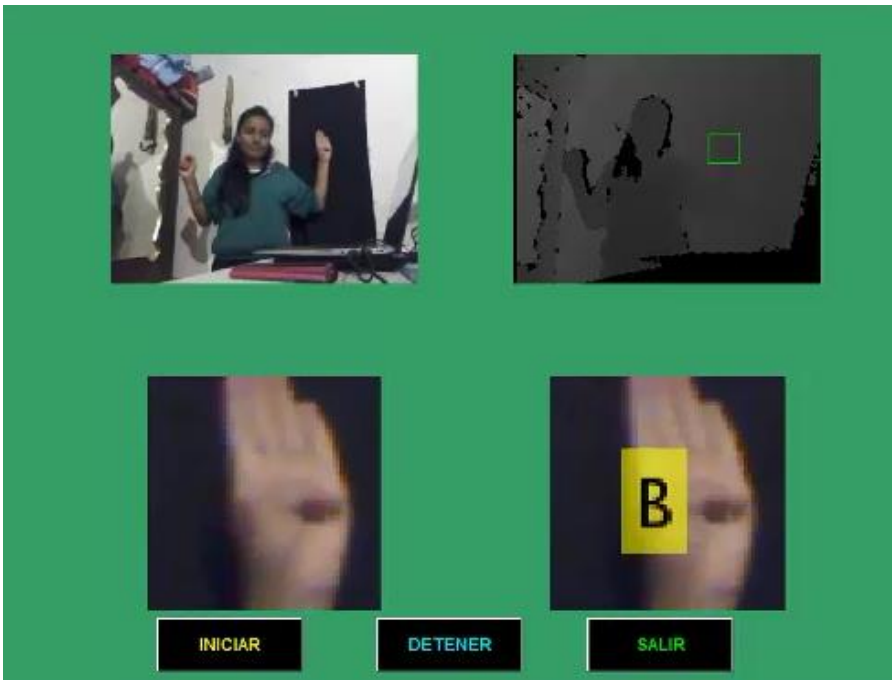


Figura 120. Traducción de la letra "B".

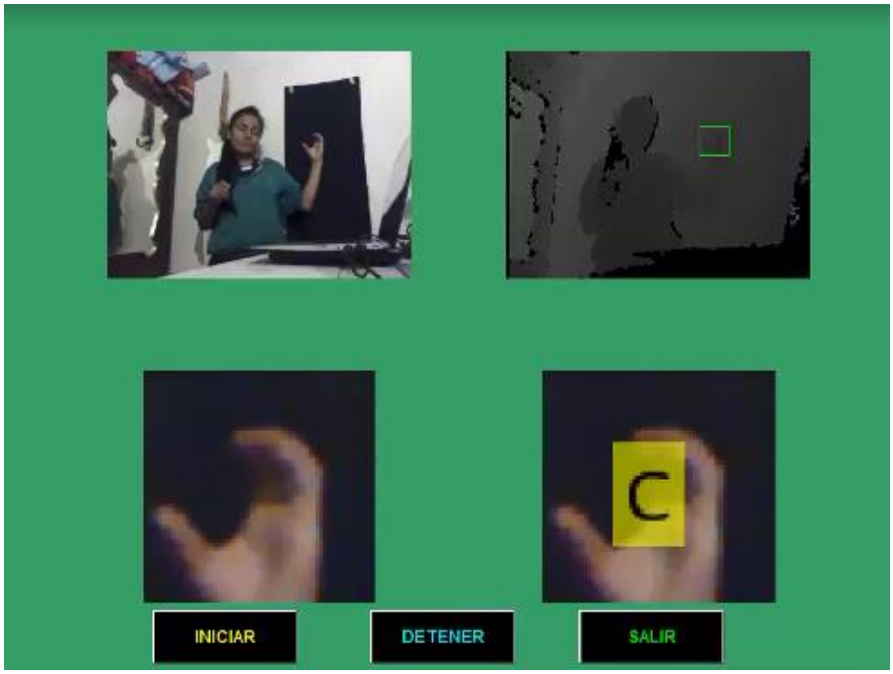


Figura 121. Traducción de la letra "C".

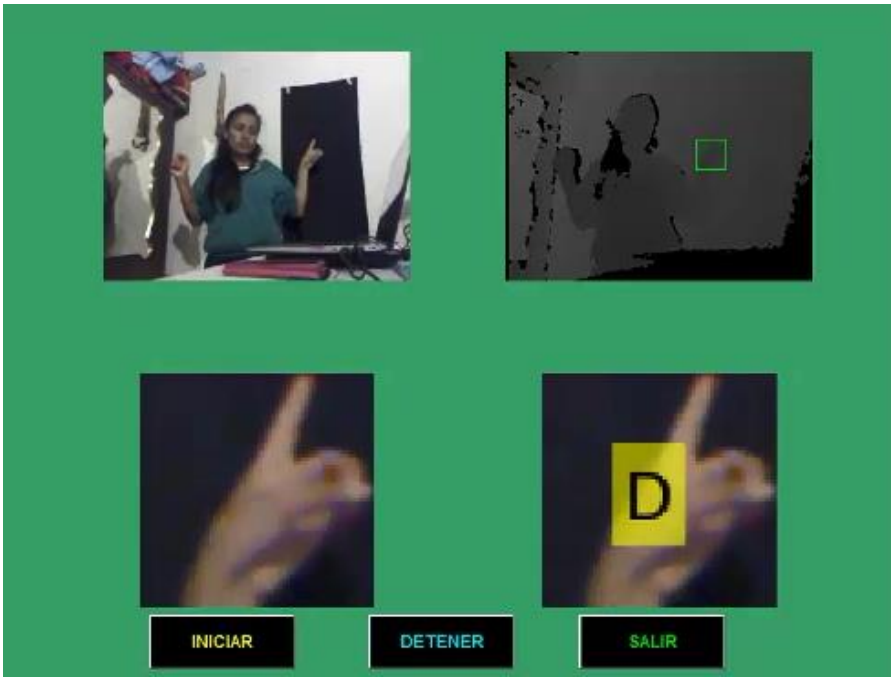


Figura 122. Traducción de la letra "D".

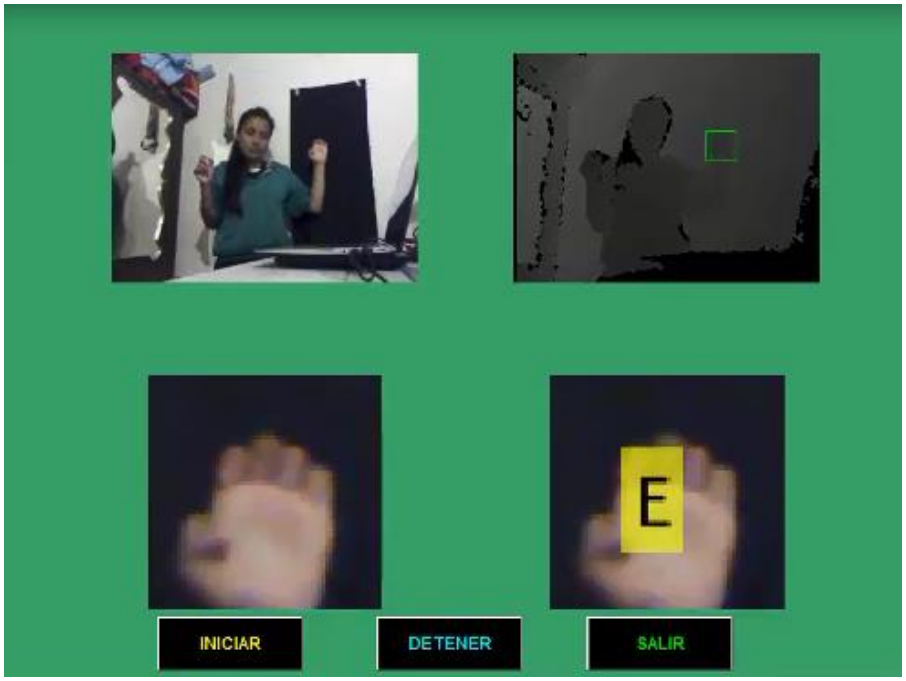


Figura 123. Traducción de la letra "E".

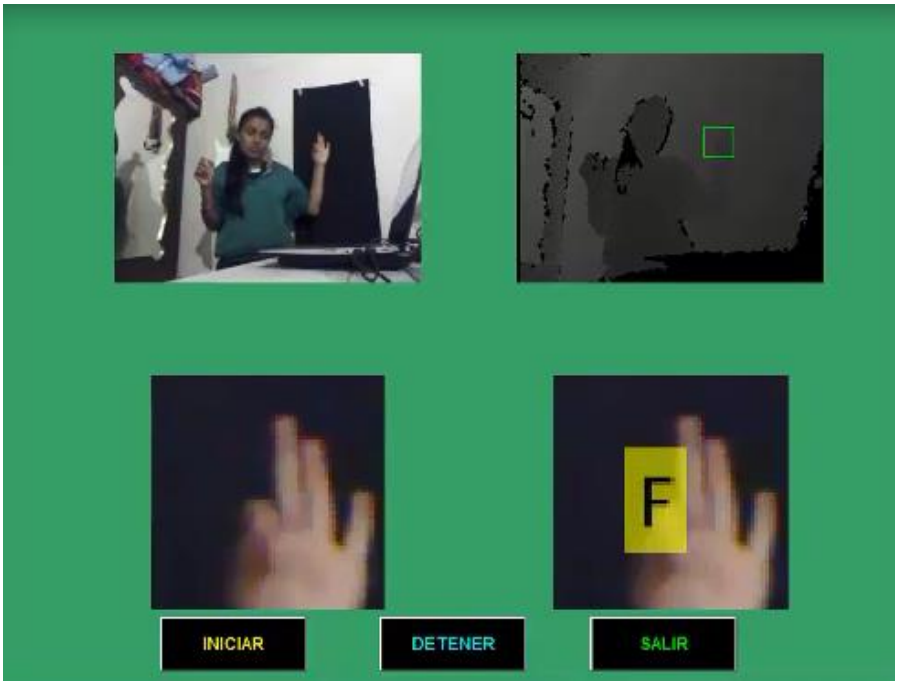


Figura 124. Traducción de la letra "F".

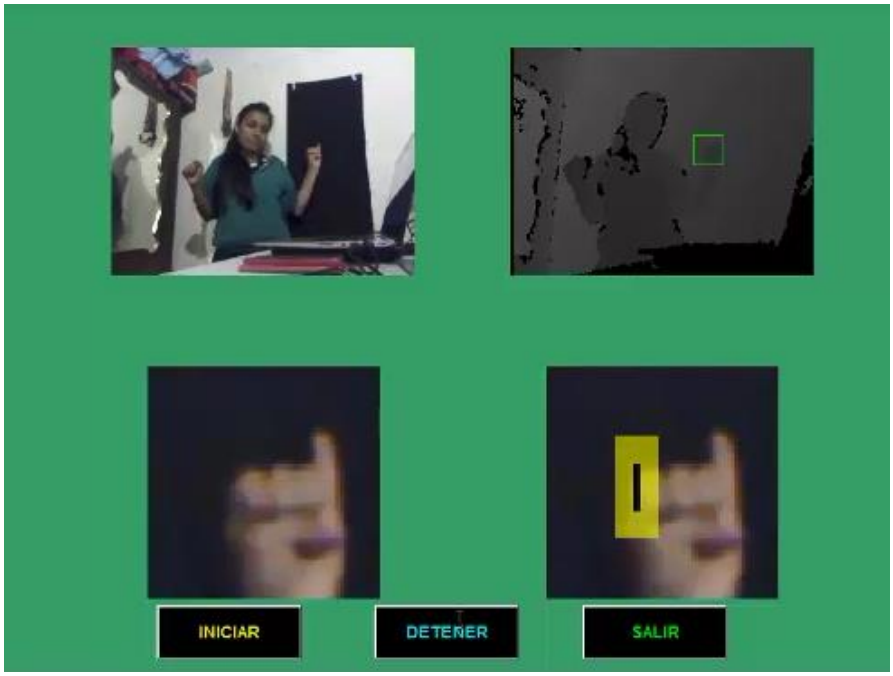


Figura 125. Traducción de la letra "I".

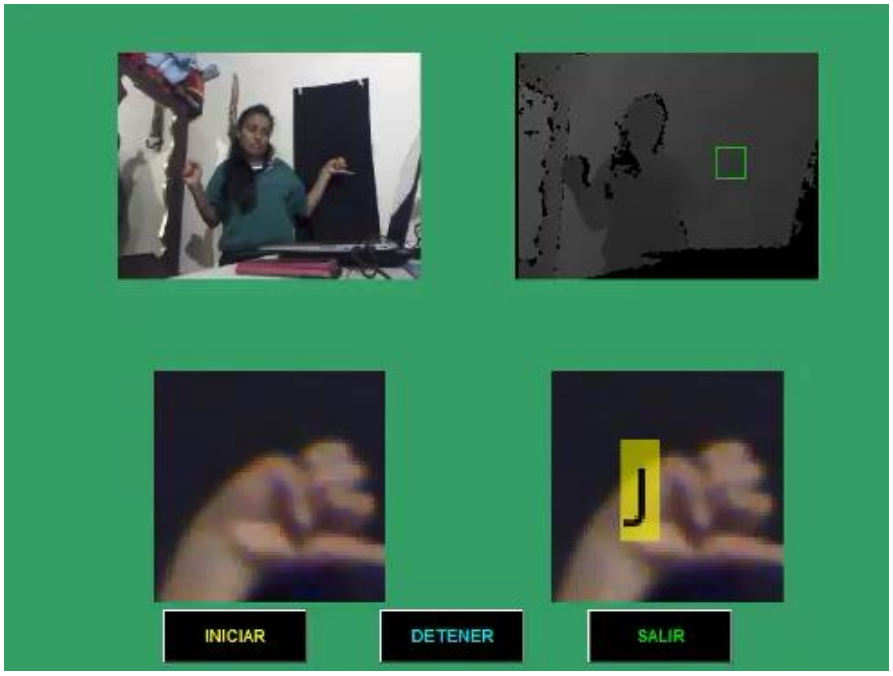


Figura 126. Traducción de la letra "J".

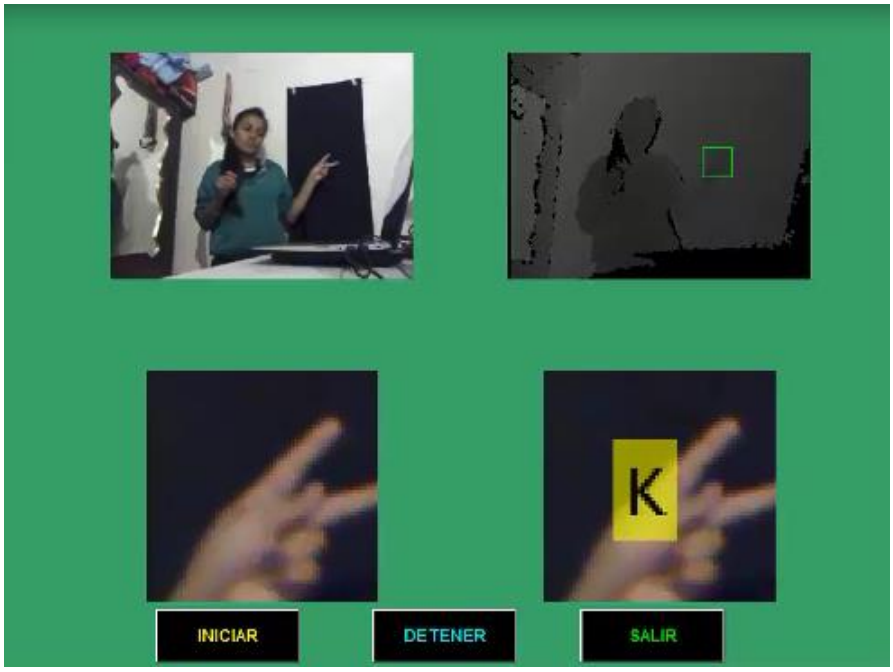


Figura 127. Traducción de la letra "K".

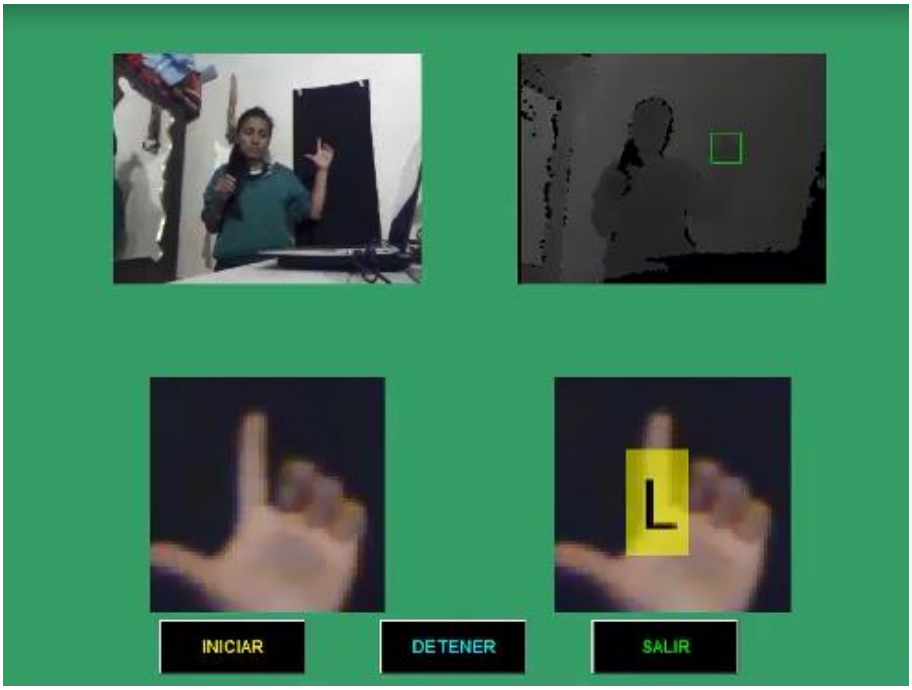


Figura 128. Traducción de la letra "L".

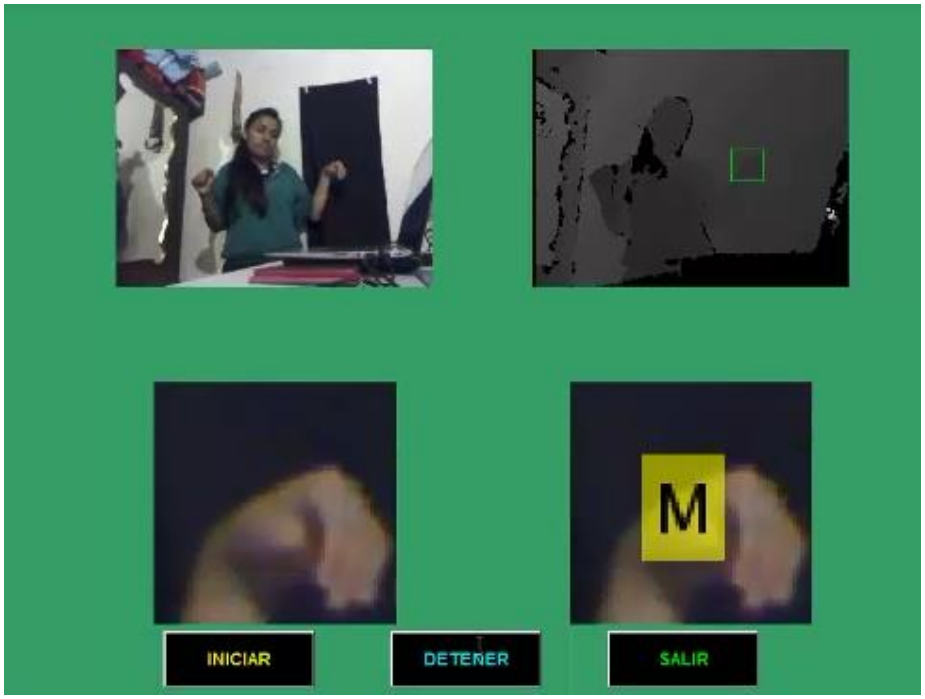


Figura 129. Traducción de la letra "M".

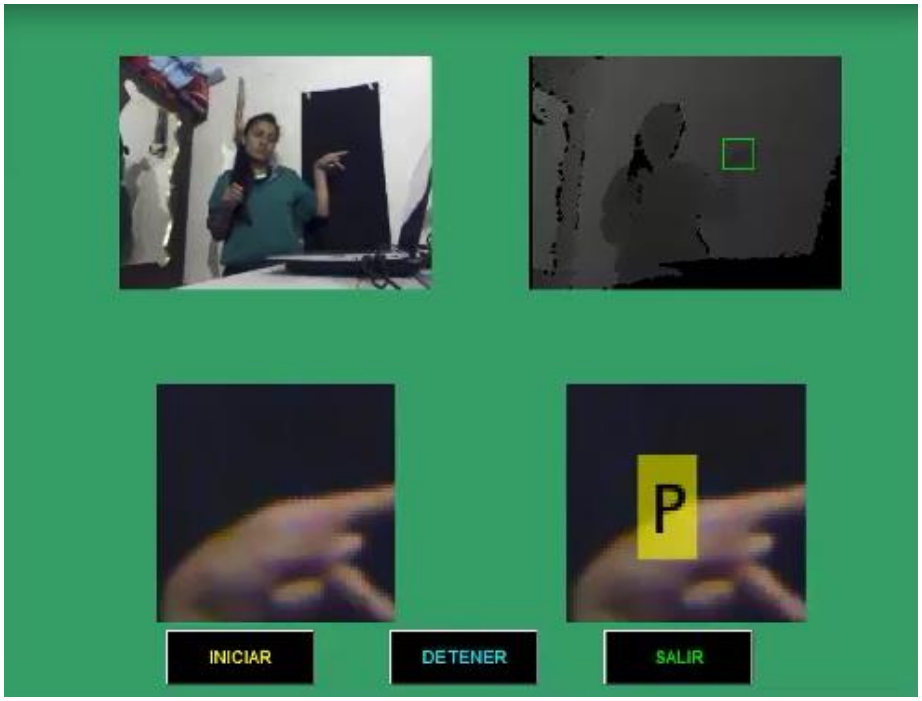


Figura 130. Traducción de la letra "P".

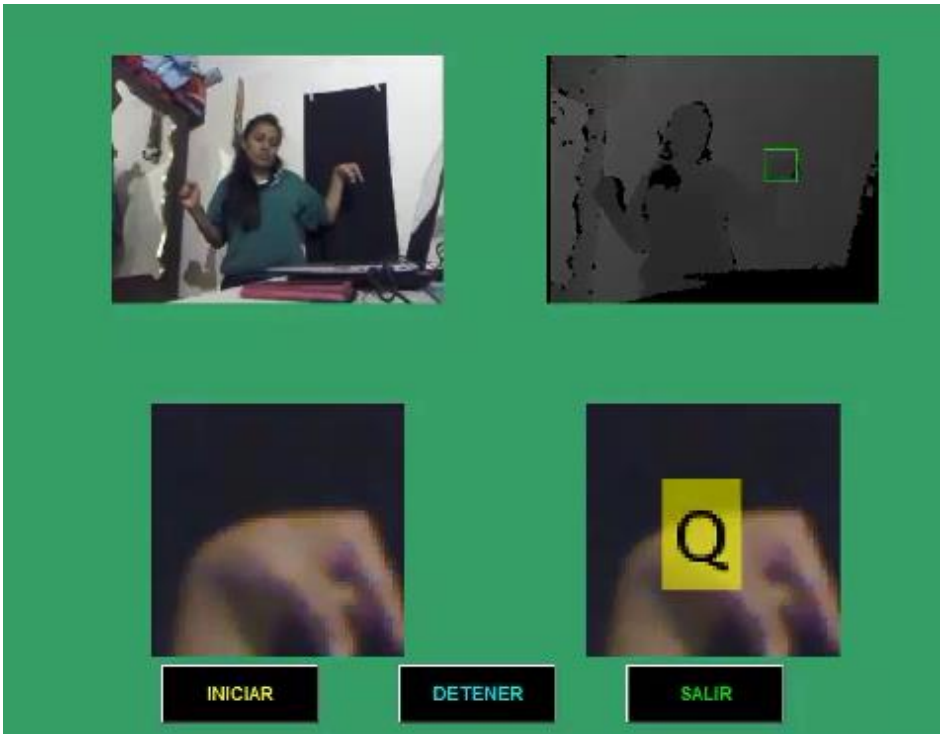


Figura 131. Traducción de la letra "Q".

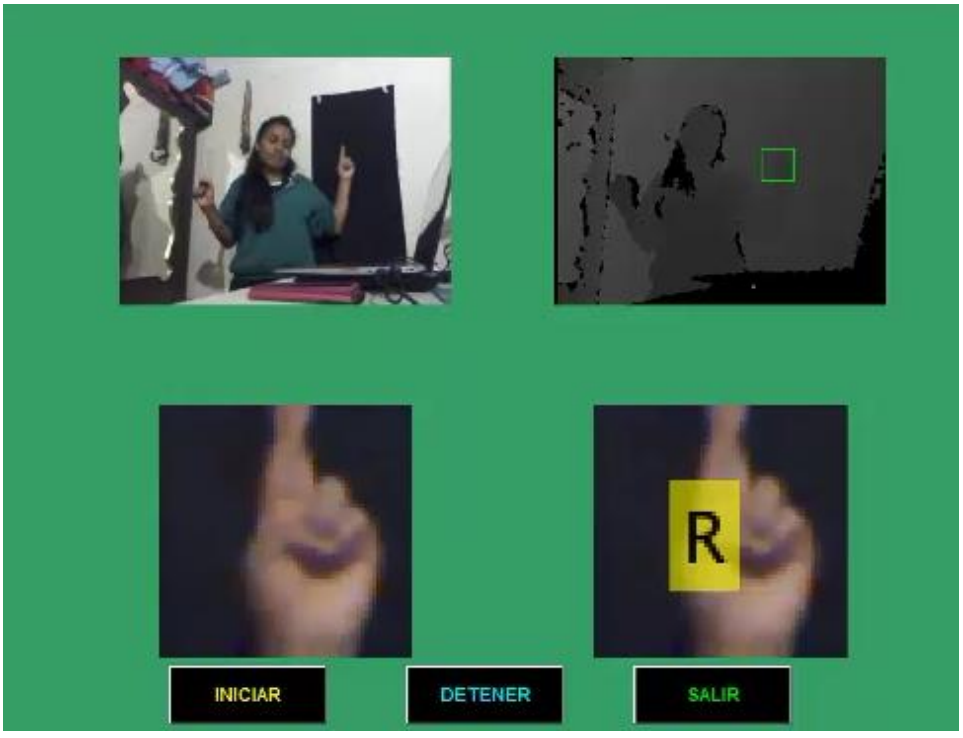


Figura 132. Traducción de la letra "R".

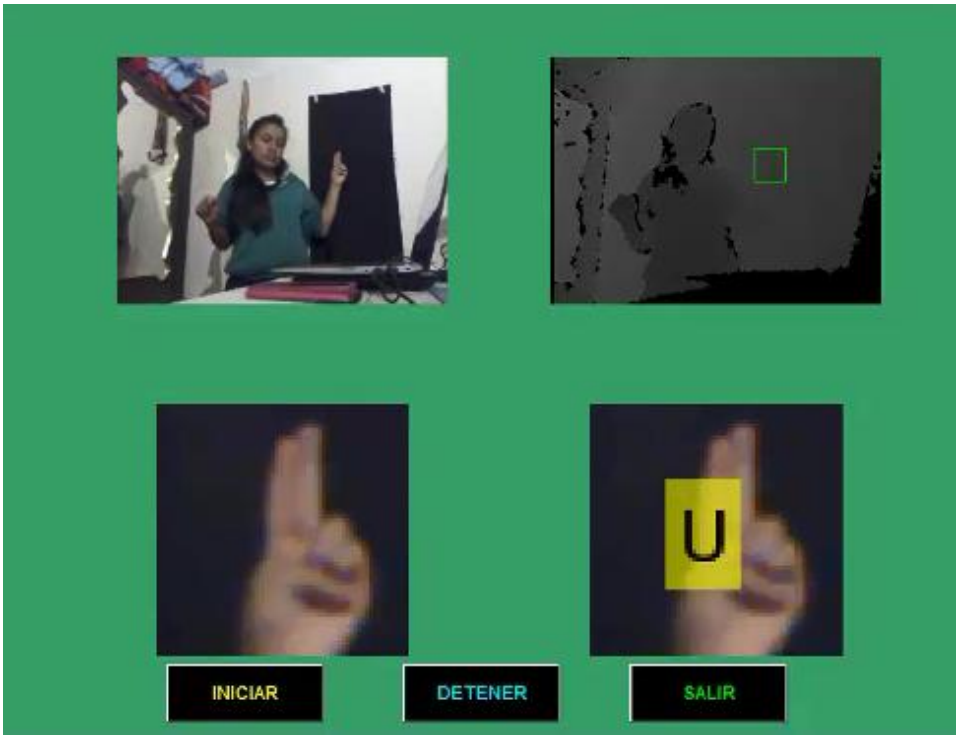


Figura 133. Traducción de la letra "U".

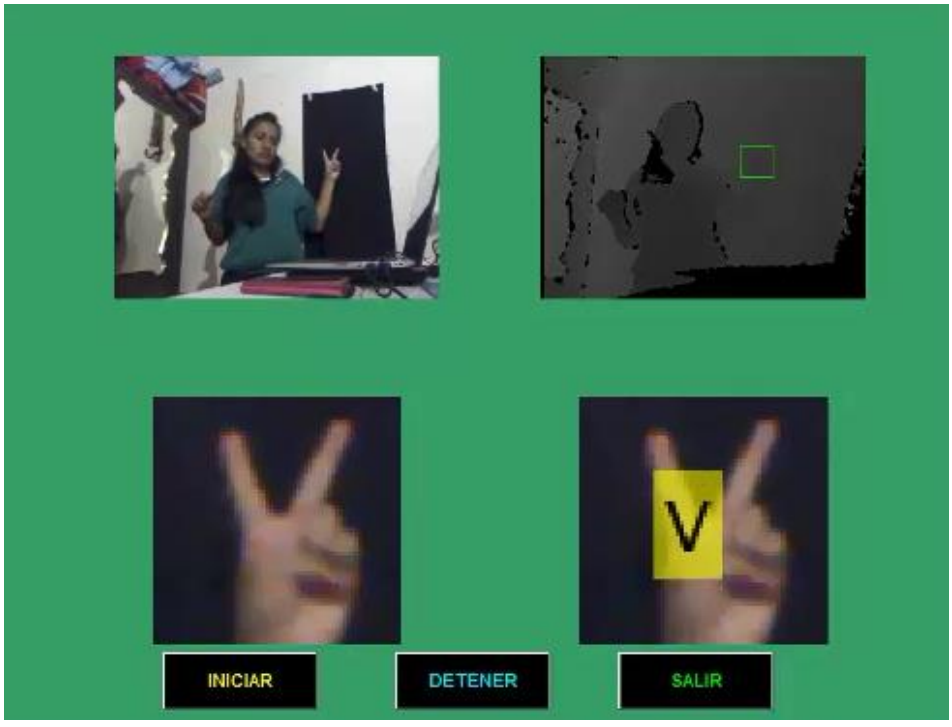


Figura 134. Traducción de la letra "V".

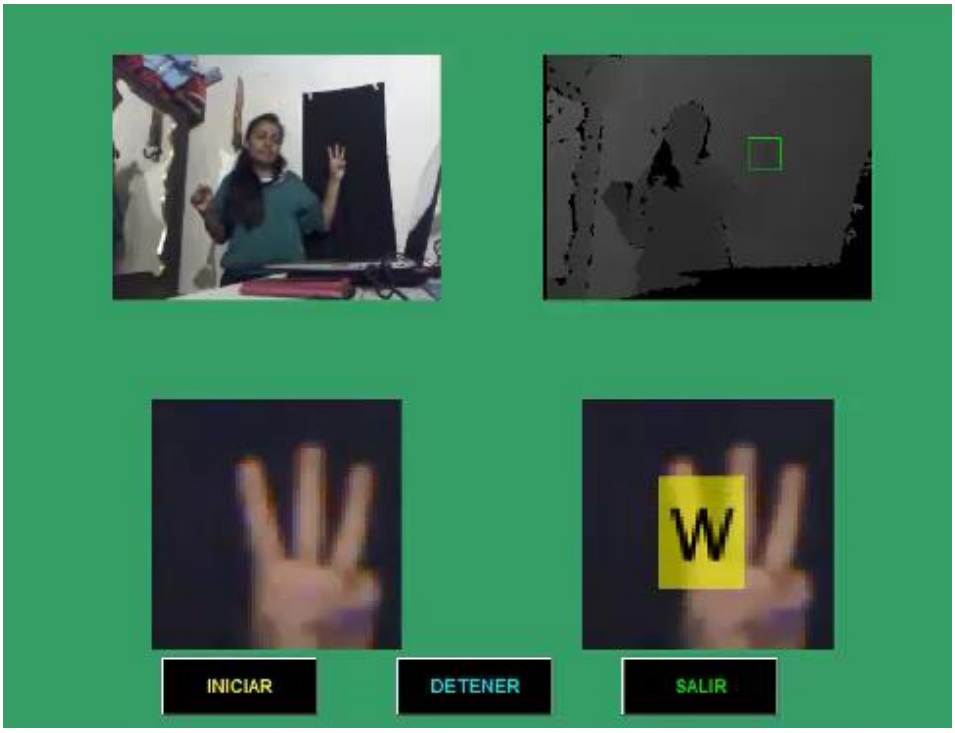


Figura 135. Traducción de la letra "W".

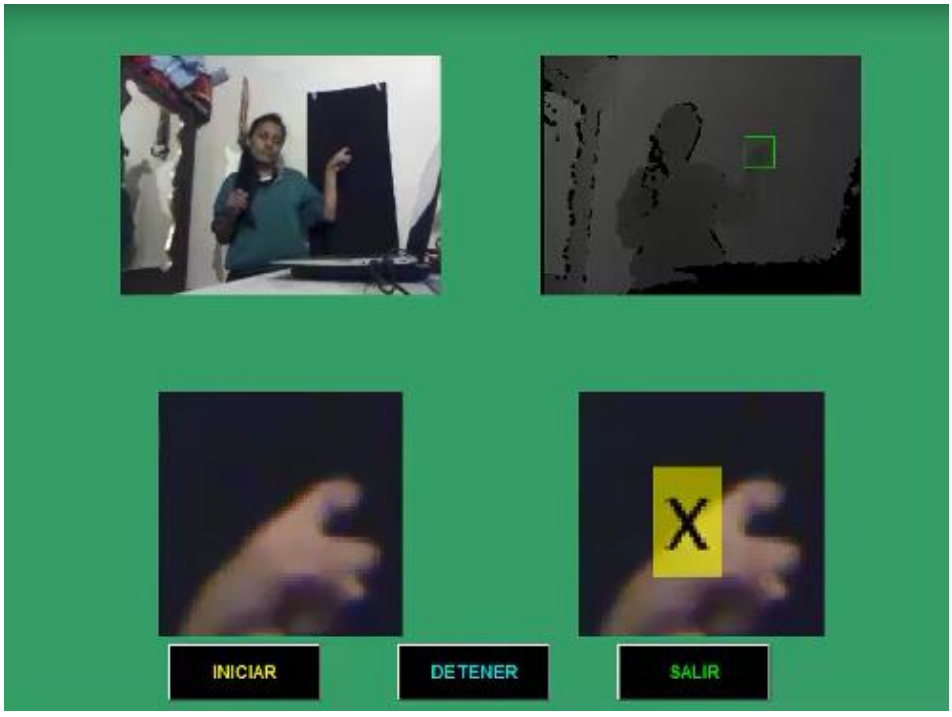


Figura 136. Traducción de la letra "X".

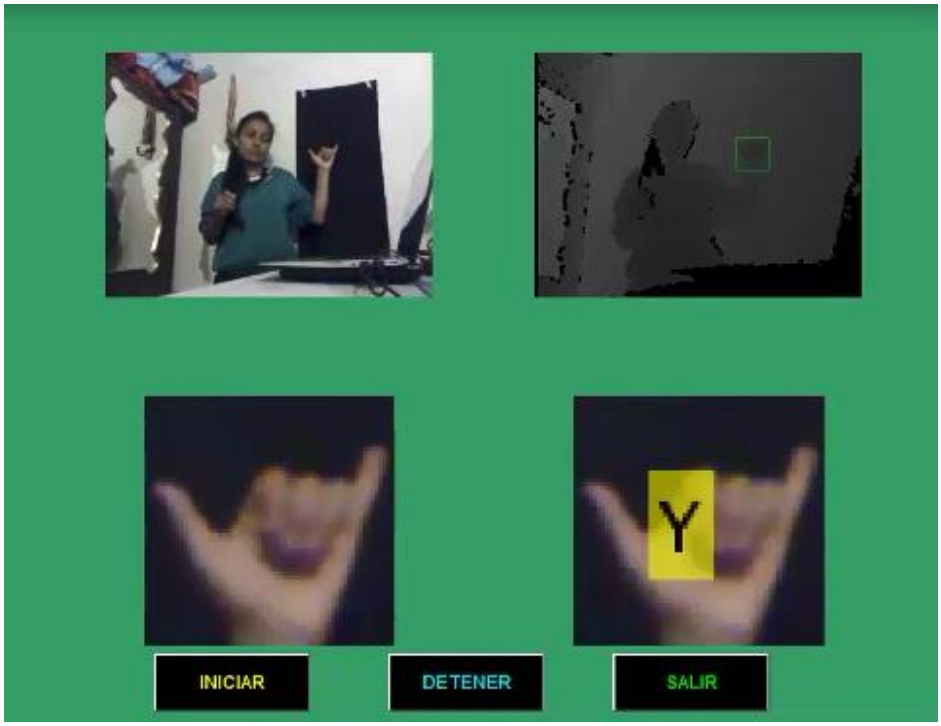


Figura 137. Traducción de la letra "Y".

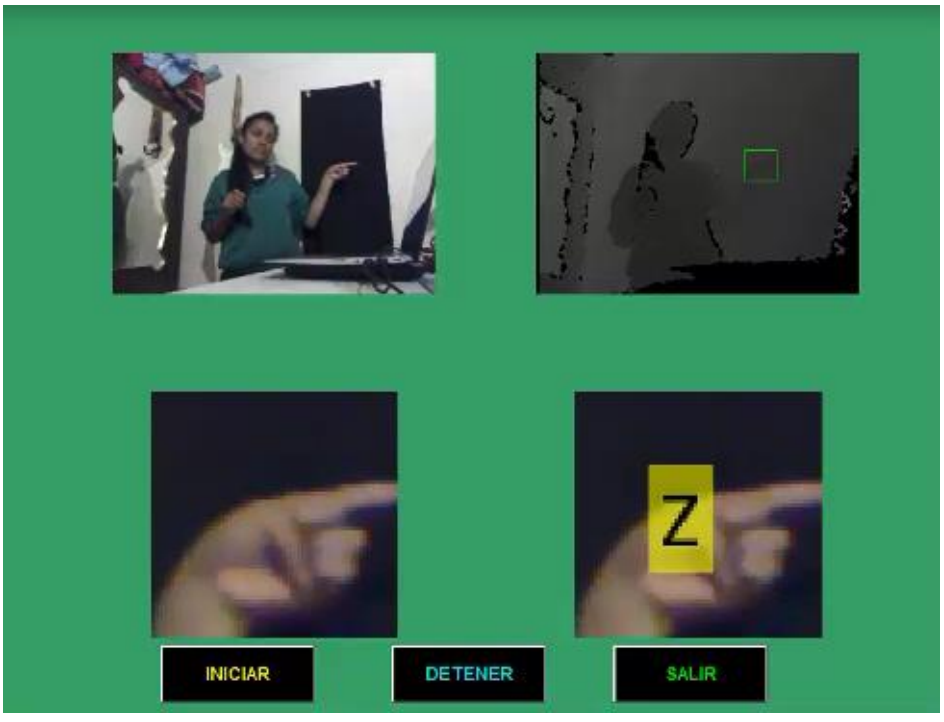


Figura 138. Traducción de la letra "Z".

7. DISCUSIÓN

El presente trabajo de investigación es de carácter investigativo, cuyo propósito es la traducción del lenguaje de señas e texto a través del dispositivo Kinect, con el fin de ayudar en cierto grado a las personas que padecen de algún tipo de discapacidad auditiva para que se puedan comunicar con las personas que no conocen o manejan este lenguaje.

- **DESARROLLO DEL PROTOTIPO**

Objetivo 1: Delimitar el alcance de la traducción de las señales dentro del lenguaje dactilológico.

Para el cumplimiento de este objetivo se delimitó la traducción únicamente de las letras del abecedario de este lenguaje, exceptuando letras dobles como la "ll", haciendo un total de 26 letras, de las cuales dos letras se las ejecuta con movimiento en la mano (letras "j" y "z"), es decir, son señas dinámicas, valiéndose de la premisa de que las personas que hacen uso de este lenguaje pueden modificar o crear señas para su mejor entendimiento, y dado que se hizo uso de técnicas de aprendizaje automático como las máquinas de soporte vectorial para el entrenamiento y aprendizaje, se creó una seña tanto para la letra "j" como para la "z", ya que estas técnicas no se aplican a la clasificación de clases que sean dinámicas.

Esto se hizo con el fin de no exceptuar ninguna letra para la traducción, cabe resaltar que esto se hizo únicamente porque es un prototipo y puede ser el punto de partida de futuros trabajos e investigaciones.

Según la bibliografía, para lograr traducir las señas dinámicas se puede hacer uso del Modelo oculto de Márkov (*del inglés Hidden Markov Model*) que es un modelo estadístico en donde se tiene como hipótesis que el proceso a modelar es un proceso de Márkov que posee parámetros desconocidos, y tiene como objetivo encontrar dichos parámetros, este proceso se puede llevar a cabo en aplicaciones de reconocimiento de patrones, de ahí su importancia; en futuros trabajos se puede aplicar este modelo para no tener que crear señas para las letras antes mencionadas sino más bien, implementar y traducir el abecedario tal y como lo establece la institución correspondiente en este caso el CONADIS.

Se supone también en este prototipo que la persona a realizar la seña es una persona diestra, por lo tanto, no funcionará si la persona es zurda, pero se puede entrenar la base

de datos capturando frames de la persona en cuestión y realizar todo el procesamiento de las imágenes (redimensionamiento y segmentación) para lograr traducir las señas ejecutadas por una persona zurda.

Objetivo 2: Extraer y procesar los datos provenientes del sensor Kinect.

El dispositivo Kinect nos ofrece la posibilidad de trabajar con las dos cámaras que tiene es su estructura, la de color y la de profundidad, se realizó pruebas con ambas cámaras; con la cámara de profundidad el procesamiento implicó menos coste computacional y un mínimo uso de recursos de la PC con la que se trabajó, ya que como se observa en las figuras 59 y 60 las imágenes capturadas a través de este sensor no están saturadas de color como lo sería si trabajáramos con las imágenes provenientes de la cámara de color, si bien es cierto se logra que los recursos computacionales sean lo suficientemente bajos posibles al trabajar con el sensor de profundidad, pero la gran desventaja que esto ocasiona es la confusión de las señas al momento de su traducción, ya que los gestos para ciertas letras se asemejan demasiado, letras como la "o", "s", "m", "n" no se logran identificar en su totalidad dado que no se distingue cada uno de los dedos empleados en su ejecución, al contrario, cuando se hizo pruebas con la cámara de color en las imágenes ya se distinguía claramente los dedos de las señas ejecutadas, sin embargo, para el procesamiento se tuvo que transformar cada una de ellas a imágenes binarias lo que ocasionó que el reconocimiento no sea tan satisfactorio, a esto se le añade las condiciones de iluminación ya que las imágenes tomadas de las hizo sin ninguna iluminaria adicional.

Para la segmentación del fondo de las imágenes almacenadas en la base de datos, se tuvo que establecer un valor de umbral que ayudó a extraer únicamente los datos que interesan, es decir lo que se observa en las tablas 5, 6 aquí se puede ver como se deja únicamente la posición donde está la mano, eliminando todo lo que no corresponde a ella, en la tabla 7 se muestra en cambio que la segmentación no fue necesaria ya que las imágenes fueron capturadas con un fondo específico, en este caso de color negro.

En la base de datos se tiene un total de 2600 imágenes, para la primera prueba realizada, pese a que representa un gran número, no se logra una traducción al 100% que sea estable, si se ampliara este número si se lograría tal objetivo, pero nuevamente se pone en juego el coste computacional al tener que procesar una gran cantidad de información; cuando se entrena el modelo se tiene que extraer características de cada una de las imágenes y

evaluar dicho modelo, lo que implica tiempo, el mismo que toma alrededor de 30 minutos con la información actual (2600 imágenes-frames) y con las características de la PC en la que se trabajó mencionadas en el apartado de materiales; como se observa el tiempo es relativamente alto, entonces se deduce que existe una relación directamente proporcional entre el número de imágenes dispuestas para el procesamiento y el tiempo que se tarda en procesar cada una de ellas

En la segunda prueba realizada la base de datos se reduce a la mitad, es decir, a 1300 imágenes obteniendo mejores resultados en esta prueba, pese a que el número de imágenes no es tan grande con en el caso anterior, esto se debe a que únicamente se procesaron imágenes con áreas que corresponden a la mano del ejecutante, lo que no ocurría en el caso anterior ya que también ahí se procesaron imágenes que tenían parte del antebrazo y la mano del usuario.

Por último en la tercera prueba realizada, la base de datos también tiene un total de 1300 imágenes, sin embargo el resultado no fue como se esperaba dado que las imágenes a pesar de distinguir perfectamente cada uno de los dedos, tuvieron que ser transformadas a imágenes binarias para su respectivo procesamiento con el fin de evitar un coste computacional demasiado alto.

Objetivo 3: Diseñar una interfaz que permita la visualización del lenguaje de texto.

Para la visualización de la traducción de cada una de las señas fue necesario el diseño de una GUI en Matlab desde la primera fase de la implementación del prototipo, ya que mediante esta, se adquirió cada una las muestras (fps) para su posterior procesamiento; en la primera fase la interfaz sólo muestra dos cuadros que corresponden al seguimiento de la mano derecha del usuario, y al recorte de la misma, respectivamente, tal y como se observa en la figura 62; en las dos siguientes pruebas ya se aumentó un cuadro más a la interfaz ya que se habilito la cámara de color de Kinect.

Para la última fase, es decir, para la interpretación de los datos, ya se añade un cuadro más a la GUI presentando en el mismo lo que corresponde a la traducción de cada una de las señas ejecutadas por el usuario, lo que ahí se muestra presenta un retardo de aproximadamente dos segundos hasta reconocer a que letra corresponde y entonces estabilizarse, dado que el método empleado para tal propósito es un método de aprendizaje que requiere dotar a las computadoras la capacidad de aprender mediante un

software (Matlab) y muestras previamente almacenadas, este retardo es aceptable ya que la PC debe reconocer constantemente de entre todo el número de frames, la que mayormente se aproxima a la seña realizada; pese a que este retardo es mínimo, existen señas que el programa no logra reconocer inmediatamente a pesar de que la seña está correctamente ejecutada ya que como se mencionaba en los resultados, existen señas ambiguas donde las imágenes capturadas presentan características similares.

Si se quisiera eliminar por completo este tipo de inconvenientes, se debería aumentar la base de datos incrementando el número de frames por segundo capturados para el respectivo entrenamiento y aprendizaje del modelo SVM, además el ambiente en el que se tomen todos estos datos debería ser un ambiente controlado, tanto en iluminación como en fondo.

8. CONCLUSIONES

- Se logró reconocer cada una de las letras del abecedario del lenguaje de señas ecuatoriano, a excepción de las mencionadas en cada una de las pruebas realizadas, obteniendo mejores resultados al trabajar con el sensor de profundidad de Kinect con imágenes de 64x 64 pixeles, en esta prueba se obtuvo un porcentaje de traducción equivalente al 84.6% a diferencia de las demás pruebas realizadas donde se obtuvo un porcentaje menor.
- Mediante el sensor de profundidad de Kinect, se logró capturar y procesar cada una de las imágenes necesarias para la base de datos, y con ellas trabajar en el proceso de entrenamiento y aprendizaje, dado que mediante este sensor, dichas imágenes requerían un menor coste computacional al redimensionarlas y segmentarlas, dado que no son imágenes de tres canales (RGB) como las que se toma del sensor correspondiente a la cámara de color de Kinect, entonces se acopló perfectamente en la PC en la que se trabajó.
- Al trabajar con la cámara de color de Kinect, dentro de las imágenes que forman parte de la base de datos se mostró claramente cada una de las características de las señas ejecutadas, sin embargo, al tener que procesarlas, dichas características se pierden en un gran porcentaje, ya que fueron transformadas a imágenes binarias con el fin de que el coste computacional sea el menor posible, logrando un porcentaje de traducción equivalente al 73.07%.
- En Matlab se diseñó una interfaz gráfica de usuario, de fácil manipulación para el usuario, para que se muestre cada una de las señas traducidas en tiempo real, dicha interfaz muestra la seña realizada por el usuario, el redimensionamiento y segmentación de la misma y finalmente la traducción que corresponde a la seña emitida originalmente.
- Mediante las máquinas de soporte vectorial, se logró entrenar y probar un modelo que permitió hallar un hiperplano ideal que logre separar cada una de las clases existentes en este trabajo, mostrando la importancia de las técnicas de aprendizaje automático para desarrollar proyectos de este tipo.

9. RECOMENDACIONES

- Para evitar que el traductor presente inconvenientes como la confusión de letras ambiguas se recomienda trabajar con la cámara de color del dispositivo en un ambiente totalmente controlado tanto en el fondo como en la iluminación ya que estas condiciones facilitaría su posterior procesamiento y traducción siempre y cuando se tenga en cuenta que el coste computacional y el tiempo que este hecho implicaría sería demasiado alto.

- Se puede ampliar la base de datos para una mejor interpretación de los mismos, y por lo tanto se logre una traducción que supere el porcentaje que se obtuvo en la diferentes pruebas realizadas en el presente trabajo, pero las características del ordenador deben ser robustas para que el procesamiento de los datos no presente un retardo demasiado grande.

- El ambiente en el que se capturen los frames, puede ser un ambiente controlado, es decir, se puede delimitar el área de trabajo y las condiciones de iluminación, colocando un fondo determinado, preferentemente negro, para facilitar la segmentación del mismo, aunque el resultado no presentaría cambios significativos ya que si se realiza un proceso de segmentación en este trabajo, dejando únicamente el área correspondiente a la mano de la seña emitida por el usuario, todo esto cuando se trabaja con el sensor de profundidad de Kinect.

- En futuros trabajos se puede hacer uso del esqueleto completo del usuario, esta función si se puede reconocer en Matlab, para que se logre traducir además de las letras del abecedario, palabras y frases completas, con lo que se podría implementar un servicio de chat para que se pueda interactuar en tiempo real con otras personas.

10. BIBLIOGRAFÍA

[1] A. E. N. Villacis, *Aplicación de DSP´s para la Transcripción de Lenguaje de Señas a Texto,* Ambato , 2014.

[2] CONADIS, «Datos estadísticos de Personas con Discapacidad,» Agosto 2015. [En línea]. Available: http://www.consejodiscapacidades.gob.ec/wp-content/uploads/downloads/2015/09/estadistica_conadis.pdf. [Último acceso: 25 Octubre 2015].

[3] C. Iberti, «Alfabeto Manual,» *Grupo Copesa,* 06 Septiembre 2012.

[4] FENASEC, de *Glosario Básico de lengua de Señas Ecuatoriano,* Quito, p. 22.

[5] A. R. d. P. Sordas, «Lenguaje de Sordos,» Colombia, 2010.

[6] M. J. V. Vilela, «La dactilología, ¿qué, cómo, cuándo...?,» España, 2005.

[7] P. S. S.A., «Aplicacion que traduce el lenguaje de señas a texto,» *Finanzas Personales,* 2011.

[8] E. Tecnología, «Google Gesture: la nueva app que traduce el lenguaje de señas en lenguaje oral,» *El Mercurio* , 21 Junio 2011.

[9] W. Gonzalez R. C., *Digital Image Processing,* Adisson Wesley, 2002.

[10] D. Malacara, *Procesamiento de Imágenes,* Mexico: Landucci, 1997.

[11] L. E. P. José Jaime Esqueda, Fundamentos de Procesamiento de Imágenes, Mexicali, Baja California: Departamento de Editorial Universitaria, 2005.

[12] J. R. Mejía, Procesamiento Digital de Imágenes, Mexico: Universidad Autónoma de San Luis Potosí, 2005.

[13] J. J. Grimaldos, Tratamiento digital de imágenes, Mexico : Centros de Profesorado de Almería, 2006.

[14] J. J. Esqueda, Fundamentos de Procesamiento de Imágenes, Baja California, 2002.

[15] G. L. J. G. Ricardo Argueta, «Desarrollo de software para extraccion de propiedades de cristales de azúcar utilizando visión artificial,» Universidad Don Bosco, San Salvador, 2008.

[16] G. M. Díez, «Segmentación de imágenes en color basada en el crecimiento de regiones,» e-Reding, Sevilla.

[17] C. A. Ordoñez, «Formatos de imagen digital,» *Revista Digital Universitaria,* vol. 5, nº 7, p. 10, 2005.

[18] V. A. Vanacloig, *El histograma de una imagen digital,* Valencia, 2010.

[19] J. M. Cabrera Canal, *Teleoperación de un robot móvil mediante percepción 3D*, Madrid: Universidad Carlos III de Madrid, 2011.

[20] C. Acevedo, «Consola Xbox y Kinect,» 17 Agosto 2012. [En línea]. Available: https://sites.google.com/site/xboxcomodidadparatucasa/historia-del-xbox-y-el-kinec. [Último acceso: 31 Octubre 2015].

[21] S. Peralta, *Interfaz de lenguaje natural usando Kinect*, Mexico DF: Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2012.

[22] M. d. C. C. K. D. Jorge Chuya, «Diseño e implementación de un sistema para el análisis dle movimiento humano usando sensoares Kinect,» Universidad Politécnica Salesiana, Cuenca, 2013.

[23] J. H. Luis Hernández, *Análisis y estudio de los códigos fuente SDK (Kit de Desarrollo de Software)*, Quito: Universidad Politécnica Salesiana, 2013.

[24] A. L. Ignacio San Román Lana, *Estimación de la posición de la cabeza en tiempo real con Kinect*, Pamplona: UPNA, 2012.

[25] M. Abrego, «Kinect for Windows Experience,» Octubre 2012. [En línea]. Available: https://malenyabrego.wordpress.com/2012/10/22/sabias-que-kinect-para-windows-puede-escuchar-y-puede-verte/. [Último acceso: 11 Noviembre 2015].

[26] M. D. España, «Reto SDK de Kinect: Detectar posturas con Skeletal tracking,» 09 Agosto 2011. [En línea]. Available: http://blogs.msdn.com/b/esmsdn/archive/2011/08/09/reto-sdk-de-kinect-detectar-poses-con-skeletal-tracking.aspx. [Último acceso: 11 Noviembre 2015].

[27] ngm, «La evolución de Kinect y la importancia real de Microsoft Research,» 07 Octubre 2013. [En línea]. Available: http://www.xatakawindows.com/xbox/la-evolucion-de-kinect-y-la-importancia-de-microsoft-research. [Último acceso: 08 Noviembre 2015].

[28] ASUSTeK Computer Inc., «Asus Xtion Pro Live-características,» 2014. [En línea]. Available: https://www.asus.com/es/Multimedia/Xtion_PRO_LIVE/specifications/.

[29] M. N. Vanessa Berazategui, «OpenNI: la alternativa open source para interaccionar con Kinect,» Uruguay , 2012.

[30] VisionLabs, «OpenNI: Framework para el desarrollo de aplicaciones con interacción natural,» 2011. [En línea]. Available: http://visionlabs.cl/blog/?p=171. [Último acceso: 05 Noviembre 2015].

[31] Microsoft, «Microsoft y el software de desarrollo de Kinect para Windows,» 2011. [En línea]. Available: http://www.microsoft.com/spain/prensa/noticia.aspx?infoid=/2011/06/n012-beta-kinect-windows.

[32] M. Research, «Programming with the Kinect for Windows SDK,» 2011. [En línea]. Available: http://research.microsoft.com/en-us/events/fs2011/jancke_kinect_programming.pdf. [Último acceso: 11 Noviembre 2015].

[33] J. O. Moll, «Detección de Personas en secuencias de vídeo en tiempo real,» Valencia, 2007.

[34] J. V. Rebaza, «Detección de bordes mediante el algoritmo de Canny,» Trujillo , 2010.

[35] J. L. Franciso Arranz, «Interacción persona-computador basada en el recococimiento visual de manos,» Universidad Complutense de Madrid, Madrid, 2012.

[36] I. V. M. Ruiz, «Representaciones de características para reconocimiento de objetos,» *Creative Commons Attribution-ShareAlike* , 14 Julio 2014.

[37] UNED, «Aprendizaje Automático,» UNED, España, 2014.

[38] F. S. Caparrini, «Aprendizaje Automático,» *Magazine Theme for PivotX,* 16 Julio 2015.

[39] V. R. Otero, «Reconocimiento de localizaciones mediante Máquinas de Soporte Vectorial,» Universidad Carlos III de Madrid, Madrid, 2008.

[40] EcuRed, *Aprendizaje Automático,* Cuba, 2012.

[41] G. Betancourt, «Las Máquinas de Soprte Vectorial (SVMs),» Scientia et Technica, Pereira, Colombia, 2005.

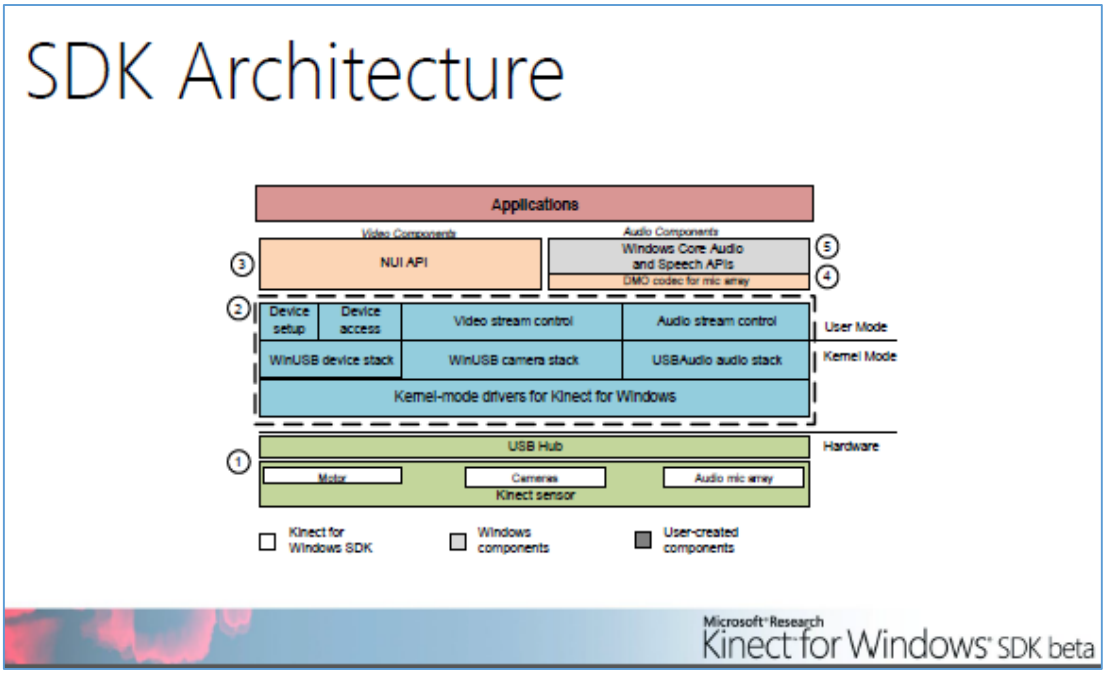
[42] G. Colmenares, «Máquina de Vectores de Soporte,» de *Inteligencia Artificial*, Venezuela, Universidad de los Andes, 2010, pp. 1-11.

[43] E. Gutiérrez, «Aplicación de las máquinas de soporte vectorial para el reconocimeinto de matriculas.,» Universidad Pontificia Comillas, Madrid, 2007.

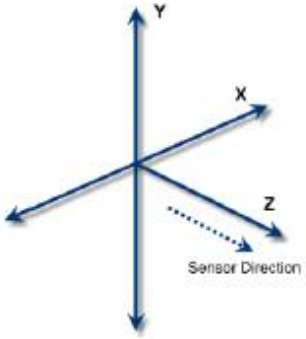
[44] Mathworks, «Mathworks,» 2015. [En línea]. Available: http://www.mathworks.com/. [Último acceso: 09 Noviembre 2015].

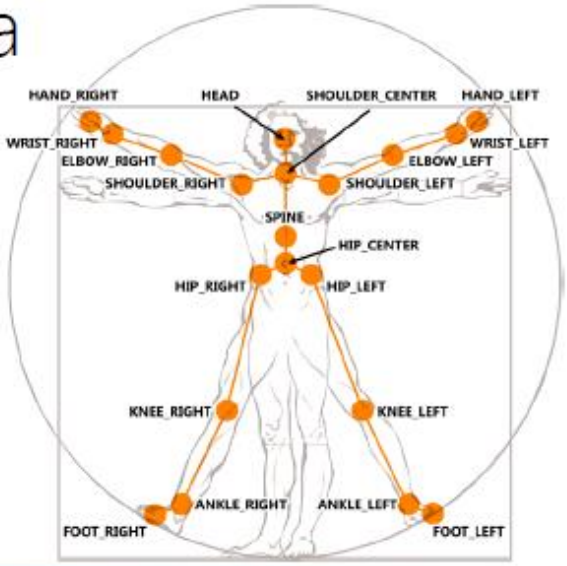
11. ANEXOS


3D DEPTH SENSORS
RGB CAMERA
XBOX 360
KINECT
MULTI-ARRAY MIC
MOTORIZED TILT
Microsoft Research
Kinect for Windows SDK beta


SDK Architecture
Applications
Video Components
Audio Components
NUI API
Windows Core Audio and Speech APIs
DMO codec for mic array
Device setup
Device access
Video stream control
Audio stream control
User Mode
WinUSB device stack
WinUSB camera stack
USBAudio audio stack
Kernel Mode
Kernel-mode drivers for Kinect for Windows
USB Hub
Hardware
Motor
Cameras
Audio mic array
Kinect sensor
Kinect for Windows SDK
Windows components
User-created components
Microsoft Research
Kinect for Windows SDK beta

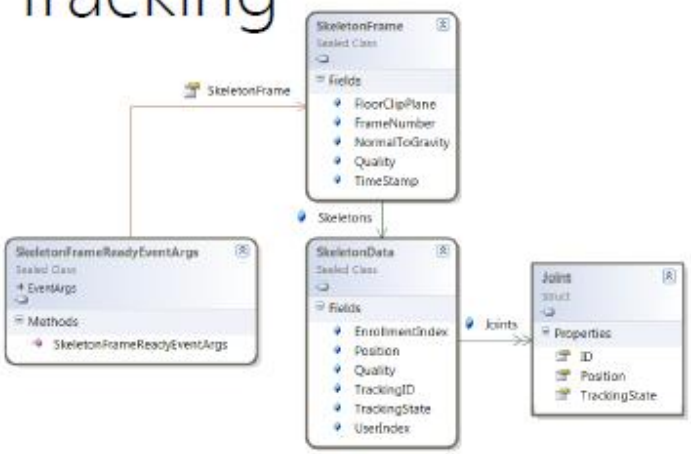
Skeleton Data





Microsoft Research Kinect for Windows SDK beta

Skeletal Tracking



Microsoft Research Kinect for Windows SDK beta