UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Diseño y desarrollo de un software para el aprendizaje del lenguaje de señas

TESIS

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Oscar Juárez Delgado

DIRECTOR DE TESIS

M. en A. Luis Yair Bautista Blanco

Ciudad Universitaria, Cd. Mx., 2017

ÍNDICE

CAPÍTULO 1. IDENTIFICACIÓN DEL PROBLEMA

Según la Organización Mundial de la Salud (OMS), discapacidad es un término general que abarca las deficiencias, las limitaciones de la actividad y las restricciones de la participación. Las deficiencias son problemas que afectan a una estructura o función corporal; las limitaciones de la actividad son dificultades para ejecutar acciones o tareas, y las restricciones de la participación son problemas para participar en situaciones vitales.

Por consiguiente, la discapacidad es un fenómeno complejo que refleja una interacción entre las características del organismo humano y las características de la sociedad en la que vive. [1]

Un elemento muy importante para el desarrollo y la interacción de una sociedad es el lenguaje, pues es el medio por el cual se transmite la información, sin embargo este elemento puede llegar a no ser bien desarrollado por personas con discapacidad auditiva y de comunicación.

De acuerdo a un artículo de la Comisión Nacional del Fomento Educativo, los niños con discapacidad auditiva enfrentan dificultad para adquirir el lenguaje. El lenguaje es una forma de conceptualizar el mundo, entenderlo y explicarlo; también, uno de los medios que nos permiten adquirir conocimientos e información acerca de nuestras experiencias y de los demás. A un niño con pérdida auditiva que no logra desarrollar un lenguaje le será muy difícil adquirir conocimientos y comprender los eventos a su alrededor. [2]

DISCAPACIDAD AUDITIVA Y DE COMUNICACIÓN EN MÉXICO.

Según datos del INEGI que datan de 2010 (Tabla 1.1), en México existen 694,464 personas con discapacidad auditiva. Por otro lado, en el país existen 477,104 personas con discapacidad del lenguaje en el país. [3]

Tabla1.Población con limitación en la actividad y su distribución porcentual según la causa y tipo de limitación en México.[3]

		Causa de limitación de la actividad (en porcentaje)					
Tipo de limitación	**Población con limitación en la actividad**	Nacimiento	Enfermedad	Accidente	Edad avanzada	Otra causa	No especificado
Caminar/moverse	3 347 849	6.74	42.88	18.72	25.45	4.72	1.49
Ver	1 561 466	10.95	42.06	7.12	25.94	12.22	1.71
Escuchar	694 464	13.44	24.98	9.05	44.51	6.55	1.47
Hablar/comunicarse	477 104	55.15	25.28	6.02	7.21	4.02	2.33
Atender al cuidado personal	315 598	16.27	40.95	14.7	22.96	2.61	2.51
Poner atención o aprender	252 942	45.92	21.83	5.15	16.1	7.54	3.46
Mental	490 472	52.81	23.45	6.68	4.49	5.61	6.96

Si se juntan ambas cifras puede notarse que hay un gran número de personas en el país a las cuales se les dificulta expresarse mediante el uso del lenguaje oral, es por eso que se ha implementado un tipo de comunicación basada en señas y con ello se ha desarrollado un lenguaje: el lenguaje de señas.

EL LENGUAJE DE SEÑAS

Esta forma de comunicación, al igual que el lenguaje oral, posee diferentes recursos y reglas para la construcción de palabras y oraciones. Además, se le conoce como una lengua visogestual porque para producir señas se requiere de las manos, del cuerpo, de los gestos y del espacio frente al cuerpo del señante, es decir, de la persona que no escucha y utiliza esta lengua. Mientras que las lenguas orales se organizan principalmente de manera simultánea y secuencial, las de las señas son espaciales, es decir, no solo aparecen estructuras en las cuales una seña va seguida de otra, sino que en algunos casos, las manos pueden tener un significado diferente, una mano puede hacer referencia a una persona y la otra a un objeto o lugar, por ejemplo, se usa una configuración de la mano para las palabras lunes, secadora y seguir (Fig.1), pero de acuerdo a variables como colocación, movimiento, orientación y expresión cobran diferentes significados.

Fig. 1. Seña correspondiente a las palabras “lunes”, “secadora” y “seguir”.

Los lenguajes de signos (o lenguajes de señas) son usados por las personas sordas en muchas partes del mundo. En México se usa un lenguaje de signos por casi todo el país. En español se le dan varios nombres, tales como “lenguaje de signos mexicano” y “lengua de señas mexicana”. Tanto en inglés como en español puede uno referirse a este lenguaje como “LSM”. En la actualidad no se conoce el número exacto de personas que utilizan el lenguaje de signos mexicanos (LSM), sin embargo, la 13 edición del "Ethnologue" publicado por SIL International, señala que en 1996 en México utilizaban este lenguaje aproximadamente 87 mil a 100 mil personas, como dato para dimensionar, la comunidad de sordos de México parece ser más grande que muchas familias completas de lenguas indígenas en el país. [4]

El LSM es distinto de los otros lenguajes de signos, como el lenguaje de signos norteamericano (ASL, que se usa en los E.U.A., Canadá y muchos otros países), los lenguajes de signos en España, y los usados en otros países de Latinoamérica. Tiene su propio vocabulario y su gramática, que son diferentes de la gramática española (aunque hay también un estilo de señas en México que arregla las palabras del LSM de acuerdo a los patrones de la gramática española). El LSM es realmente un lenguaje por su propio derecho, completamente capaz de expresar tan amplia gama de pensamientos y emociones como cualquier otra lengua. [4]

En México, pese a que la Lengua de Señas Mexicana es reconocida como patrimonio lingüístico y cultural del país, el acceso a cursos es escaso y casi siempre tienen un alto costo además de que son impartidos en centros muy específicos por lo que la distancia puede ser un impedimento para un gran número de personas.

Una alternativa es el uso de recursos electrónicos, cursos online y tutoriales; existen videos en internet que contienen animaciones o personas que fungen con el papel de instructor, por lo que se le deja al usuario la tarea de replicar movimientos, el acceso a este tipo de recursos es relativamente sencillo sin embargo como ya se mencionó el canal de comunicación es unilateral, además para hacer uso de esta opción debe tenerse algún dispositivo electrónico (computadora, celular Smartphone o tablet).

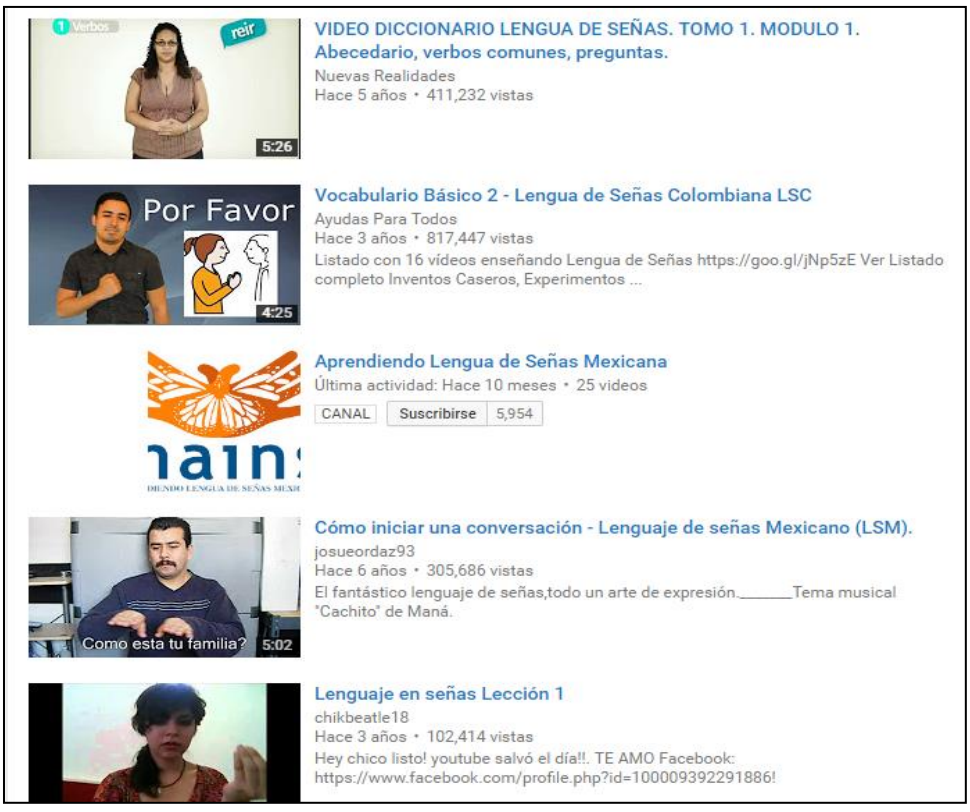

Fig. 2. En plataformas como YouTube existen videos tutoriales para aprender el lenguaje de señas. [5]

El Instituto Nacional de Estadística y Geografía (INEGI) afirma que hay 42.3 millones de mexicanos con una computadora y 37.6 millones con acceso a internet, esta última cifra incluye a los usuarios que tienen un Smartphone, teléfono celular o similares. Se estima que en México existen nueve millones de hogares con una computadora, lo que equivale al 30% de los hogares del país y un crecimiento de 6.9% en comparación con los datos del 2010. [6]

Es difícil empatar las cifras de personas que no pueden hacer uso del lenguaje oral y que posean alguna herramienta electrónica puesto que no hay un estudio previo especializado en este grupo de la población.

Recapitulando, se puede observar que hay un número importante de personas en el país con discapacidad auditiva y del lenguaje en México lo cual hace muy difícil la tarea de desarrollarse dentro de la sociedad, el medio por el cual se ha tratado de resolver dicho problema es el de adoptar el lenguaje de señas como sustituto al lenguaje oral, sin embargo la enseñanza del mismo recae principalmente en centros especializados de los cuales existen pocos en el país y por lo tanto se encuentran centralizados, esto da paso al problema de la accesibilidad de la enseñanza la cual se compone de cursos presenciales, algunos tienen un costo elevado, y de cursos en línea los cuales son mayoritariamente compuestos por video tutoriales.

Los cursos presenciales son el recurso más efectivo, pues se imparte por instructores con un amplio conocimiento del lenguaje de señas pero por supuesto tiene sus limitaciones, una de ellas es, como ya se mencionó, el acceso, pues son impartidos en sitios y horarios muy específicos dando como resultado problemas de traslado y de sincronización de horarios para los interesados, el número de estudiantes por clase es limitado al enfrentarse a problemas de espacio, mobiliario y uno de los mas importantes que es la pérdida de la atención especializada en cuanto mayor sea el número de estudiantes por profesor.

La otra alternativa es la de los cursos online (video tutoriales) los cuales necesitan de un medio electrónico y en la mayoría de casos de internet, estos resuelven el problema del desplazamiento y en su mayoría son recursos gratuitos, sin embargo son un medio de comunicación unidireccional y por lo tanto carecen de retroalimentación, elemento fundamental para el aprendizaje.

Estos son eslabones débiles en la enseñanza del lenguaje de señas, la cual es un pilar en el objetivo primordial que es lograr la comunicación entre una persona que depende del lenguaje de señas y una que se comunica mediante el lenguaje oral.

ESTADO DEL ARTE

En los últimos años se han registrado una gran cantidad de proyectos enfocados a facilitar dicha comunicación; el medio y las soluciones planteadas son muy variados, a continuación se mostrarán algunas de las más destacadas.

GUANTES TRADUCTORES

Estos, son sistemas que tratan de ser embebidos, toda la instrumentación se concentra en el guante y algunos hacen uso de dispositivos externos como celulares o computadoras para interactuar con el usuario, ha habido proyectos nacionales y extranjeros de este tipo.

Guante traductor desarrollado por el IPN

Este prototipo fue creado por el doctor Miguel Félix Mata y la licenciada Helena Luna García en el año 2015, consiste en un guante que detecta los movimientos de la mano realizados por el usuario y los asocia con letras del alfabeto internacional de 26 letras, las palabras formadas por el dispositivo son transmitidas a un dispositivo móvil vía inalámbrica, mismo que sintetiza y reproduce en voz. [7]

Fig. 3. Guante traductor y su aplicación móvil desarrollados por el IPN. [7]

Guante traductor desarrollado por la Universidad de Washington

El proyecto fue realizado por los estudiantes Navid Azodi y Thomas Pryor en el año 2016, el cual fue merecedor del premio Lemelson del MIT, el nombre del proyecto es “SignAloud” y se trata de unos guantes que traducen el lenguaje de señas americano en diálogo y texto.

Cada uno de los guantes tiene sensores que registran el movimiento y los gestos, luego transmiten la información de forma inalámbrica a una computadora central, la computadora recibe los datos y los analiza para encontrar coincidencias con una palabra o frase para después transmitirla por un altavoz. La diferencia con el anterior dispositivo es que éste se encuentra mejor instrumentado y tiene un procesamiento más robusto y completo pues no solo se limita a traducir las señas del alfabeto sino que puede traducir gestos más complejos. [8]

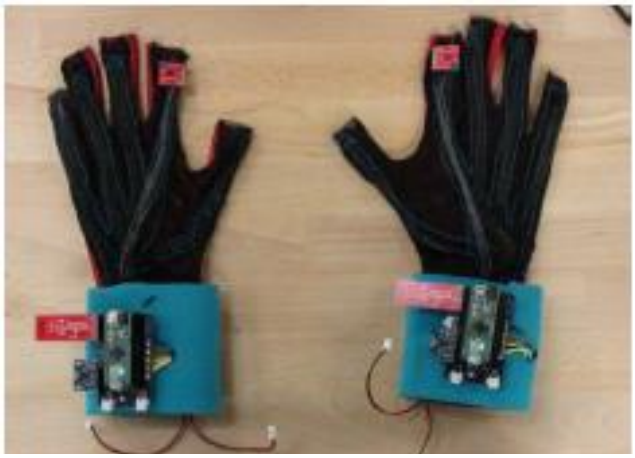

Fig. 4. Guantes traductores “SignAloud” de la Universidad de Washington. [8]

TRADUCTORES BASADOS EN VIDEO

Este tipo de proyectos sustituye el gran número de sensores que usan los proyectos basados en guantes y los sustituyen por un único tipo de sensor: una cámara de video, el número de elementos es considerablemente menor lo cual da como resultado un sistema más cómodo y menos invasivo sin embargo el procesamiento y la forma de obtener todas las variables se convierte en una tarea más compleja pues requiere de una base teórica muy robusta. Dicha tarea depende de técnicas de procesamiento de imágenes mismas que la mayoría de las veces hacen uso de complejos algoritmos, como redes neuronales y reconocimiento de patrones, para obtener de las imágenes la información necesaria para conocer las variables de interés.

Tableta electrónica traductora

Este dispositivo fue desarrollado por la empresa MotionSavvy, una startup con sede en California; el desarrollo estuvo a cargo de Ryan Hait Campell y Alexandr Opalka.

Para la detección de los gestos se utilizó el Leap Motion, el cual es un pequeño dispositivo con entrada USB (Bus Serial Universal) que hace uso de dos cámaras infrarrojas monocromáticas y tres LEDs (Diodo Emisor de Luz) infrarrojos, la información que aportan las cámaras es procesada utilizando "matemáticas complejas" mismas que no han sido reveladas por la compañía, con lo cual pueden procesarse hasta 300 signos, además de un altavoz para reproducir el mensaje al lenguaje oral. El precio de lanzamiento de este dispositivo fue de 800 dólares más una suscripción mensual de 200 dólares para acceder al Constructor Sign que es un software para complementar su uso. [9]

Fig. 5. Dispositivo *tablet* traductora de MotionSavvy . [9]

Fig. 6. Sensor *Leap Motion.* [10]

Traductor MICROSOFT

La iniciativa es conjunta entre el área de investigación de Microsoft Asia y el Instituto de Tecnología Computacional de la Academia China de Ciencias, la idea es aprovechar la capacidad del Kinect (que es un controlador de juego libre y entretenimiento, desarrollado por Microsoft para su consola Xbox 360, el cual permite que los usuarios controlen e interactúen con la consola mediante una interfaz natural de usuario que reconoce gestos, comandos de voz, y objetos e imágenes), de leer el movimiento humano y digitalizarlo para que sirva para reconocer y traducir el lenguaje de señas. El sensor de movimiento sigue al circuito que dibujan las manos y el cuerpo, interpreta y traduce. [11]

Fig. 7. Interfaz del traductor basado en el uso del sensor *Kinect.* [11]

TRADUCTOR GESTUAL ONLINE

A diferencia de los demás traductores, éste realiza el tránsito de comunicación de manera contraria pues en vez de traducir en voz para la persona que se comunica de manera oral, traduce un mensaje a señas para que pueda interpretarlo la persona que se comunica usando el lenguaje de gestos; esto se hace mediante un *avatar* llamada Iris la cual ejecuta los movimientos de la frase en cuestión. Este traductor se encuentra online y es gratuito, aquí su liga http://hetah.net/traductor. [12]

Fig. 8. Apariencia del traductor online, se selecciona una letra y el avatar realiza la seña correspondiente. [12]

En resumen, existe un gran número de alternativas de solución, algunas con mayor alcance y mejor desempeño que otras, sin embargo todas ellas tienen claro el objetivo que se mencionó anteriormente: lograr la comunicación entre una persona que depende del lenguaje de señas y una que se comunica por lenguaje oral y muchas de ellas si aún no lo cumplen de manera fluida muy pronto lo harán, sin embargo se destaca que dicho objetivo tiene dos posibles vías de solución: introducir al señante al mundo del lenguaje oral basándose en la traducción, como la gran mayoría de los desarrollos anteriormente mencionados lo hacen, o hacerlo de forma contraria, que la persona que hace uso del lenguaje oral pueda aprender el lenguaje de señas para lograr una comunicación entre dos señantes.

No se puede señalar cuál de los dos enfoques es mejor, pero lo que sí se puede establecer es que para llevar a cabo cualquiera de las dos es indispensable la presencia del lenguaje de señas, haciendo hincapié en la importancia de la enseñanza del mismo como piedra angular.

CAPÍTULO 2. REQUERIMIENTOS Y ESPECIFICACIONES DE DISEÑO

Identificado el problema de la comunicación para las personas con discapacidad auditiva en México y tomando en consideración la información presentada en el anterior capítulo, se realizará un software que permita aprender el lenguaje de señas a los usuarios, brindando la información necesaria para poder realizar la seña y evaluando el desempeño de éste. Para la primera aproximación de este proyecto se realizará una prueba de concepto en donde se involucren los primeros elementos del lenguaje, aquellos necesarios para una comunicación básica.

El sector al que va dirigido es a las personas que se quieren iniciar en el lenguaje de señas mexicano que puedan manejar una computadora.

En el capítulo anterior se mencionaron las limitaciones de las alternativas de solución ya existentes, tanto de los nuevos proyectos que se están desarrollando como de los cursos ya existentes, sin embargo es en este capítulo en el que se puntualizarán dichas características y se llegará a un listado de cualidades que el proyecto a desarrollar debería tener.

Se mencionó que uno de los principales problemas de los cursos presenciales es el traslado y la disponibilidad de horario, esto lleva a que **el software podrá ejecutarse a cualquier hora del día y en cualquier lugar, además podrá ejecutarse en el mayor número de dispositivos posible**.

Esta característica podría considerarse como una no muy reveladora ni innovadora puesto que los cursos online cumplen con dicho requerimiento, sin embargo también se mencionó una carencia de estos, lo cual nos lleva al siguiente requerimiento:

El software podrá evaluar el desempeño del usuario, pues como también se indicó, la retroalimentación (tanto positiva como negativa) constituye uno de los elementos más importantes para el aprendizaje de cualquier teoría.

Como ya se mencionó la enseñanza del lenguaje de señas podría considerarse como el objetivo primordial, y para alcanzarlo **el software será amigable con las personas que se quieran introducir en el lenguaje de señas o aquellos que se encuentren en etapas tempranas del aprendizaje**, permitiendo que las personas aprendan y practiquen.

Se ha señalado que el lenguaje de señas es complejo y depende de no solo la posición de los dedos de la mano, sino que también la posición de la mano con respecto al cuerpo del señante puede significar palabras o ideas muy diferentes, es por eso que es importante **que el vocabulario no se quede solamente en la interpretación del alfabeto sino que podría tener otros elementos básicos del lenguaje** como días de la semana, meses del año, etc.

Otro aspecto aunado a la gran cantidad de sensores es el tamaño del dispositivo, el cual puede llegar a considerarse como aparatoso aspecto que puede llegar a ser incómodo para el usuario, para evitar esta condición **el sistema será simple**.

Definidos los elementos básicos del proyecto, el siguiente paso es traducirlos en requerimientos.

Se abordarán en el mismo orden en el que se fueron obteniendo:

- **El software podrá ejecutarse a cualquier hora del día y en cualquier lugar, además podrá ejecutarse en el mayor número de dispositivos posible**.
  Ya que se ha establecido que se trata de un software, queda igualmente fijo que es necesario de un dispositivo electrónico en el cual se ejecute, para poder cumplir con la segunda parte del enunciado se necesita saber cuál es el sistema operativo con el mayor número de usuarios, para los dos más importantes dispositivos (computadora y celular/tableta electrónica), los sistemas operativos que cumplen con dicha condición son:

Computadora ---------- Windows
Celular/tableta electrónica --------- Android

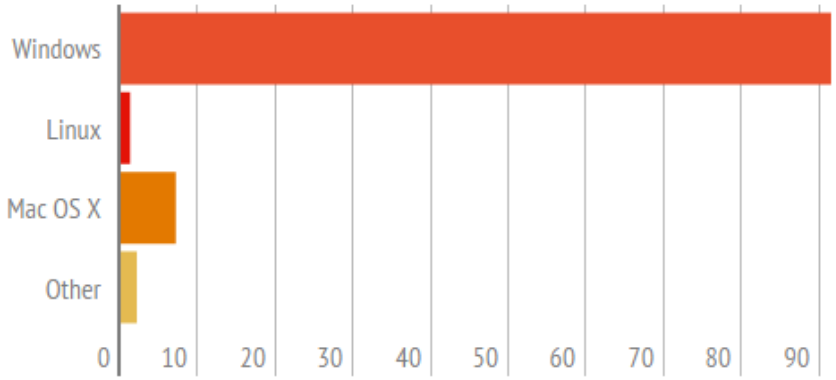

Fig. 9. Porcentaje de usuarios de los sistemas operativos para computadoras más populares. [13]

PARTICIPACIÓN DE MERCADO

OS	2013	2017
	75.3%	68.3%
iOS	16.9%	17.9%
	3.9%	10.2%
	2.7%	1.7%
Otros	1.2%	1.9%
TOTAL	100%	100%

Fig. 10. Porcentaje de usuarios de los sistemas operativos para dispositivos móviles más populares. [14]

Ahora se abordará la primera parte del enunciado, pero éste tiene una estrecha relación con el siguiente requerimiento:

- **El software podrá evaluar el desempeño del usuario.**

Para poder lograr que el software pueda evaluar correctamente no importando grandes variaciones del entorno se tienen dos posibles soluciones, dependiendo del modo de funcionamiento:

1.- Si la forma de obtener los parámetros a evaluar no depende de las condiciones del entorno entonces se cumple de manera indirecta el primer requerimiento, y no habrá que tomar mayores consideraciones.

2.- Si la forma de obtener los parámetros a evaluar sí depende de las condiciones del entorno entonces habrá que considerar qué tanto impacto tendrán las variaciones del mismo en su funcionamiento y de ser posible lograr un sistema que se adapte a dichas variaciones.

- **El software será amigable con las personas que se quieran introducir en el lenguaje de señas o aquellos que se encuentren en etapas tempranas del aprendizaje**

  Un elemento cualitativo como “amigable” es difícil de traducir a elementos cuantitativos, sin embargo para ello se proponen parámetros como [15]:

  Simple de instalar

  La instalación es el primer punto de contacto de los usuarios, ¿qué mejor que sea un proceso fácil? No importa si se trata de un sistema operativo o una aplicación de usuario de un solo cliente, la instalación debe ser sencilla y bien documentada.

  Fácil de actualizar

  Si las actualizaciones son complejas, lo más probable es que los usuarios se salten este proceso. A menudo esto puede dejar atrás un rastro de malos resultados, como muchas actualizaciones de parches de seguridad, pérdidas de memoria y otros problemas. Las actualizaciones deben ser lo suficientemente simples como para que los usuarios sigan beneficiándose de la ardua labor de los creadores del software. Cuando los usuarios no actualizan el software se vuelve menos confiable y seguro.

  Intuitivo

  El software es tan bueno como su interfaz gráfica de usuario. Si la interfaz gráfica no está bien pensada o ejecutada, las personas tendrán problemas para utilizar el producto. Una interfaz gráfica bien diseñada puede superar una mala estructura o mala codificación.

  Eficiente

  No solo una pieza de software debería funcionar como se espera, sino que también debe ser eficaz. Debe ser optimizado para una arquitectura específica y debe funcionar sin problemas bajo las estructuras subyacentes y subsistemas.

  Agradable y fácil de navegar en la GUI

  Cuando un diseñador opta por ir por las tendencias en lugar de lo que funciona, hace una experiencia desagradable para el usuario final. El diseño de los menús desplegables está probado y ha trabajado durante años, pero desde hace tiempo ya necesitaba una actualización, pero esta actualización no debe hacerse a expensas de la intuición. El objetivo principal de la GUI es hacerle el trabajo más fácil al usuario final.

Fácil de desinstalar

Además de ser fácil de instalar y utilizar, una pieza de software debe ser fácil de quitar. Sin un proceso de eliminación simple, el software se vuelve engorroso. Por mucho que los desarrolladores no quieran que sus usuarios extraigan su software, el proceso de eliminación puede ser la última impresión que deje su software.

Fácil de solucionar

Ningún software es perfecto. Y cuando algo va mal con una pieza de software, es importante que el usuario final pueda llamar a la asistencia y que puedan resolver el problema.

Apegarse a los estándares

Las normas son creadas por una razón, para hacer la interconexión entre las aplicaciones o hardware fácil. Los problemas empiezan a surgir cuando los desarrolladores no se adhieren a las normas.

Efectivo control de errores

¿Qué sucede cuando una pieza de software encuentra un error? ¿Sólo desaparece sin previo aviso? ¿Se trata de corregir el problema? Cuando un programa se encuentra con un error debe hacerse conocido, al menos para los desarrolladores. No es responsabilidad de los usuarios finales informar de fallos, pero si les das opción de informar de los errores pueden ayudar a mejorar el software.

- **Que el vocabulario no se quede solamente en la interpretación del alfabeto si no que incluya otros elementos básicos del lenguaje.**

La mayoría de los proyectos de ésta índole se basan en el alfabeto de señas, puesto que es el elemento más básico del lenguaje, evidentemente no es una mala elección sin embargo pensando en lo vasto que es el lenguaje de señas, este elemento constituye solo una pequeña parte y si el señante depende de solo este conocimiento la única manera de comunicarse sería deletreando la palabra que quisiera expresar, lo cual es sumamente ineficiente, sobre todo cuando existen señas específicas para comunicar palabras o ideas completas.
Para definir a los elementos básicos del lenguaje se plantea tomar la estructura de aprendizaje de un idioma, por ejemplo el idioma inglés, si se considera ¿qué es lo que aprendemos primero del idioma? Se encuentra con el abecedario, saludos, lugares, comida, días de la semana, meses del año, números, etc.

- **Que el sistema sea simple**

  La “simpleza” es un elemento que es difícil de definir, pero para este proyecto se relacionará con el número de elementos ajenos al software que asimismo abarca a todo el dispositivo electrónico, entonces, el software deberá contar con el menor número de complementos auxiliares posible, sin embargo no solo se puede atacar por este medio, pues la tableta electrónica de *MotionSavy* depende del sensor *LeapMotion* como único complemento, así que además de depender de un pequeño número de complementos, estos deben ser de fácil obtención, es decir que sean asequibles.

Lo anterior lleva a la siguiente lista de “Requerimientos y especificaciones”:

Tabla 2.Requerimientos y especificaciones del software.

Requerimientos	Especificaciones	Rango	Unidad
El software podrá ejecutarse a cualquier hora del día y en cualquier lugar, además podrá ejecutarse en el mayor número de dispositivos posible.	Versiones de Windows o Android de los dispositivos a ejecutarse.	Windows7 a Windows10 O Android 4.4 a Android 6	
El software podrá evaluar el desempeño del usuario.	Error entre seña modelo y seña realizada por el usuario.	10 - 20	%
El software será amigable con las personas que se quieran introducir en el lenguaje de señas o aquellos que se encuentren en etapas tempranas del aprendizaje.	Tiempo de retraso.	1 - 3	Segundos
	Tiempo de instalación y procesos de instalación.	1 - 2 1 - 20	Procesos Minutos
	Ventanas con no más de 5 botones.	1 - 5	Botones
	Número máximo de diferentes tipos de ventanas.	3 - 10	Ventanas
Que el vocabulario no se quede solamente en la interpretación del alfabeto si no que tendrá otros elementos básicos del lenguaje.	Número de señas.	26 -	Señas
Que el sistema sea simple.	Número de elementos o dispositivos externos necesarios para su funcionamiento.	0 - 2	Elementos

CAPÍTULO 3. DISEÑO CONCEPTUAL

Una de las especificaciones es que se necesiten como máximo de dos elementos externos al dispositivo necesarios para el funcionamiento del mismo, pero el ideal es que ese número se pudiera reducir a cero. El primer concepto del proyecto se tiene en las siguientes figuras (Fig. 11 y Fig. 12), como se observa el dispositivo podría ser capaz de reconocer los gestos del señante, esto sin necesidad de un medio externo auxiliar.

El proceso se reduciría a colocarse frente al dispositivo y que este pudiera instruir y evaluar las señas del usuario.

Fig.11. El usuario se colocará frente al dispositivo para comenzar con el aprendizaje del lenguaje de señas.

Fig. 12. El sistema será capaz de captar los movimientos del usuario.

Siguiendo con el concepto de software de enseñanza, el programa podrá manejarse de manera fluida, esto ayudándose de una interfaz intuitiva, pero para cubrir cualquier contratiempo la guía que ofrezca el mismo programa será importante, videos demostrativos de uso y ventanas que indiquen al usuario la acción a realizar será la forma de guiar al usuario.

El funcionamiento del programa se basará en la “imitación”, se plantea una ventana previa la cual contenga la seña a realizar, el concepto se bosqueja en la Fig.13.

Fig. 13. Se indicará de la forma más clara posible la seña a reproducir.

Cuando el usuario se encuentre listo para reproducir la seña se pasará a una nueva ventana que se encargará de evaluar si la seña se ha realizado correctamente, respondiendo con mensajes de retroalimentación de acuerdo con el desempeño del usuario, Fig. 14 y Fig. 15.

Fig. 14. Mensaje de felicitación cuando el usuario realiza la seña correctamente.

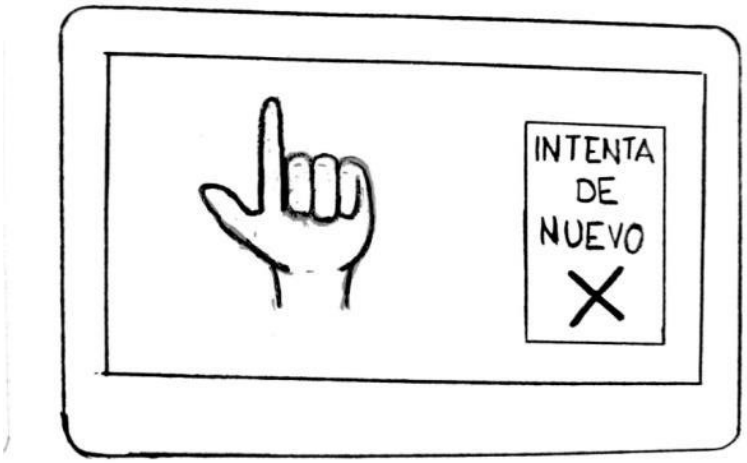

Fig. 15. Si el usuario falla realizando la seña aparecerá un mensaje de retroalimentación negativa.

La dinámica del programa no se ha pensado al azar, ya que siendo finalmente el objetivo enseñar el lenguaje de señas, se han estudiado los sitios más populares para aprender idiomas online y se han tomado como base para desarrollar la dinámica del software.

Esto conlleva a tener un menú principal, éste contendrá las señas de acuerdo a una clasificación, separándolas en “Abecedario”, “Días de la semana”, etc. (Fig.16).

Fig. 16. El menú será la pantalla principal y con ella el usuario podrá escoger las señas que quiere aprender o practicar.

Un lenguaje se compone de un gran número de palabras, por ejemplo el diccionario de la RAE contiene 88.000 palabras [16], sin embargo al aprender un idioma se hace por niveles, regularmente se hace uso de un “vocabulario básico”, para ello se tomará como base el contenido de un sitio de enseñanza de idiomas, Duolingo [17].

Duolingo es una exitosa aplicación para la enseñanza de idiomas, con una calificación de 4.6 de 5 puntos posibles, y excelentes opiniones de diferentes instituciones [18]:

"Entre las aplicaciones para aprender o practicar idiomas, no hay nada mejor que Duolingo" —PC Magazine, selección del editor como mejor aplicación para aprender idiomas.

"Sin la menor duda, la mejor aplicación para aprender idiomas" —The Wall Street Journal.

"También hemos probado con otros métodos como libros y Rosetta Stone. Y ninguno de los dos funcionó. Por eso me entusiasmé con la oportunidad de usar Duolingo... y créanme que es adictivo" —Fluent in 3 Months.

Aquí un listado de los principales elementos:

- Saludos
- Viajes
- Restaurante
- Ropa
- Escuela
- Animales
- Amigos
- Colores
- Comida
- Preguntas
- El cuerpo
- Preposiciones
- Fechas
- Objetos
- Números
- Vocabulario
- Deportes
- Arte
- Medicina
- Política
- Naturaleza
- Ciencia

Todas las palabras contenidas en los elementos podrían ser parte de un “vocabulario meta”, es deseable que el software contenga todas esas palabras pero debido a que no se puede establecer una relación directa ente un idioma y otro deben cumplir con que **las palabras tengan traducción.**

Entonces, se tendrá un menú principal con todas las señas que el usuario podrá aprender, el sistema mostrará la seña a realizar de forma clara, captará los movimientos del usuario y determinará si la seña se reprodujo de forma correcta, respondiendo con un mensaje de retroalimentación.

CAPÍTULO 4. DISEÑO DE DETALLE

Como ya se planteó anteriormente, la idea es que el sistema pueda captar el movimiento del usuario, si se apega al boceto del capítulo anterior se puede relacionar directamente que la forma de hacerlo será mediante el uso de una cámara la cual estará integrada en el dispositivo, entonces, el dispositivo será una computadora o un dispositivo móvil, celular o tableta electrónica.

Debido a que se trabajará con una cámara, las imágenes serán el medio por el cual se obtendrá la información necesaria para poder evaluar al usuario, por lo tanto se hará uso de procesamiento de imágenes, o sea, visión artificial.

En la ingeniería eléctrica e informática, el procesamiento de imágenes (visión artificial) es cualquier forma de procesamiento de la señal para la cual la entrada es una imagen, como fotografías o marcos de vídeo; la salida de procesamiento de imagen puede ser ya sea una imagen o un conjunto de características o parámetros relacionados con la imagen. [19]

HERRAMIENTAS PARA EL PROCESAMIENTO DE IMÁGENES.

Hay un gran número de herramientas de visión artificial, las cuales en su mayoría se traducen a paquetes de software, que van desde programas de tarea específica los cuales son comercializados por empresas especializadas hasta programas open source sin costo.

- **NI Vision Builder**

Este software pertenece a la empresa National Instruments, la descripción de éste es obtenida de su página oficial:

Vision Builder AI le proporciona una manera fácil de configurar, e implementar un sistema que se ocupa de las aplicaciones de visión de coincidencia de patrones para la lectura de códigos de detección de presencia y de alineación de precisión y clasificación. Un entorno de desarrollo basado en menús interactivos sustituye a la complejidad de la programación, por lo que el proceso de desarrollo y mantenimiento es simple, sin sacrificar el rendimiento o la gama de funcionalidad. [20]

Si bien su uso generalmente es en la industria, tiene algunas herramientas predefinidas que serían muy útiles para cualquier otra aplicación pues sus ejemplos van desde identificación de piezas en una línea de producción hasta el seguimiento de un automóvil en una carretera.

Su lenguaje de programación es del más alto nivel pues al igual que el entorno de programación LabView se basa en interconexión de bloques predefinidos. Pero como se podría prever esta herramienta tiene un costo, es cual es de MX$ 40,940 por una licencia.

- **OpenCV**

OpenCV es liberado bajo una licencia BSD y por lo tanto está libre tanto para uso académico y comercial. Tiene interfaces de C++, C, Python y Java y es compatible con Windows, Linux, Mac OS, iOS y Android. OpenCV fue diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real. Escrito en optimizado C / C++, la biblioteca puede tomar ventaja de procesamiento multi-núcleo. Se activa con OpenCL, se puede aprovechar la aceleración de hardware de la plataforma de computación heterogénea subyacente. Adoptado en todo el mundo, OpenCV tiene más de 47 mil personas de la comunidad de usuarios y el número estimado de descargas superiores a 9 millones. [21]

Como puede verse, se trata de un software sumamente versátil, dado que puede trabajar en prácticamente cualquier sistema operativo y se puede desarrollar en los lenguajes de programación más populares.

- **Matlab, image processing**

MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux .

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

El toolbox a usar sería el de Image Processing, el cual es especializado para el procesamiento de imágenes.

- **Software de Microsoft (Kinect)**

No es la primera vez que éste dispositivo es mencionado en la presente tesis, y no es para menos, éste elemento no solo se compone de una cámara, sino que funciona con dos diferentes tipos de cámaras, una cámara RGB y una cámara IR (infrarroja), con esto no solo puede obtenerse una imagen en dos dimensiones sino que puede obtenerse una tercera, es decir, puede trabajar con profundidad.

El sensor tiene un costo de aprox. $2000 M.N., y se ha vuelto muy recurrido para un gran número de desarrolladores, pues Microsoft, empresa dueña de la patente, ha decidido hacerlo open source, proveyendo de una librería básica para comenzar con su programación. Si bien en primera instancia se creó como una herramienta de juego para la consola Xbox 360, su uso se ha expandido a programarse con una computadora.

Por supuesto existen más herramientas dedicadas al procesamiento de imágenes, sin embargo se han contemplado las anteriores por popularidad y documentación. Cada una de ellas tiene sus puntos fuertes que las hacen opciones sumamente viables para ser la herramienta de desarrollo del proyecto. Para elegir la más adecuada se procederá a elaborar una matriz de decisión.

Los parámetros a evaluar serán:

- Precio: pensando en las etapas de desarrollo y producción, es importante cuidar este rubro.
- Programación GUI: ya que la idea es poder realizar un producto terminado con el que pueda interactuar un usuario cualquiera, el que la herramienta permita hacer una interfaz gráfica se vuelve un punto importante.
- Conocimiento del lenguaje: tomando en cuenta que el objetivo es poder realizar el algoritmo básico y que se buscará hacerlo de modo que pueda migrarse a otro lenguaje de programación, en primer plano estará el que se pueda realizar con una herramienta conocida o con la que se haya trabajado antes para ahorrar tiempo en familiarizarse con otro lenguaje desconocido.
- Elemento externo: siendo éste también un parámetro de diseño, si la herramienta es en sí misma un elemento ajeno al dispositivo se tomará como un parámetro en contra.
- Documentación: este parámetro puede ir de la mano con el de “conocimiento del lenguaje”, sin embargo *visión artificial* es una teoría que no se enseña con profundidad en la Universidad y que por lo mismo necesitará de ser autodidacta y apoyarse de tutoriales e información que, ya sea la herramienta misma u otros usuarios provean.
- Compatibilidad con sistemas operativos: este punto habla de versatilidad y con ello se pude expandir el proyecto a un mayor número de dispositivos, por lo tanto llegando también a un mayor número de personas, esta condición se toma con dispositivos de la misma naturaleza.
- Compatibilidad con dispositivos: se asemeja al parámetro anterior sin embargo esta evalúa de manera más general, pues en este se considera que la herramienta pueda ejecutarse tanto en computadoras, celulares y tabletas electrónicas.

La escala de evaluación irá del 0 al 5, donde el 0 se trata de la peor calificación y 5 la mejor.

Tabla 3. Matriz de desición para evaluar las características de los diferentes entornos de desarrollo.

HERRAMIENTA / CARACTERÍSTICA	NI Vision Builder	OpenCV	Matlab	Kinect
PRECIO	1	5	4	3
PROGRAMACIÓN GUI	2	3	5	2
CONOCIMIENTO DEL LENGUAJE	3	3	5	3
ELEMENTO EXTERNO	5	5	5	0
DOCUMENTACIÓN	2	4	4	4
COMPATIBILIDAD CON SISTEMAS OPERATIVOS	3	5	4	4
COMPATIBILIDAD CON DISPOSITIVOS	4	4	5	4
TOTAL	**20**	**29**	**32**	**20**

De acuerdo a la matriz de decisión, la puntuación de Matlab supera por un pequeño margen a las otras herramientas, sobre todo por la facilidad de crear GUI y que es un software muy usual en las últimas etapas de la carrera por lo que el conocimiento de su programación se tiene presente en mayor medida.

Sin embargo se plantea como objetivo que se pueda lograr un modo de operación que se pueda migrar fácilmente a las otras herramientas mencionadas.

De forma directa delimitamos el proyecto a que pueda ejecutarse en una computadora con cámara. Es importante decir que se tiene otro beneficio en primera instancia, que es la capacidad de procesamiento, se espera que primero se evalúe en un sistema con mejores capacidades como primera aproximación y si este está sobrado, pueda migrarse también a celulares o tabletas electrónicas.

ALGORITMO PARA LA DETECCIÓN DE SEÑAS

Como primer elemento básico en el lenguaje de señas se tiene al alfabeto, y se considera también a éste también como el elemento mínimo que debe poder evaluar el software, así que se plantea el primer reto: que el software pueda reconocer la mano.

La información básica que puede obtenerse de una imagen es la del color de los pixeles, y dado que las cámaras que poseen las computadoras trabajan en la escala de colores RGB (Rojo, Verde y Azul, en inglés) se tratarán a estos tres valores como la primera información con la cual se trabajará.

En primera instancia se trabajará con la siguiente fotografía modelo:

Fig. 17. Imagen que servirá como modelo para las primeras operaciones.

Debe poder reconocerse la mano del resto de la imagen, es decir, del fondo. Para este ejemplo la tarea no parece compleja pues se observa un gran contraste entre un elemento y otro, para el siguiente paso se usará el concepto de binarización, esto consiste en colorear de blanco los pixeles de interés y los pixeles que componen el fondo tornarlos de color negro.

Entonces se recorre toda la imagen, pixel por pixel decidiendo si el color del pixel en cuestión es o no el de interés, es decir, se realizará un proceso de comparación, esto traducido a un lenguaje de programación consiste en un operador “if”.

Para realizar este proceso se necesita de un valor o rango de valores deseados, para ello se optará por lo segundo, ya que si volvemos a analizar la imagen se puede observar que si bien un color puede ser predominante hay zonas de la mano que varían en tonalidad y que en caso de optar por comparar con un valor fijo se podría descartar a zonas de la mano demasiado grandes perdiendo así la forma.

El rango de valores debe seleccionarse de forma cuidadosa pues de hacerse muy pequeño se corre el riesgo de perder pixeles importantes y en caso de elegir un rango muy amplio de tener información innecesaria.

Esto último se ilustra a continuación:

Probamos con un valor promedio de R=170, G=130, B=160, y una tolerancia de 60, lo que da como resultado la siguiente imagen binarizada.

Fig. 18. A la izquierda la imagen sin tratamiento alguno, al centro el color resultado del valor RGB promedio, a la derecha la imagen binarizada en torno a dicho valor RGB.

Ahora, si se mantiene el valor RGB promedio pero se disminuye la tolerancia a 30:

Fig. 19. Imagen resultado de reducir la tolerancia, en ésta, la forma de la mano se pierde casi por completo resaltando solo un pequeño número de pixeles.

Como era de esperarse al reducirse la tolerancia se reduce también el número de pixeles que cumple con las condiciones establecidas.

Entonces, se observa que con una buena selección del valor RGB promedio y de la tolerancia se puede aislar el área de interés, sin embargo con esta primera operación llega también el primer contratiempo:

¿Qué sucede si hay un cambio en la tonalidad de la piel?, esto provocaría que el rango de comparación sea también variable, lo cual puede presentarse de dos maneras:

1.- Personas con diferente color de tez.

2.- Cambio en la iluminación del ambiente.

Es la segunda uno de los problemas más comunes en el procesamiento de imágenes, de modo que se han desarrollado complejos algoritmos para eliminar este “ruido”.

Para lidiar con este problema se han investigado filtros comunes en los distintos entornos de programación, siendo uno de ellos el *imfilter*.

Fig. 20. La acción de *imfilter* en una imagen RGB. [22]

Como se puede observar, se pierde nitidez y detalle, sin embargo estas condiciones no interesan por el modo en el que se está atacando el problema, siendo en realidad benéfico porque ahora se tiene solo el parámetro más importante: el color.

Este filtro aplicado a la imagen en cuestión da como resultado:

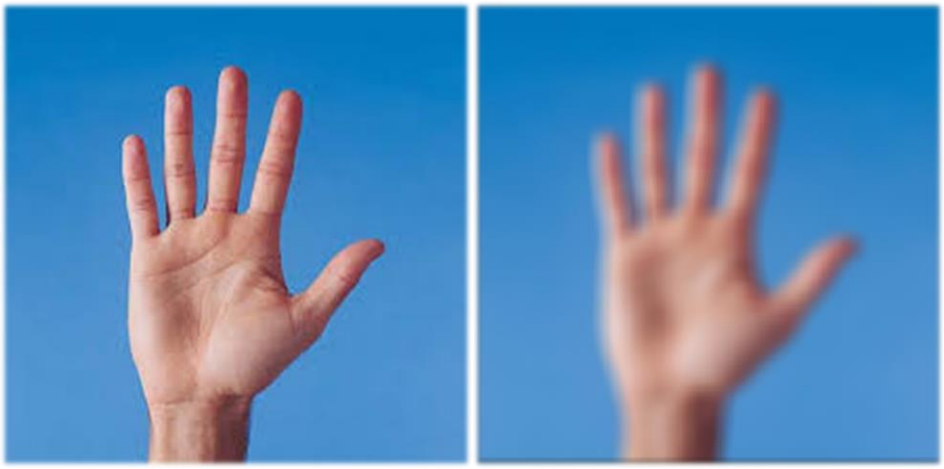

Fig. 21. A la izquierda la imagen sin ningún filtro, a la derecha la misma imagen pero bajo el filtro de *imfilter*.

Ahora, si se juntan la operación de binarización y el filtro, el resultado es el siguiente:

Fig. 22. A la izquierda la imagen sin tratamiento alguno, al centro el color resultado del valor RGB promedio, a la derecha la imagen resultado de aplicar el filtro *imfilter* y binarizada en torno a dicho valor RGB.

La forma de la mano es más distinguible, es decir, hay un mayor número de pixeles pertenecientes a la mano identificados de manera exitosa, o sea iluminados de color blanco, y el "ruido" disminuye considerablemente.

Con una posible solución a la variación de luz se aborda el otro punto: la variación de la tonalidad de la piel. Pero, ¿cómo evaluar un parámetro tan variante?, pues haciendo la referencia variante también. La forma que se propone para lidiar con este problema es que se pase por una etapa de calibración, en ella se extraerá cuál es el color predominante y se establecerá como la nueva referencia del color.

Se le pide al usuario colocar su mano en una zona específica de la pantalla y se extraerá la información del color RGB de los pixeles, se obtendrá un valor promedio y se elaborará un rango de valores en torno a ese nuevo valor de referencia.

Como se tienen ya algunos pasos del tratamiento de la imagen planteados, a partir de esta etapa se trabaja de forma paralela la del diseño de la interfaz y el algoritmo de tratamiento de las imágenes.

Entonces se plantea una ventana de calibración, obteniéndose la siguiente apariencia:

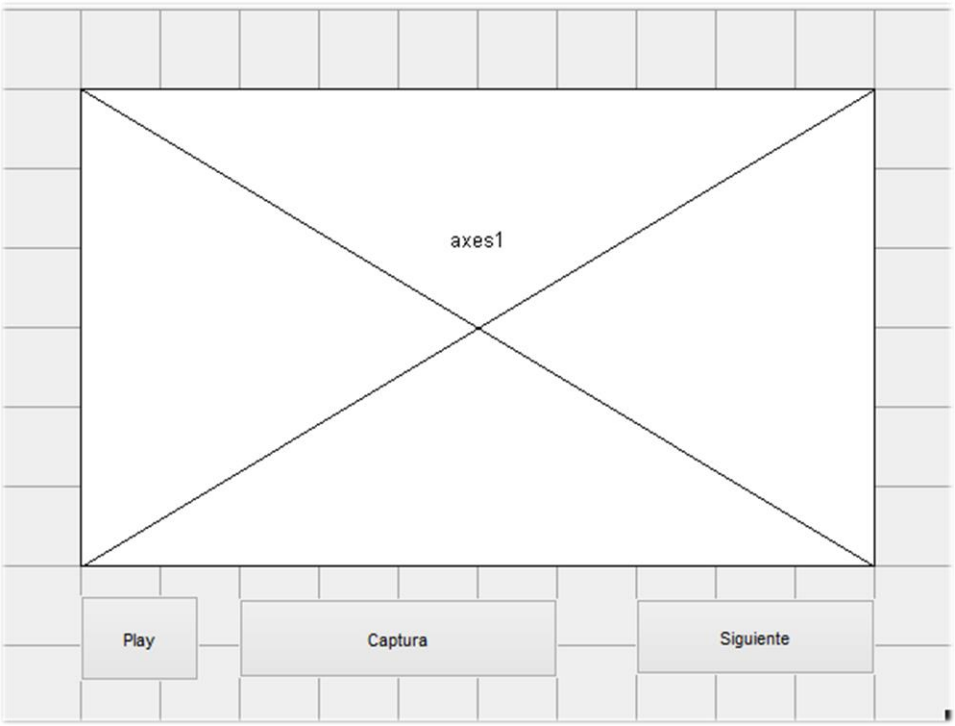

Fig. 23. Apariencia de la ventana de calibración en la herramienta de diseño de GUI.

La ventana contiene tres botones: *Play, Captura, Siguiente.*

Al presionar el botón “Play” la cámara comenzará a captar las imágenes, una vez que el usuario esté listo presionará el botón “Captura” y el programa obtendrá el valor promedio RGB.

La apariencia de la ventana en funcionamiento es la siguiente:

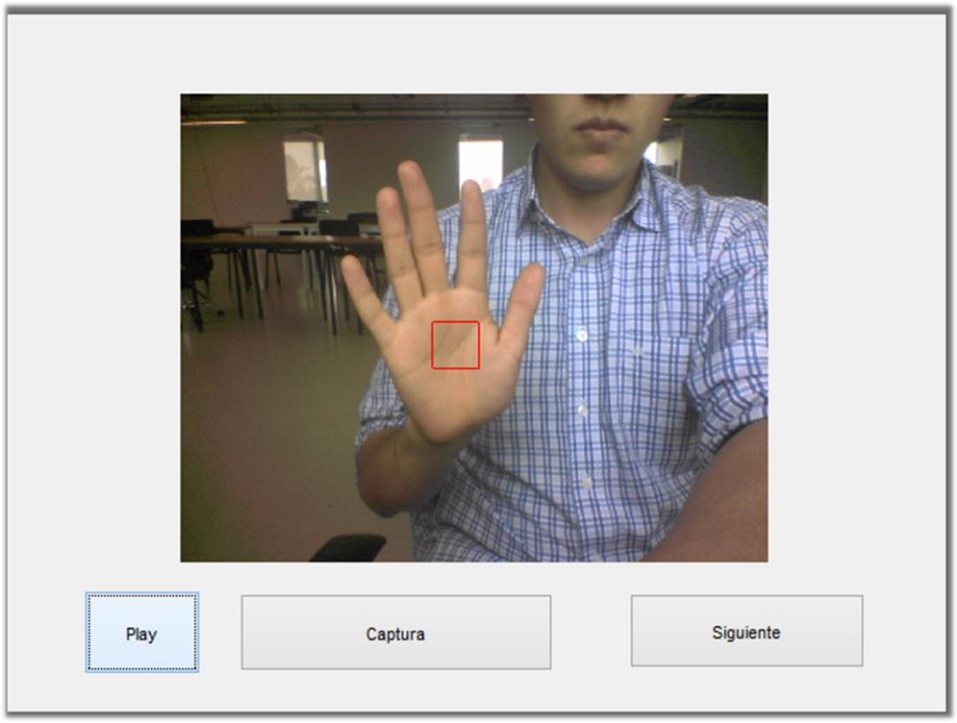

Fig. 24. Apariencia de la ventana de calibración una vez ejecutando el código.

El cuadrado de color rojo representa la zona de la cual se extraerá la información del color de la piel de la mano, con esto se tendrá la nueva referencia para elegir el rango de valores para realizar la binarización.

Con los pasos anteriores implementados, se prueban en tiempo real y con un fondo no controlado, pues si bien el lector pudo notarlo con las pruebas anteriores, la imagen ejemplo tenía un fondo de color uniforme, los resultados son los que se muestran a continuación:

Fig. 25. En la parte superior, la imagen sin ningún tipo de procesamiento. En la parte inferior el resultado de la imagen después de toda la etapa de calibración y aplicación del filtro.

Si se observa la imagen procesada, ésta demuestra que se pudo identificar la coloración de la mano, y por ende la de la piel del usuario, sin embargo hay partes del fondo cuya coloración es muy parecida y por lo tanto el software la procesa como pixeles de interés.

Este problema se presenta porque el color de algún objeto de fondo es muy parecido a la de la tez del usuario aunado al intervalo de tolerancia manejado para la binarización.

Además siendo el primer objetivo la identificación de la mano, refiriéndose a la imagen procesada, además se identifican brazos y cara, por ser de la misma coloración, esto puede llegar a ser de utilidad en próximas etapas, ya que como se mencionó en capítulos anteriores el lenguaje de señas no solo se compone de señas estáticas realizadas por las manos, sin embargo para este primer objetivo dicha identificación está de sobra.

En este punto se tiene como prioridad aislar las manos del medio, eso incluye a cualquier otra parte del cuerpo del usuario.

Para solucionar esto, se plantea el uso de un elemento auxiliar: guantes.

Si bien era un objetivo de diseño el usar el menor número de elementos externos auxiliares, se consideran a los guantes como un elemento bastante simple, además de que es muy asequible, sin embargo, dado que ya se tuvieron complicaciones por coloraciones semejantes, este guante debe tener una particularidad, es decir, que sea distinguible del entorno.

Dicha condición se puede cumplir con cualquier color que no se encuentre en fondo, sin embargo la idea del proyecto es proporcionar al usuario el sistema completo, tanto el software como los elementos con los cuales éste va a interactuar.

Como una condición importante, señalada también como objetivo de diseño, es que pueda funcionar en prácticamente cualquier ambiente, se propondrán las características de los guantes, esto conlleva a que se retome la vía del desarrollo del proyecto: el color, y es que el guante debe ser de un color poco usual y si es posible que ayude además a su fácil identificación.

De acuerdo a investigaciones realizadas con respecto a la percepción del color [23] en las prendas de vestir, el color naranja es el menos usado, pues en casi todos los gráficos es el peor posicionado, por ende es el menos usado, esta condición resulta sumamente beneficiosa para el sistema pues al ser un color que no se encuentra fácilmente tiene menos probabilidades de confundirse con el fondo de una imagen.

El guante elegido es el siguiente:

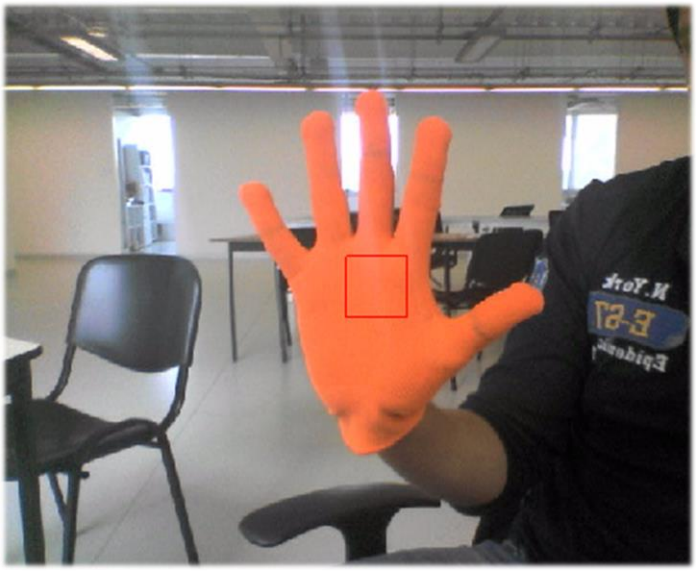

Fig. 26. Imagen del guante elegido, se pude notar que su color resalta de los colores que componen al fondo.

Y realizando la primera prueba introduciendo este nuevo elemento, obtenemos mejores resultados:

Fig. 27. Imagen procesada del guante y por lo tanto también de la mano.

Siendo éste el mejor resultado hasta el momento, pues la mano es distinguible prácticamente en su totalidad, es decir los pixeles que componen a la mano (guante) han sido casi todos reconocidos y diferenciados del fondo, y el entorno, el cual es el mismo bajo en cual se realizaron las pruebas anteriores, no tuvo injerencia alguna en la prueba, aunque aún son necesarias pruebas en diferentes ambientes.

Ya que se tiene la imagen procesada y se han obtenido buenos resultados con la identificación de la mano, es momento de empezar a procesar y obtener de ella información útil.

Para ello se introducirá un nuevo concepto, que es el etiquetado, en la imagen anterior se tiene solo un objeto, del cual se puede extraer información de manera relativamente sencilla, pero cuando se tengan en pantalla más de un objeto, por ejemplo, ambas manos, el software debe poder identificar que se trata de dos objetos con información propia.

El proceso puede ejemplificarse con la siguiente imagen:

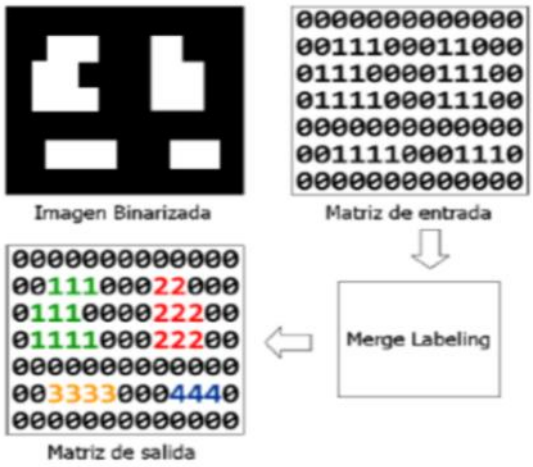

Fig. 28. Proceso de etiquetado, se parte de una imagen binarizada, en la cual los pixeles de interés son identificados con el número 1, el etiquetado asigna diferentes números a regiones distintas de modo que estos números sirven como etiquetas.

De la imagen podemos ver que se le asigna un número “etiqueta” a cada región o figura, esto permite que se puedan tratar como regiones diferentes, de modo que el centroide puede ser localizado fácilmente.

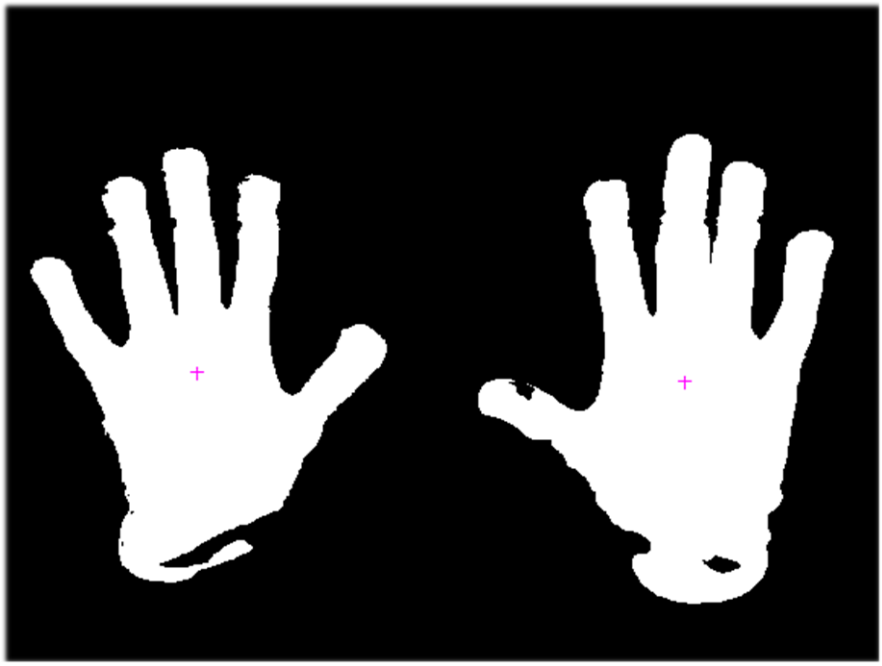

Fig. 29. Después del etiquetado se puede acceder a las propiedades de cada región, como el centroide, en la imagen se reconoce el centroide de cada mano con una cruz.

El siguiente paso es identificar los dedos de la mano, pues como se ha mencionado anteriormente, la posición de la mano conforma la parte más básica del lenguaje de señas; existen algunas técnicas que se apoyan de algoritmos básicos como la detección de bordes y la acción “ConvexHull” la cual forma un polígono con los puntos exteriores de la figura, lo siguiente es encontrar las intersecciones entre ambas trayectorias, dicho procedimiento se ilustra en la figura siguiente:

Fig. 30. Figura de identificación de los dedos basada en las operaciones de detección de bordes y ConvexHull. [24]

Sin embargo, para el presente proyecto se propone realizarlo de otra forma, para ahorrar procesamiento en operaciones se basará en la operación realizada anteriormente, o sea el etiquetado.

Y es que si ya se ha decidido la implementación de un guante, con la correcta modificación del mismo se pueden reducir las operaciones del programa haciéndolo más rápido.

Dicha modificación consiste en “separar” la zona media de las falanges del resto de la mano, esto se logra mediante una línea de otro color, preferentemente contrastante, siendo el color negro el elegido en este caso, el guante pasa a tener la siguiente apariencia:

Fig. 31. Apariencia final de guante, las lineas negras separan las "yemas" de los dedos del resto de la mano.

Lo cual repercute en la imagen binarizada con un mayor número de cuerpos.

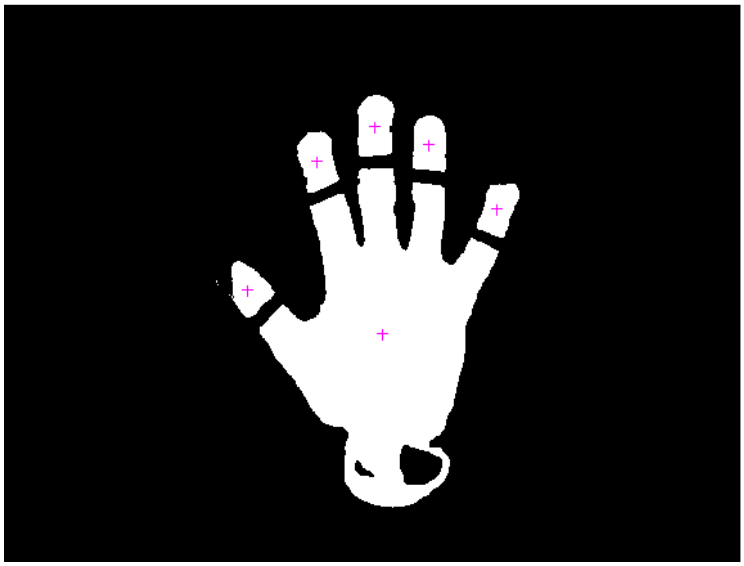

Fig. 32. Al aplicar todas las operaciones mencionadas se obtiene una imagen con 6 cuerpos identificados.

Con dicha modificación, el software etiquetaría 6 cuerpos, cada uno de ellos con sus propiedades, en este caso la del centroide sería la propiedad que nos será de utilidad pues para poder evaluar la posición de los dedos se trazarán rectas del centro de la mano a los respectivos centros de las áreas que representan a los dedos.

Con ambas operaciones realizadas se tiene el siguiente resultado:

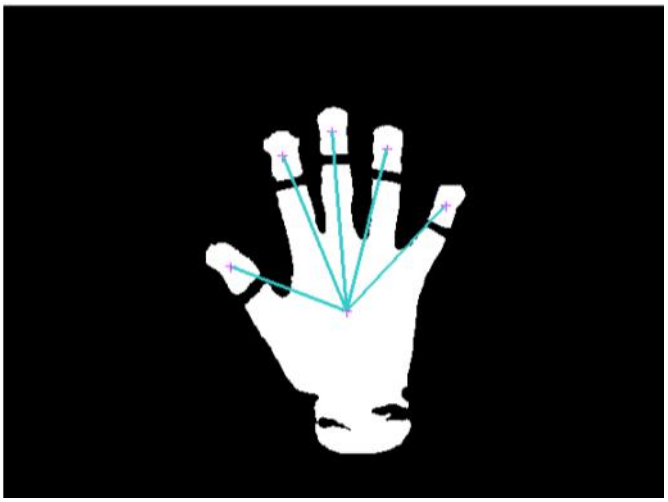

Fig. 33. Cada línea proporcionará la información de la posición de los dedos de la mano.

El proceso se ha mostrado con una sola mano en pantalla, siendo prácticamente el mismo para cuando se encuentren ambas manos frente a la cámara, sin embargo hay algunas adecuaciones que se han hecho para que no se tracen líneas de una mano a otra, siendo la clave la distancia, pues solo se trazan líneas entre elementos que se encuentren en un rango de distancias.

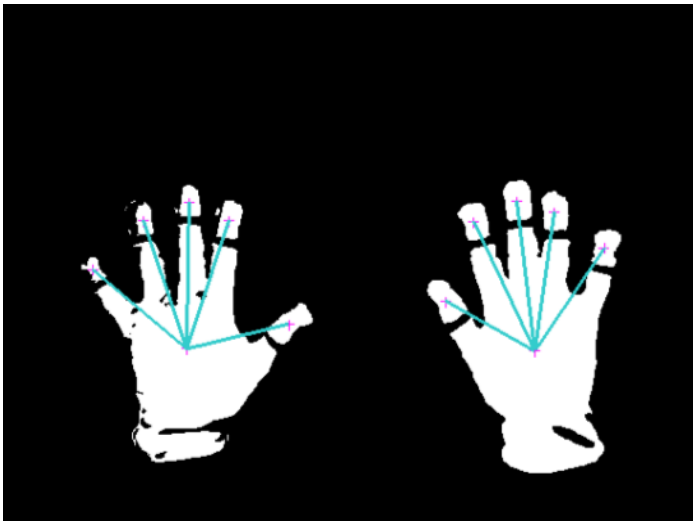

Fig. 34. El programa es capaz de poder obtener la posición de los dedos de ambas manos.

Con ambas manos en cuadro se puede observar que se puede extraer los parámetros de las líneas, que a su vez son representativas de los dedos.

Con los dedos, técnicamente identificados a través de las líneas, se debe decidir qué parámetros son los que se evaluarán, se sabe que pueden obtenerse tanto longitud como pendiente (ángulo); la longitud será un parámetro muy variable pues dependerá de qué tan cerca de la cámara se realiza la seña, es claro que entre más cerca se esté la longitud aumentará, y siendo así ésta se vuelve una cantidad no muy confiable, por otro lado el ángulo será siempre igual pues es independiente de la longitud de la recta, siendo este, entonces, el parámetro a evaluar.

Entonces, se obtiene el ángulo de cada una de las rectas que representan a los dedos.

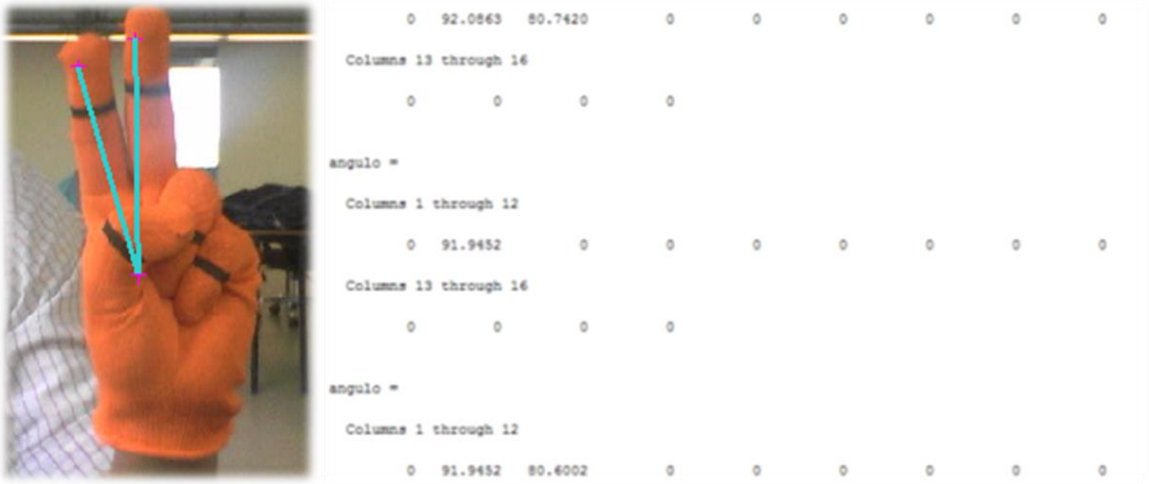

Fig.35. A la izquierda se muestra la imagen de la mano con dos líneas rectas que representan a los dedos, a la derecha los valores de los ángulos de dichas rectas.

Con dicha información se puede ahora, evaluar la correcta realización de una seña, y con estos parámetros el primer objetivo, que es el de poder evaluar la posición de las manos, se ha alcanzado.

El siguiente paso, es poder evaluar señas dinámicas, es decir, no solo que se está cumpliendo con la posición de la mano sino que el movimiento de la misma se realiza correctamente. Sin embargo, a lo largo del desarrollo del programa se ha observado que el tiempo de procesamiento se convierte en un problema, pues el programa debe realizar todas las correcciones y operaciones a la imagen para poder obtener el valor de los ángulos, proceso que lleva un determinado tiempo (que depende también de la capacidad de procesamiento del dispositivo), dicho tiempo, es, para una tarea de seguimiento, demasiado grande.

Uno de los parámetros a configurar en la obtención de imágenes es el tiempo entre capturas, y se ha comprobado que debe existir congruencia entre el tiempo de procesamiento y el tiempo entre cada captura, si el tiempo de procesamiento es pequeño las imágenes pueden tomarse con mayor frecuencia, pero en la ventana de evaluación de la seña el análisis es lo suficientemente complejo para que la tarea de seguimiento quede por el momento descartada.

Ahora que se ha desechado la idea de poder obtener el ángulo de los dedos de la mano en todo momento el modo de atacarlo debe replantearse.

El primer punto es que, dado los recursos con los que se está trabajando, no es posible un análisis tan completo, es decir que se pueda evaluar la posición de los dedos de la mano a cada punto de la trayectoria por lo que el modo de operación (posterior a la etapa de calibración) se plantea de la siguiente manera:

1.- El programa enseñará al usuario una ventana con la imagen a reproducir en ella.

2.-El usuario intentará copiar la seña, de hacerlo correctamente se pasará a una nueva ventana.

3.- En la nueva ventana se evaluará la posición de toda la mano, y ya no la posición de los dedos de la mano como se hacía anteriormente, la cual tendrá que cumplir con una trayectoria específica.

4.- En caso de haber cumplido con las dos etapas de manera satisfactoria se mostrará un mensaje de felicitación.

Ya que se ha definido la forma en la que se atacará el cumplimiento de la trayectoria, se han generado dos propuestas:

La primera consiste en mostrar un video demostrativo con la seña a realizar, seguido de una ventana en la cual el usuario intentará realizar la seña en cuestión, para apoyar en el aspecto visual, el movimiento de las manos estará representado por una serie de marcadores los cuales en su conjunto formarán la trayectoria que ha seguido la mano, funcionando como una especie de rastro, una vez que se haya terminado con dicho proceso el software se encargará de comparar la trayectoria formada por la serie de puntos con la trayectoria “modelo”, teniendo como resultado un porcentaje de error cuyo valor determinará si el movimiento se ha realizado correctamente, el proceso se ve ilustrado en la Fig. 36.

La segunda propuesta consta de menos pasos, pues consistiría de una única ventana, en ella se encontrará una trayectoria a cumplir, compuesta por una serie de puntos o marcadores, el usuario reproducirá el movimiento marcado por dicha trayectoria, conforme se cumpla con ella los puntos se irán desvaneciendo, por supuesto, la seña se contará como realizada correctamente cuando se cumpla con todos los puntos, es decir, cuando todos los marcadores se desvanezcan, tal como se puede observar en la Fig. 37.

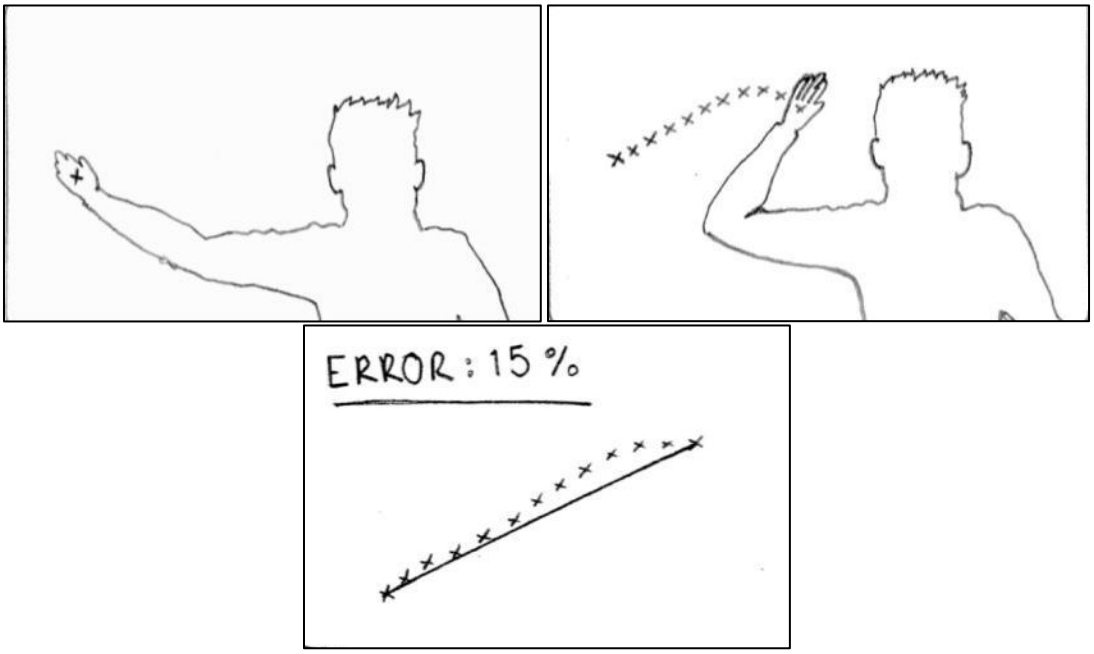

Figura 36. Ilustraciones correspondientes a la primer propuesta, se muestra como el usuario deja la "estela" por el movimiento de su mano, posteriormente se obtiene el error entre la trayectoria ideal y la trayectoria realizada por el usuario.

Figura 37. Ilustraciones correspondientes a la segunda propuesta, en ella la trayectoria deseada se marca desde el principio y conforme el usuario la cumple se desvanece, terminar con los puntos significa haber realizado correctamente la seña.

Independientemente de cual sea la propuesta elegida hay un parámetro que debe de cumplirse para que puedan llevarse a cabo:

Como se ha mencionado el movimiento de las manos es sumamente importante al realizar una seña, pues las trayectorias son muy específicas, algunos movimientos deben realizarse a la altura de la cabeza, otras en la zona media del cuerpo, o algunas señalan partes específicas del cuerpo. Es por eso que es imperativo tener la referencia de la posición del cuerpo.

Existen algoritmos, basados en reconocimiento de patrones que identifican las partes del cuerpo, sin embargo, como ya se ha comentado, la velocidad de procesamiento es un punto que debe cuidarse, y un proceso complejo lleva por consiguiente más tiempo de procesamiento, así que como alternativa se propone un escenario más controlado, esto es, que el usuario se mantenga a una distancia específica de la computadora, con esto se tendrá una idea muy aproximada de la posición del cuerpo y de las diferentes partes que lo componen.

Esto se logrará mediante una marca en la imagen del video, se plantea que la manera más sencilla de lograrlo es poner un óvalo que marque la posición de la cabeza, de este modo, quedando fija la posición del cuerpo del usuario los puntos que componen las trayectorias a seguir quedarán fijos igualmente.

Debido a que se trata de un proceso simple, porque reduce el número de ventanas y la evaluación requiere de menos operaciones, se ha elegido la segunda propuesta para evaluar las trayectorias.

Una vez implementados los puntos anteriores, se llega a los siguientes resultados:

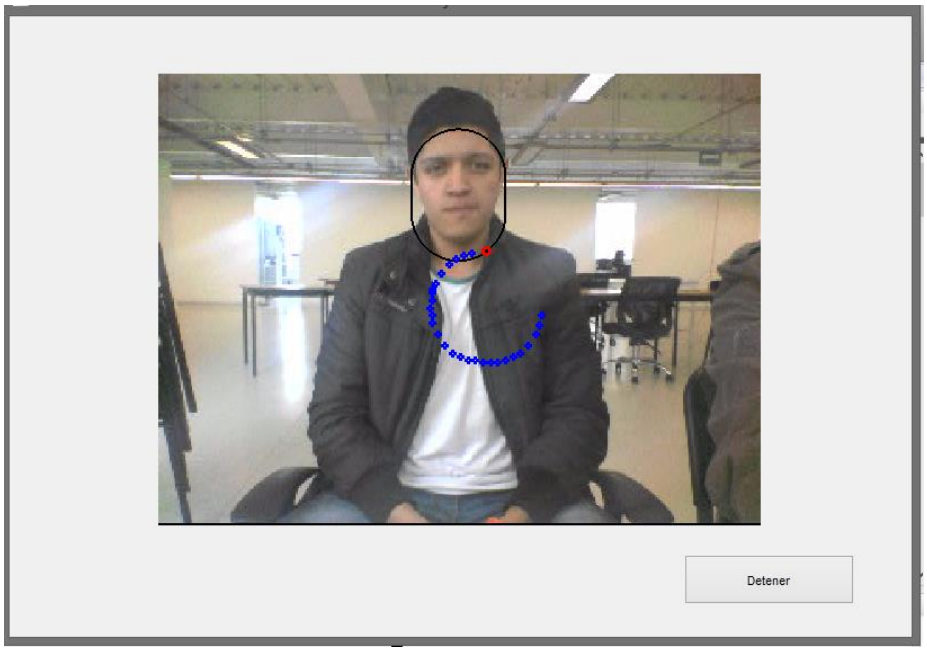

Figura 38. El óvalo de color negro indica la posición en la cual debe estar la cara. En azul se marca la trayectoria a seguir, el comienzo de la misma se marca de color rojo

El proceso de evaluación en este punto ha culminado, pues se puede evaluar la posición de los dedos de la mano, y la posición de las manos también, aspectos necesarios para poder evaluar una seña.

Hasta este punto, ya que se tiene todo el algoritmo planteado, es necesario probarlo en un ambiente diferente al regular en donde se desarrolló la mayoría del programa.

Figura 38. Resultados de la prueba al aire libre.

El sistema se probó en un entorno al aire libre, como se observa en la Fig. 38, la identificación del guante tiene buenos resultados pues logra identificar la mayoría de los dedos, sin embargo se ve afectado por el “ruido” o mejor dicho por pequeñas zonas del fondo que confunde por la coloración, en la imagen pueden observarse dos de ellas, ambas en el brazo, una sin injerencia porque no hay ningúna línea que la una con otro punto, pero la otra al estar cerca de la posición de la mano es confundida por un dedo, este fenómeno afectará en el momento de evaluar la seña.

Sin embargo las condiciones desfavorables de luz en el ambiente pueden ser desfavorables en cualquier entorno no controlado, aún en el lugar regular donde se ha realizado el programa, tal como se observa en la Fig. 39, donde si bien no hay trazos determinantes se observa que se han identificado zonas que debía ignorar.

Figura 39. Con ciertas condiciones de luz, el sistema puede confundir aún zonas del fondo.

A pesar de que se han aplicado operaciones a las imágenes adquiridas, hay ambientes en los cuales dichas operaciones no son suficientes, pues el software no es capaz de distinguir de manera eficaz el guante del fondo de la imagen, es decir, la cantidad de luz sigue afectando el funcionamiento del programa. Hay que considerar que se ha estado trabajando con una cámara de 1.31 Megapixeles, resolución baja para el promedio de computadoras en el mercado, aspecto que mas allá de considerarse conflictivo o limitante será tomado como un punto benéfico, pues si el sistema trabaja de manera correcta con una baja resolución no tendrá problemas para hacerlo con cualquier otra.

Ahora, dado que ninguno de los procesos de evaluación tiene injerencia directa en la imagen y solo se ha aplicado una operación (*imfilter*), aún pueden hacerse algunas otras operaciones para lograr manipular la imagen de modo que sea más fácil la adquisición de los datos.

Para ello se abordarán dos características básicas en una imagen:

- Contraste
- Brillo

Debido a que se tratan de operaciones básicas en el tratamiento de imágenes, la gran mayoría de herramientas de procesamiento de imágenes las incluyen en sus librerías, en este caso no es la excepción y hay un comando que hace esas correcciones de forma automática.

Dicho comando es *imadjust*, en la página del software se pueden ver ejemplos de lo que esta operación pude lograr en una imagen, figuras 41 y 42.

Figura 41. Ejemplo de la aplicación de la herramienta *imadjust* en una imagen *gray.* [25]

Figura 42. Ejemplo de la aplicación de la herramienta *imadjust* en una imagen *RGB.* [25]

Los colores son más distinguibles, es decir, existe un mayor contraste entre ellos.

Dicho comando admite como argumentos el contraste y el brillo en R, G, y B, dando un total de seis, representándose con valores de 0 a 1.

Para ejemplificar la injerencia de la operación, se tomará como modelo una imagen antes empleada (Fig. 26).

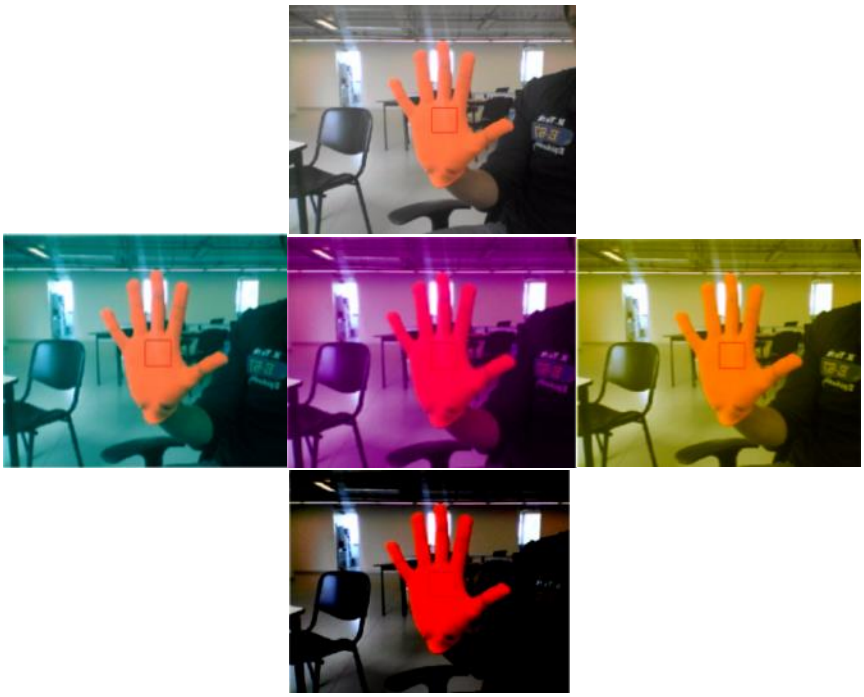

Figura 43. En la parte superior se observa a la imagen sin procesamiento alguno; en la parte central la misma imagen después de ajustar el contraste en R, G, y B respectivamente; en la parte inferior se muestra la imagen resultado de aplicar los mismos ajusten en R, G, y B de forma simultánea.

Como se puede notar en la Fig. 43, el contraste con algun cambio en el valor de R,G,B, o en los tres juntos producen un notable cambio en la imagen original, y dependerá del entorno que la modificación de dichos parámetros deba adecuarse para obtener la imagen idónea para procesarla.

Figura 44. De izquierda a derecha se muestra el aumento de brillo en la imagen modelo.

El brillo no tiene gran injerencia en la composición de los colores de la imagen como se aprecia en la Fig. 44, y su trabajo es el de aclarar u obscurecer una imagen sin embargo en ciertas situaciones puede tratarse de una herramienta muy útil.

Como el tratamiento de la imagen es uno de los elementos iniciales, debe aplicarse desde la etapa de calibración, por lo que se ha decidido configurarla de la siguiente manera:

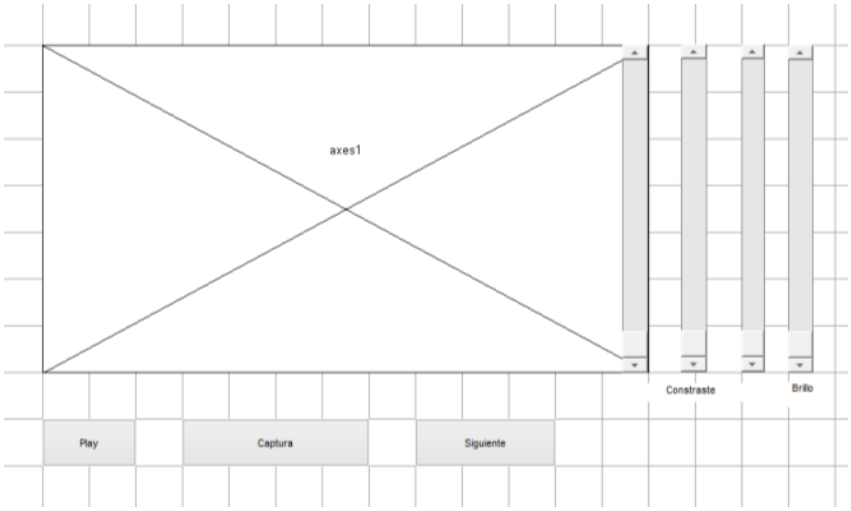

Figura 45. Apariencia final de la ventana de calibración, la diferencia visible es la inclusión de *sliders*.

A diferencia de la ventana de calibración inicial (Fig. 23), se le da al usuario la oportunidad de ajustar el contraste y brillo de la imagen, mediante el uso de una serie de *sliders.*

Para probar la utilidad de los cambios antes hechos se realizará una prueba en una ambiente en el cual la luminosidad no es favorable.

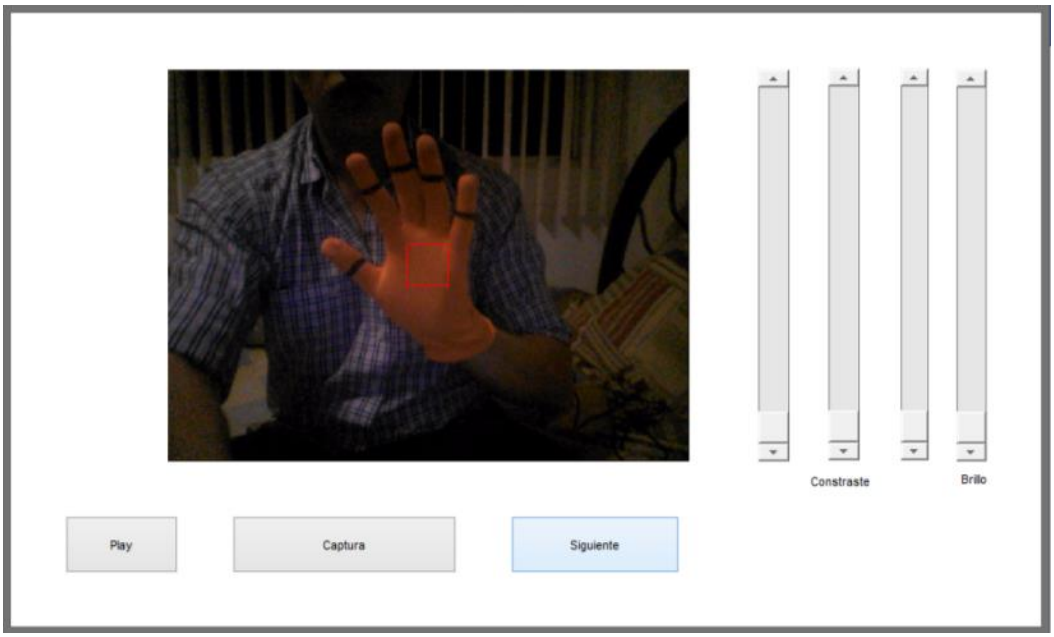

Figura 46. Ventana de calibración durante la prueba.

Como se puede observar en la ventana de calibración, Fig. 46, no es tan distinguible el guante del resto de la imagen lo cual aumenta la probabilidad de que los colores sean confundidos y no se logre con éxito el reconocimiento.

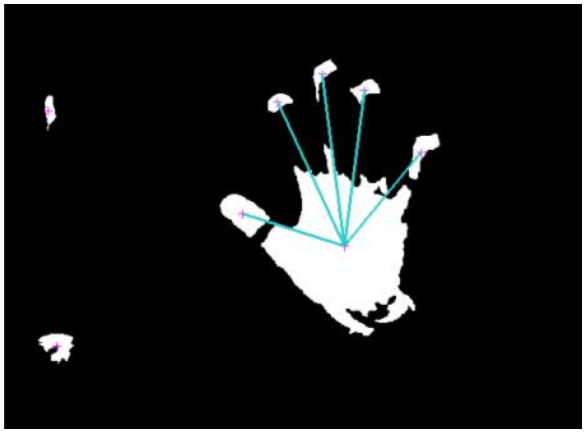

Figura 47. Imagen del reconocimiento de la mano.

Como se observa en la Fig. 47, existen zonas de la imagen que el software confunde con el color del guante, dichas zonas son identificadas como objetos, debido a los ajustes hechos en la programación del algoritmo no son reconocidas como dedos, sin embargo de haberse encontrado más cerca al correspondiente centroide de la mano si lo hubiese tomado como un dedo extra. Ahora se mostrará el efecto del contraste y brillo en el programa (Figuras 48 y 49)

Figura 48. Ventana de calibración con ajuste de contraste y brillo.

Como se muestra en la figura 48 se han modificado los valores de contraste en R, G y B, al mismo tiempo que se ha aumentado el brillo de la imagen, a simple vista puede notarse un mayor contraste entre el guante y otras zonas del fondo de la imagen lo cual favorece al algoritmo, igualmente puede resaltarse que el contraste en las tres coloraciones no es el mismo, esta condición favorece la decisión de que se haya dado dichos grados de libertad en el programa.

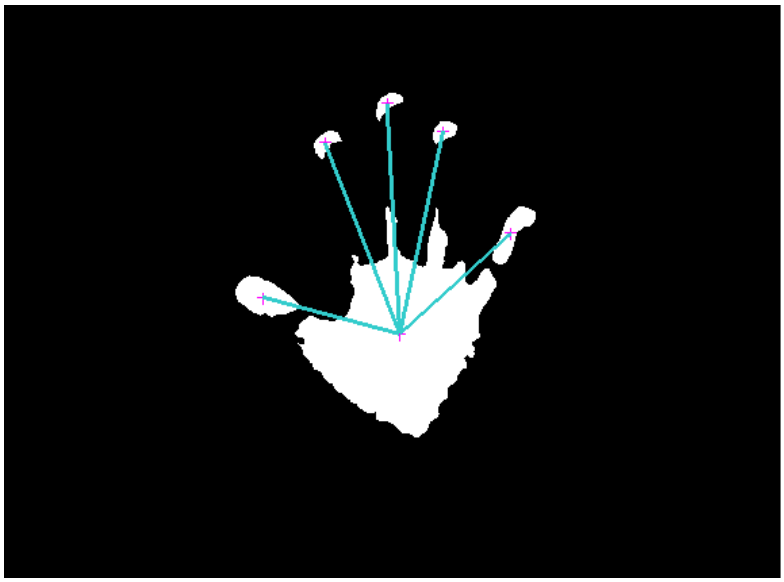

Figura 49. Imagen correspondiente al reconocimiento de la mano después de modificar contraste y brillo.

Si se comparan las imágenes 47 y 49 se podría pensar que en la primera es más reconocible la forma de la mano, lo cual es cierto y que bajo dicha condición los ajustes a la imagen no son efectivos, sin embargo en la imagen 48 se observa que no hay zonas ajenas a la mano, así que, si bien se ha vuelto más pequeño el rango del filtro, esto ha ayudado a que se reconozca mejor la zona de interés. Con las nuevas operaciones a las imágenes se ha aumentado la versatilidad del sistema, pues es capaz de adaptarse a variaciones de luz.

En este punto se ha realizado todo el procesamiento necesario, el siguiente paso es el diseño de las ventanas y la transición entre ellas.

DISEÑO Y TRANSICIÓN DE VENTANAS

Durante el desarrollo de éste tópico se deben respetar las consideraciones de diseño hechas en los capítulos anteriores, tanto por número de ventanas, número de botones por ventana, etc.

Las ventanas se mostrarán según el orden de aparición decidido:

Figura 50.Primera ventana, se considerará como una ventana de presentación del programa.

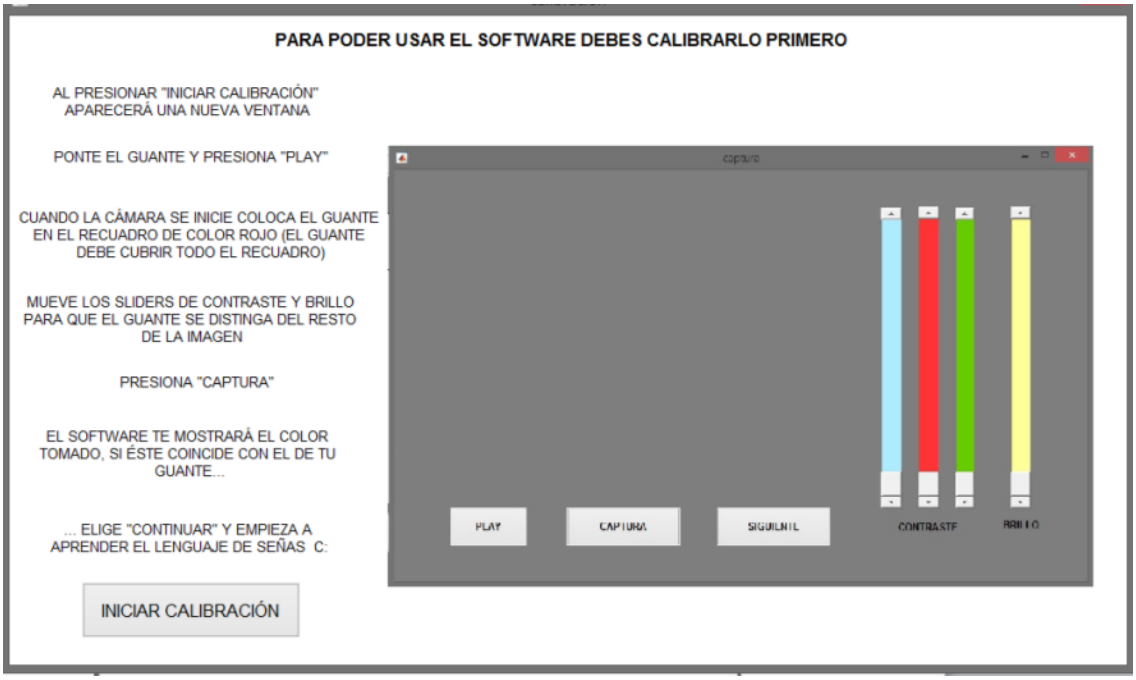

Figura 51. Segunda ventana, en ella se agregará una breve descripción del funcionamiento del programa.

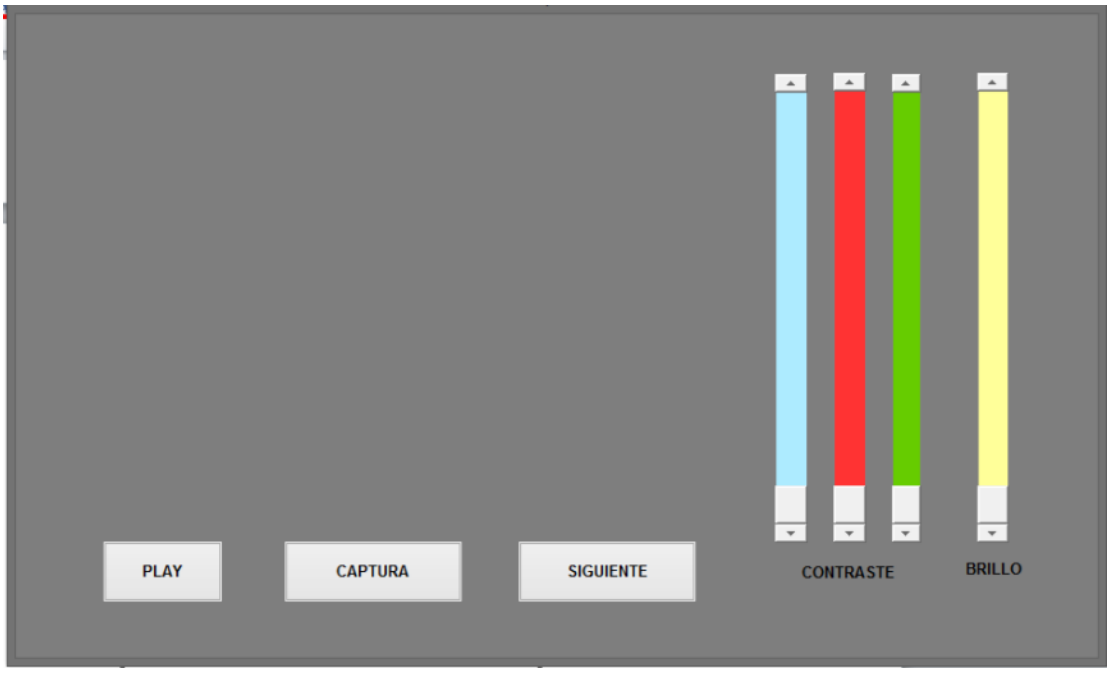

Figura 52. Ventana de calibración, esta es la tercera ventana en aparición.

Figura 53. Menú de señas, esta se considera la ventana principal pues a través de ella se navegará por el contenido del programa.

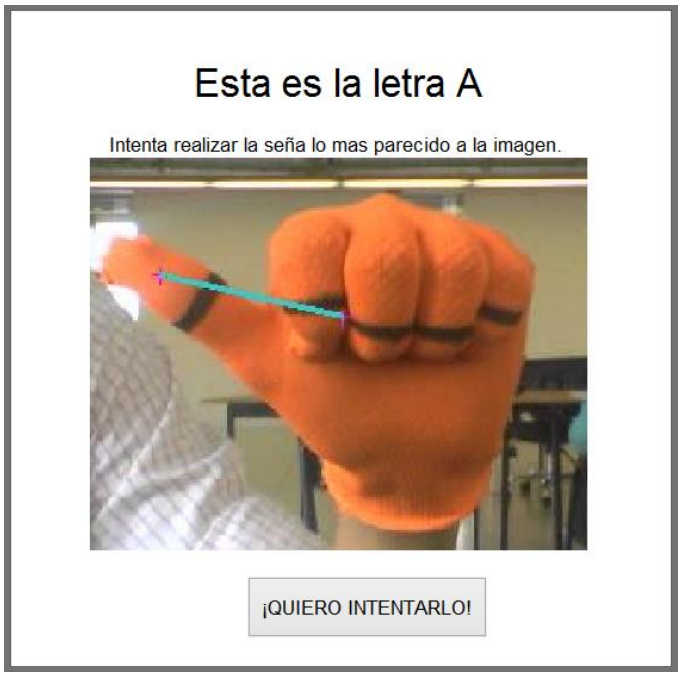

Figura 54. Al elegir el tipo de seña a aprender, o practicar, aparecerá una ventana que contendrá la imagen de la seña que el usuario debe reproducir.

Figura 55. Ventanas de retroalimentación, aparecerán de acuerdo al desempeño del usuario.

CAPÍTULO 5. PRUEBAS

Con el fin de evaluar el programa se desarrollará un protocolo de pruebas, este protocolo permitirá a algunos usuarios probar el programa, evaluar su desempeño con el aprendizaje del lenguaje de señas y con ello evaluar también al proyecto en cuestión.

El primer elemento a considerar será que el usuario de prueba no tenga conocimiento del lenguaje de señas, pues, como se mencionaba en los primeros capítulos del presente documento, el programa está principalmente dirigido a este sector.

El usuario no recibirá ayuda en la navegación del programa, como se ha pensado que el proyecto pueda distribuirse al público en general, no contará con el auxilio del desarrollador del mismo.

Se le pedirá al usuario que aprenda las señas correspondientes a los días de la semana, esto debido a que se trata de un vocabulario básico, además de que no son solamente señas estáticas a diferencia del abecedario, sino que deben cumplirse con una trayectoria en específico.

El tiempo durante el cual el usuario interactúe con el software es muy importante, se proponen de cinco a diez minutos para ello, aunque el tiempo exacto, que esté dentro de dicho intervalo, dependerá de ellos.

Culminado el tiempo de interacción con el software se tomarán cinco minutos para que el usuario dedique su atención a una actividad completamente diferente a la del lenguaje de señas, se propondrá la lectura de un texto, que el usuario navegue por cualquiera de sus sitios de internet favoritos, o que hagan cualquier otra actividad de su preferencia, esta etapa quedará oculta al usuario, pues no se quiere condicionar su atención.

Pasados los cinco minutos, se realizarán una serie de cuestionamientos relacionados con las señas estudiadas, es decir, se les pedirá que recreen dichas señas.

Se evaluará la retención de cada usuario de acuerdo a su desempeño y se asociará al éxito del software como herramienta de enseñanza.

Al culminar se pedirá al usuario que llene una encuesta en la cual se evaluarán los elementos visuales y el método empleado por el software para fungir como elemento para la enseñanza.

La apariencia de dicha encuesta es la siguiente:

Responde las preguntas según haya sido tu experiencia con el software

1.- ¿Crees que el programa te proporcionó las instrucciones suficientes para interactuar con el mismo?

2.- ¿Qué te pareció la apariencia de las ventanas del programa, te gustaron los colores y las ilustraciones?

3.- ¿El uso del guante te pareció incómodo?

4.- ¿Cómo dirías que fue tu experiencia con la transición de las ventanas y el cumplimiento de las señas?

El proceso será grabado en video para facilitar extraer la información importante y detallarla lo mejor posible.

Con el fin de evaluar al sistema bajo diversas condiciones se plantea realizar las pruebas en ambientes diferentes y bajo distintas condiciones, al aire libre, en un espacio cerrado iluminado con luz natural, y en espacio cerrado bajo la influencia de luz artificial. Además, las condiciones y detalles relacionados con cada prueba serán reportados en el siguiente capítulo.

CAPÍTULO 6. RESULTADOS

Culminadas las pruebas con seis personas, se reportarán aquí lo ocurrido en ellas, se reportará el nombre, el número de aciertos y la encuesta correspondiente, su desempeño y los pormenores de cada prueba serán igualmente descritos.

Nombre del usuario: Usuario prueba 1

Número de aciertos: 6/6

Descripción:

La prueba se desarrolló en al aire libre, Fig. 56, el tamaño del guante es mayor al ideal para el usuario por un pequeño margen aunque no resultó ser un factor determinante para el desempeño en general, se realizó un único proceso de calibración el cual fue suficiente para que se pudiesen identificar todas las señas, el usuario se adaptó rápidamente al software pues no presentó titubeos en la navegación a lo largo de la prueba, la cual completó en nueve minutos.

Figura 56. Imagen que muestra el entorno en que se realizó la prueba uno.

Encuesta:

Responde las preguntas según haya sido tu experiencia con el software

1.- ¿Crees que el programa te proporcionó las instrucciones suficientes para interactuar con el mismo?

Sí, considero que son instrucciones concretas y suficientes para poder interactuar con él.

2.- ¿Qué te pareció la apariencia de las ventanas del programa, te gustaron los colores y las ilustraciones?

Las ventanas me parecen un poco rústicas, sin embargo en lo personal creo que ello no influye con el objetivo del software. Por otro lado, las imágenes me gustan y me parece que son colores adecuados y amigables visualmente hablando.

3.- ¿El uso del guante te pareció incómodo?

No, considero que es una buena idea, es confortable y funcional.

4.- ¿Cómo dirías que fue tu experiencia con la transición de las ventanas y el cumplimiento de las señas?

Considero que es adecuada, sin embargo podría ser más práctico que todo el proceso de selección de la seña y la ejecución de la misma pudiera desarrollarse en una misma ventana para una mayor comodidad y así no tener que regresar al menú en cada una de ellas.

Nombre del usuario: Usuario prueba 2

Número de aciertos: 5/6

Descripción:

La prueba se desarrolló en un espacio cerrado frente a una ventana por la cual entraba luz natural, el guante es del tamaño correcto para este usuario, se realizaron dos procesos de calibración debido a que por la cantidad de luz el software confundía partes de la coloración de la piel del usuario, debido a que había un retardo entre la transición de algunas ventanas hubo un par de momentos de confusión aunque adquirió fluidez en momentos posteriores, la prueba duró ocho minutos y medio.

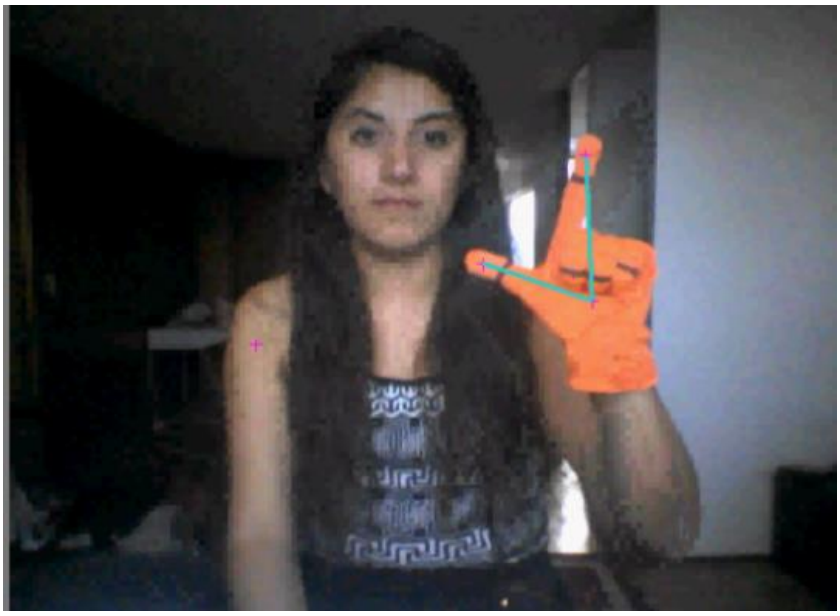

Figura 57. Imagen que muestra el entorno en que se realizó la prueba dos.

Encuesta:

Responde las preguntas según haya sido tu experiencia con el software

1.- ¿Crees que el programa te proporcionó las instrucciones suficientes para interactuar con el mismo?

Sí, tanto para comenzar la calibración como en las ventanas de evaluación la información es suficiente

2.- ¿Qué te pareció la apariencia de las ventanas del programa, te gustaron los colores y las ilustraciones?

Al usarlo usarlo por primera vez la transición de ventanas me conflictuó un poco, no me quedaba del todo claro si debía esperar o ya había terminado el ejercicio.
Sí, las ilustraciones son amigables

3.- ¿El uso del guante te pareció incómodo?
Sí, por el tamaño y al usarlo en la mano derecha y ser diestra no podía usar el touchpad.

4.- ¿Cómo dirías que fue tu experiencia con la transición de las ventanas y el cumplimiento de las señas?

Un poco lenta. En algunas ocasiones no me reconocía la seña al primer intento, pero al seguir intentando notas que no estás cumpliendo realmente, tal vez podría ser de ayuda volver a ver la seña.

Nombre del usuario: Usuario prueba 3

Número de aciertos: 5/6

Descripción:

La prueba se realizó en un lugar donde se había probado al software con anterioridad, se necesitó de calibrar dos veces, el desempeño se realizó de manera fluida, aunque hubo un detalle en la transición de las ventanas debido a que se cerró de manera intempestiva una cámara que hacía uso del video, aunque igual pudo corregirse por el usuario para poder seguir con la prueba, misma que tuvo una duración de once minutos.

Figura 58. Imagen que muestra el entorno en que se realizó la prueba tres.

Encuesta:

Responde las preguntas según haya sido tu experiencia con el software 1.- ¿Crees que el programa te proporcionó las instrucciones suficientes para interactuar con el mismo? Sí, las instrucciones fueron claras, quizá solo cambiar el botón de Play a Encender cámara o semejante por las personas que no hablan nada de inglés.

2.- ¿Qué te pareció la apariencia de las ventanas del programa, te gustaron los colores y las ilustraciones?

Muy ilustrativas y amigables para la vista.

3.- ¿El uso del guante te pareció incómodo?

Para nada, incluso me lo quisiera quedar.

4.- ¿Cómo dirías que fue tu experiencia con la transición de las ventanas y el cumplimiento de las señas?

Las ventanas se abren con cierto retardo pero son completamente funcionales. Hubo un día de la semana en el que se me detectó la seña sin haberla hecho pero los demás se detectaron sin problema alguno. Al momento de hacer el círculo, a veces la referencia central del guante se perdía, dependiendo de la inclinación del guante pero la trayectoria se seguía con fidelidad cuando se detectaba.

Nombre del usuario: Usuario prueba 4

Número de aciertos: 6/6

Descripción:

La prueba se desarrolló en el mismo ambiente que en la prueba anterior, aunque debido al estado del tiempo la luz natural jugaba una mala pasada, el tamaño del guante es demasiado grande para el usuario pero se decidió continuar con la prueba para observar qué tan robusto es el sistema, se realizó solo un proceso de calibración, bajo las condiciones anteriores la prueba resultó más complicada para este usuario sin embargo el desempeño fue bastante bueno pues la prueba se completó en nueve minutos.

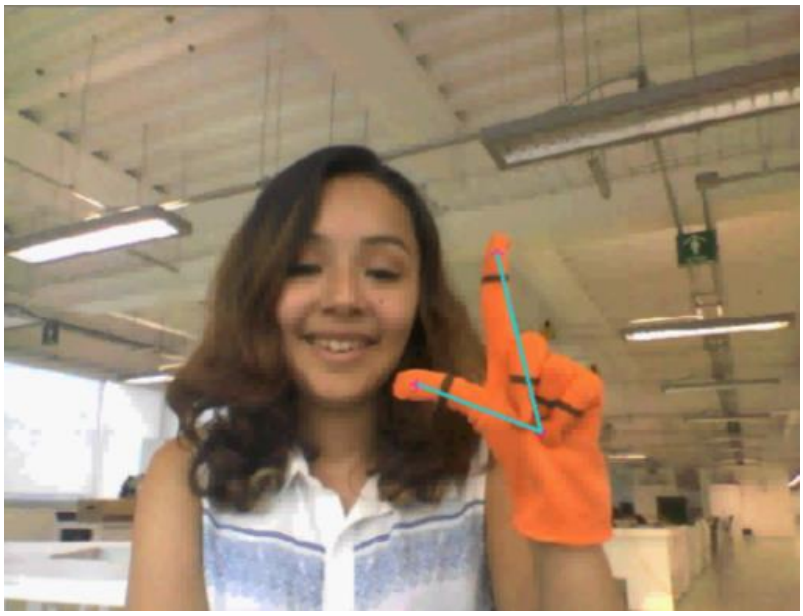

Figura 59. Imagen que muestra el entorno en que se realizó la prueba cuatro.

Encuesta:

Responde las preguntas según haya sido tu experiencia con el software

1.- ¿Crees que el programa te proporcionó las instrucciones suficientes para interactuar con el mismo?

Sí, es muy entendible, las indicaciones son suficientemente claras.

2.- ¿Qué te pareció la apariencia de las ventanas del programa, te gustaron los colores y las ilustraciones?

Las ilustraciones me gustaron, se ven lindas y congruentes entre sí y proporcionan retroalimentación agradable de acuerdo al desempeño.

3.- ¿El uso del guante te pareció incómodo?

Sí es cómodo y ya que no se puede utilizar e touchpad entonces promueve ejercitar el otro hemisferio del cerebro o se puede usar mouse.

4.- ¿Cómo dirías que fue tu experiencia con la transición de las ventanas y el cumplimiento de las señas?

Es muy intuitivo por lo que logré utilizar fácilmente el programa y aprender las señas.

Nombre del usuario: Usuario prueba 5

Número de aciertos: 5/6

Descripción:

La prueba se realizó por por la noche en un ambiente cerrado por lo que se dependía de la luz artificial para una buena iluminación, el sitio correcto fue difícil de determinar debido a que había lámparas en varios puntos de la habitación lo cual causaba sombras indeseables, lo cual causó que el usuario calibrara un total de tres veces reflejándose en el tiempo para completar la prueba, dando un total de nueve minutos.

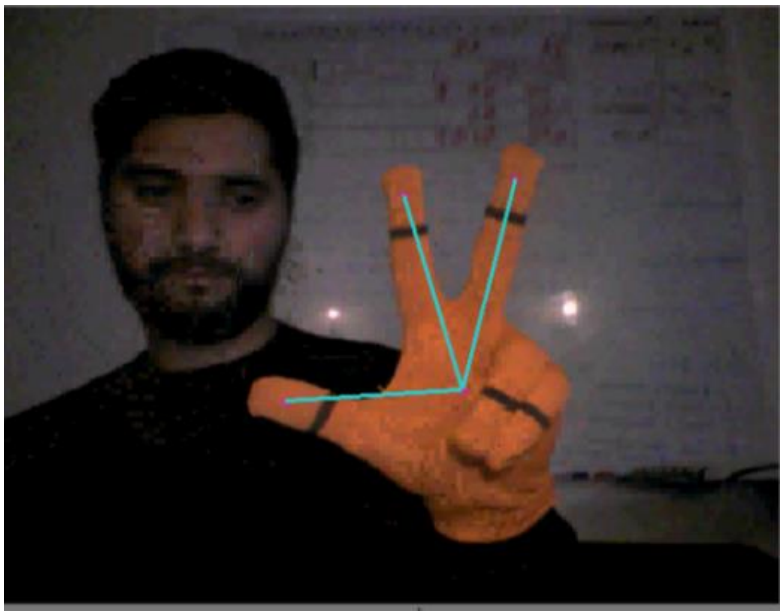

Figura 60. Imagen que muestra el entorno en que se realizó la prueba cinco.

Cuestionario:

Responde las preguntas según haya sido tu experiencia con el software

1.- ¿Crees que el programa te proporcionó las instrucciones suficientes para interactuar con el mismo?

Sí, pero apoyos gráficos ayudarían a que fuera más claro.

2.- ¿Qué te pareció la apariencia de las ventanas del programa, te gustaron los colores y las ilustraciones?

Sí, las ilustraciones son muy alegres y la interfaz intuitiva.

3.- ¿El uso del guante te pareció incómodo?

No

4.- ¿Cómo dirías que fue tu experiencia con la transición de las ventanas y el cumplimiento de las señas?

Fue un poco lento, pero una vez que te acostumbras a la interfaz fue mucho más rápido

Nombre del usuario: Usuario prueba 6

Número de aciertos: 6/6

Descripción:

La prueba se realizó bajo las mismas condiciones que la del usuario anterior, teniendo el inconveniente de la luz y las sombras, a pesar de ello bastó con una calibración para poder interactuar con todo el programa, misma que fue bastante fluida pues no hubo confusión con la transición de las ventanas y se cumplió con el aprendizaje de las seis señas en un total de siete minutos.

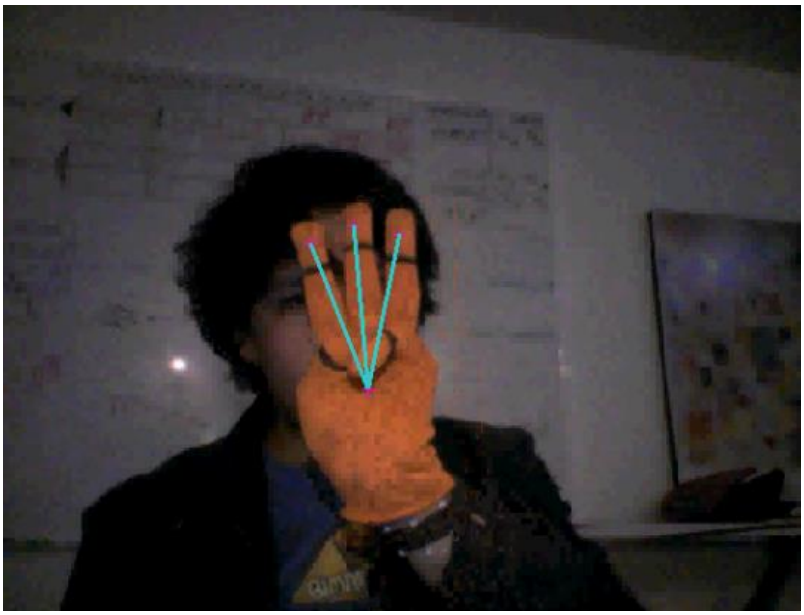

Figura 61. Imagen que muestra el entorno en que se realizó la prueba seis.

Encuesta:

Responde las preguntas según haya sido tu experiencia con el software

1.- ¿Crees que el programa te proporcionó las instrucciones suficientes para interactuar con el mismo?

Sí.

2.- ¿Qué te pareció la apariencia de las ventanas del programa, te gustaron los colores y las ilustraciones?

Sí, fue suficiente.

3.- ¿El uso del guante te pareció incómodo?

Jamás.

4.- ¿Cómo dirías que fue tu experiencia con la transición de las ventanas y el cumplimiento de las señas?

Fluido pero hay espacio para marcar transiciones con más capas.

ANÁLISIS DE RESULTADOS.

De los resultados se pueden destacar diversos puntos, el primero de ellos es el resultado promedio, todos los usuarios prueba oscilaron entre cinco y seis aciertos, siendo seis el máximo puntaje, a reserva de contar con una población más grande, en este punto se puede hablar de una efectividad de enseñanza-aprendizaje muy alta teniendo un puntaje promedio de 5.5/6, o sea de 91.6%, cumpliendo con el objetivo del software de enseñar.

Ahora, citando los resultados de la encuesta:

A cinco personas las instrucciones les parecieron suficientemente claras para la navegación dentro del programa, uno de ellos también las consideró claras pero con un margen de mejora apoyándose de más ilustraciones; retomando algunos puntos de los objetivos de diseño marcados en el capítulo segundo se cumplen los parámetros de ser intuitivo, eficiente y con gran facilidad de navegación por la GUI.

La segunda pregunta estaba orientada a la apariencia del GUI, con lo que se consiguieron resultados positivos, pues a los seis encuestados les pareció visualmente agradable la inclusión de ilustraciones, aunque un par de aspectos valen la pena de ser citados, el primero de ellos establece que la apariencia de las ventanas es un tanto “rústica” y el segundo que el tiempo entre la aparición entre una ventana es en ocasiones muy alto, lo cual llegó a confundir a veces al usuario, este último aunque va más orientado a desempeño es lo suficientemente valioso para resaltarlo.

La tercera pregunta cuestiona el uso del único elemento externo implementado en el sistema: el guante, a cinco personas no les resultó incómodo, dejando así a una persona inconforme, misma que argumenta dificultad para manejar el touchpad del dispositivo; este punto resalta que el guante fue seleccionado por coloración dejando a un lado las texturas, puesto que era lo importante en la funcionalidad y la interacción con el algoritmo propuesto.

La cuarta pregunta pide al usuario evaluar la transición de las ventanas y su experiencia con el reconocimiento de las señas, en ella se recalca el retardo entre la transición de las ventanas aunque la gran mayoría finalmente sorteaba los pequeños detalles ya sea recalibrando o aprendiendo la forma en que se tenía que reproducir la seña.

Como desarrollador, es importante recalcar algunos puntos importantes, debido a que a lo largo de las puebas se presentaron situaciones que podían afectar el desempeño del sistema y que se sortearon o no de manera adecuada:

Las pruebas se realizaron en ambientes diferentes, al aire libre, con la ayuda de luz artificial y de luz natural, aspecto destacable en cuanto a la capacidad de adaptabilidad del sistema.

Ningún video rebasó los quince minutos, siendo el máximo de trece minutos, aunque se había establecido un rango de entre cinco y diez minutos para la prueba no se contemplaba el tiempo que el usuario necesitara para familiarizarse con el software además del tiempo para lograr una calibración exitosa, entonces, se pude establecer que se necesitaron menos de trece minutos para que los usuarios aprendieran a usar el software y estudiaran las seis señas correspondientes a los días de la semana. Esto de nuevo se puede traducir en una alta efectividad correspondiente al punto de la enseñanza del sistema.

Todos los usuarios pudieron calibrar de manera exitosa el programa, unos más favorecidos por la luz del ambiente que otros lo cual se traducía en un menor número de intentos pero al final la calibración no representó un problema mayor.

Evidentemente, el tiempo entre la transición de las ventanas es un problema, sobre todo cuando después de cumplir con la seña estática correspondía realizar el cumplimiento de la trayectoria.

Con todos los detalles de las pruebas señalados, se tienen los elementos suficientes para poder elaborar conclusiones referentes al proyecto.

CAPÍTULO 7. CONCLUSIONES

Al final de todo el proceso descrito en los capítulos anteriores, se tiene un software completamente funcional, el cual puede ejecutarse en un equipo de cómputo que tenga los siguientes requerimientos mínimos: sistema operativo Windows 8, 4 GB de memoria RAM, procesador de tercera generación y cámara de video de 1.3 Megapixeles como características mínimas (especificaciones del equipo en el cual se desarrolló).

Este software se une al grupo de alternativas tecnológicas descritas en el capítulo primero, sin embargo posee características que lo hacen distinguirse del resto, una de ellas es el enfoque en el cual se sustenta, primero destacando que la enseñanza es la piedra angular para lograr que el lenguaje de señas tome más fuerza y cada vez más personas lo conozcan y, más importante aún, lo aprendan. Este aspecto fortalece las bases sobre las cuales se erigen los traductores desarrollados en los últimos años, pero, debe recordarse que también se hablaba de que no es la única manera de lograr la comunicación entre un señante y una persona que hace uso del lenguaje oral, pues estos logran que el canal de comunicación se sostenga vía oral, sin embargo dicha comunicación también puede sostenerse mediante la vía visogestual, es decir que ambas personas se comuniquen utilizando el lenguaje de señas, esto mirando hacia el principal objetivo que es el de la inclusión de las personas con discapacidad auditiva en una sociedad basada en la comunicación oral.

Entonces, ya que se plantea a la enseñanza como el elemento primordial, debe tenerse en cuenta que es necesaria una enseñanza de calidad, es decir, que el software aporte un método efectivo para que las personas aprendan el lenguaje de señas, este punto queda demostrado con las pruebas realizadas en el capítulo anterior, pues la efectividad de enseñanza-aprendizaje es bastante alta.

Este último dato aunado a la repetitividad, o lo que es lo mismo con una práctica constante da como resultado una buena memoria muscular y por lo tanto un aprendizaje más fluido y completo.

Se ha conseguido también tener un sistema robusto, debido a que se le dotó de adaptabilidad a condiciones ambientales, específicamente a variaciones de iluminación, permitiendo que con solo tomar un par de consideraciones de uso (posición con respecto a la luz predominante y cambio de contraste e iluminación) se pueda interactuar de forma efectiva en casi cualquier lugar a cualquier hora del día.

Con el algoritmo planteado se permitió aportar al sistema con retroalimentación para el usuario, aspecto clave para la enseñanza, basándose en nada más que una cámara de video y un hardware externo simple y muy asequible como lo es un guante de color.

Se diseñó al software para que tuviera una buena interacción con el usuario, lo que en su momento se señaló como “amigable”, dotándolo de una navegación entre ventanas simple e intuitiva y de procesos cortos y bien instruidos dando como resultado una buena GUI.

El software cuenta con un considerable número de señas como base para esta primera aproximación, con un total de setenta, total que puede expandirse con cierta facilidad, siempre y cuando las señas cumplan con las condiciones establecidas en los capítulos anteriores.

Un punto recurrente a lo largo de las pruebas tenía que ver con la rapidez del programa, destacando la correspondiente a la transición de las ventanas, dicho aspecto no escapa a la vista del desarrollo del proyecto, sin embargo este más unos cuantos relacionados con el diseño de las ventanas y los gráficos se consideran detalles que si bien no son prioridad debido al objetivo de la presente tesis se consideran importantes para próximas etapas cuando se tome como objetivo el volverlo un producto competitivo y líder en su ramo. Estos puntos serán desarrollados en el siguiente capítulo: “Trabajo futuro”.

CAPÍTULO 8. TRABAJO FUTURO

La primer actividad que se podría realizar es obtener el ejecutable del programa, se espera que al hacerlo el software funcione más rápido dando como resultado un programa más dinámico, reduciendo así el aspecto de la lentitud entre algunas transiciones de las ventanas, la razón por la cual un ejecutable es más rápido es que hace uso solamente de los elementos necesarios para que el programa opere, quitando otros procesos que el programa en el entorno de desarrollo ejecuta y que al hacerlo hace más lenta su ejecución.

Se ha demostrado que la etapa de calibración es sumamente útil y en muchos casos hasta indispensable para el correcto funcionamiento del algoritmo, sin embargo depende de la destreza y la experiencia del usuario en cuestión para que rinda o no mejores resultados, un punto a mejorar es el de hacer de esta etapa una operación automática que haga uso de la capacidad de cálculo de la computadora para prescindir de la calibración manual.

Se seguirán realizando pruebas con nuevos usuarios y otros equipos con el fin de recopilar sugerencias y de evaluar desempeño o en dado caso resolver problemas de compatibilidad, siendo este aspecto el motivo del siguiente punto a desarrollar.

Con la idea de hacer del software un sistema más versátil se plantea migrarlo a software de desarrollo open source, además de buscar que sea compatible con un mayor número de dispositivos, como celulares y tabletas electrónicas, así la idea de llegar a un mayor número de personas que puedan aprender el lenguaje de señas se ve cada vez más viable.

Se ha hecho hincapié en que para lograr la inclusión de las personas con discapacidad auditiva en la sociedad es necesario que cada vez más personas conozcan el Lenguaje de Señas Mexicano, pero para lograrlo es necesario de una buena difusión y distribución de este tipo de herramientas, es por eso que se buscará promocionar este proyecto en congresos nacionales e internacionales y demás eventos de alto impacto para que se pueda lograr el cometido de que más personas puedan comunicarse haciendo uso de este lenguaje.

ANEXO. CÓDIGO

---------- CÓDIGO DE PÁGINA INICIAL----------

```
function varargout = principal(varargin)
%Inicialización de la función, no se debe editar
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @principal_OpeningFcn, ...
                   'gui_OutputFcn',  @principal_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% Finalización de la función


% --- Esta función se ejecuta cuando la ventana está visible.
function principal_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);
%Se declara la variable global "Letras" que contiene a la base de datos
global Letras;
Letras=zeros(70,16);
%El término 16 es el error
Letras(1,:)=[0,10,0,0,0,0,0,0,0,0,0,0,0,0,0,9];
Letras(2,:)=[115,0,95,85,75,0,0,0,0,0,0,0,0,0,0,15];
Letras(3,:)=[0,45,5,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(4,:)=[0,83,23,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(5,:)=[125,0,105,95,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(6,:)=[0,85,18,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(7,:)=[0,93,22,8,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(8,:)=[120,0,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(9,:)=[0,325,10,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(10,:)=[0,80,10,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(11,:)=[0,255,275,300,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(12,:)=[0,270,300,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(13,:)=[0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(14,:)=[0,80,30,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(15,:)=[0,340,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(16,:)=[0,83,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(17,:)=[0,65,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(18,:)=[0,90,80,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(19,:)=[0,100,75,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(20,:)=[110,0,90,70,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(21,:)=[0,315,355,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(22,:)=[0,90,340,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(23,:)=[0,80,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(24,:)=[0,80,10,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(25,:)=[0,90,75,20,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(26,:)=[0,110,90,70,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(27,:)=[120,0,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(28,:)=[0,100,75,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(29,:)=[0,80,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(30,:)=[0,80,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(31,:)=[0,90,80,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(32,:)=[110,0,90,70,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(33,:)=[137,0,101,83,58,0,0,0,0,0,0,0,0,0,0,15];
```

```
Letras(34,:)=[138,0,106,85,62,22,0,0,0,0,0,0,0,0,0,15];
Letras(35,:)=[0,90,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(36,:)=[0,85,18,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(37,:)=[0,93,22,8,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(38,:)=[0,90,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(39,:)=[0,85,18,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(40,:)=[0,93,22,8,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(41,:)=[0,325,17,350,1,0,0,0,0,0,0,0,0,0,0,15];
Letras(42,:)=[0,80,10,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(43,:)=[0,80,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(44,:)=[120,10,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(45,:)=[0,85,18,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(46,:)=[0,90,80,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(47,:)=[0,42,2,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(48,:)=[0,98,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(49,:)=[0,35,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(50,:)=[128,0,102,86,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(51,:)=[125,0,102,88,67,23,0,0,0,0,0,0,0,0,0,15];
Letras(52,:)=[0,85,18,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(53,:)=[125,0,102,88,67,23,0,0,0,0,0,0,0,0,0,15];
Letras(54,:)=[0,17,3,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(55,:)=[0,85,18,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(56,:)=[0,88,43,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(57,:)=[0,104,86,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(58,:)=[0,340,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(59,:)=[166,0,11,59,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(60,:)=[116,0,87,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(61,:)=[0,77,21,3,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(62,:)=[0,68,331,17,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(63,:)=[0,68,331,17,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(64,:)=[0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(65,:)=[0,58,318,346,21,2,0,0,0,0,0,0,0,0,0,15];
Letras(66,:)=[0,45,5,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(67,:)=[153,0,110,75,34,0,0,0,0,0,0,0,0,0,0,15];
Letras(68,:)=[0,68,331,17,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(69,:)=[0,80,30,0,0,0,0,0,0,0,0,0,0,0,0,15];
Letras(70,:)=[0,343,0,0,0,0,0,0,0,0,0,0,0,0,0,15];


%Las salidas de esta función aparecen en las líneas de comandos
function varargout = principal_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
%Se crea el objeto de la imagen y se despliega
unam1=imread('unam1.jpg');
imshow(unam1)


%Esta función se ejecuta cuando se presiona el botón de la pantalla
function pushbutton1_Callback(hObject, eventdata, handles)
%Si se presiona el botón llama a la ventana llamada “calibración”
calibracion;
```

---------- **CÓDIGO DE INSTRUCCIONES** ----------

```
function varargout = calibracion(varargin)
%Inicialización de la función, no se debe editar
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @calibracion_OpeningFcn, ...
                   'gui_OutputFcn',  @calibracion_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
%Finalización de la función.

% Esta función se ejecuta cuando la ventana es visible.
function calibracion_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);



%Las salidas de esta función aparecen en las líneas de comandos
function varargout = calibracion_OutputFcn(hObject, eventdata, handles)
%Se cierra la ventana anterior
close(principal);
varargout{1} = handles.output;
%Se carga la imagen y se despliega
unam1=imread('CapCalib.jpg');
imshow(unam1)


%Esta función se ejecuta cuando se presiona el botón de la ventana
function pushbutton1_Callback(hObject, eventdata, handles)
%Cuando se presiona el botón se llama a la siguiente ventana llamada captura
captura;
```

---------- **CÓDIGO DE CALIBRACIÓN** ----------

```
function varargout = captura(varargin)
%Declaración de la función, no se debe editar
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @captura_OpeningFcn, ...
                   'gui_OutputFcn',  @captura_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% Finalización de la función

% Se ejecuta justo antes de que la ventana sea visible
function captura_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
%Se cierra la ventana anterior
close(calibracion)


%Las salidas de esta función aparecen en las líneas de comandos
function varargout = captura_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

%Esta función se ejecuta cuando se presiona el botón llamado captura
function CAPTURA_Callback(hObject, eventdata, handles)
%Se declaran las variables globales
global bandera data vid promedio1 promedio2 promedio3
bandera=1;
hcontadj = vision.ContrastAdjuster;
diff_ima = data(215:265,295:345,1:3);
%Se realiza el tratamiento a la imagen
foto_R=diff_ima(:,:,1);
foto_G=diff_ima(:,:,2);
foto_B=diff_ima(:,:,3);
h2=fspecial('average',[10,10]);
media2R=imfilter(foto_R,h2);
media2G=imfilter(foto_G,h2);
media2B=imfilter(foto_B,h2);
foto2(:,:,1)=media2R;
foto2(:,:,2)=media2G;
foto2(:,:,3)=media2B;

muestra1=0;
muestra2=0;
muestra3=0;
%Se recorre la imagen obteniendo el promedio de RGB
for i=1:50
    for j=1:50
         intc1 = int32(foto2(i,j,1))+muestra1;
         intc2 = int32(foto2(i,j,2))+muestra2;
         intc3 = int32(foto2(i,j,3))+muestra3;
         muestra1=intc1;
         muestra2=intc2;
         muestra3=intc3;

    end
end
```

```
promedio1=intc1/2500;
promedio2=intc2/2500;
promedio3=intc3/2500;
%Se detiene el objeto de video y se borra lo que haya guardado la variable
stop(vid)
flushdata(vid);
imshow(foto2)

% Esta función se ejecuta cuando se presiona el botón "Play"
function pushbutton4_Callback(hObject, eventdata, handles)
% Se declaran las variables globales
global bandera cuadro data vid valor valorS1 valorS2 valorS3 valorS4
bandera=0;
vid=videoinput('winvideo',1,'MJPG_640x480');
% Se configura las opciones de adquision de video
set(vid, 'FramesPerTrigger', Inf);
set(vid, 'ReturnedColorspace', 'rgb')
vid.FrameGrabInterval = 4;
valorS1=0.01;
valorS2=0.01;
valorS3=0.01;
valorS4=0.01;
start(vid)
contadorFlush=0;
while(bandera==0)
% se toma una snapshot del stream y se la almacena en data para trabajar mas facil
data = getsnapshot(vid);
%Se aplican las correcciones de contraste y brillo realizadas por el usuario
data=imadjust(data,[valorS1 valorS2 valorS3;1 1 1],[valorS4 valorS4 valorS4;1 1 1]);
data2=fliplr(data);
imshow(data2)
ancho=295;
alto=215;
hold on
%este es un bucle para encerrar el objeto rojo en un rectangulo y una cruz en el
%centroide(solo es programacion basica de matlab)
rectangle('Position',[ancho alto 50 50],'EdgeColor','r')
hold off
contadorFlush=contadorFlush+1;
%Para que la variable no se sature se libera cada 80 ciclos
if(contadorFlush==80)
    flushdata(vid);
    contadorFlush=0;
end
end
cuadro=getsnapshot(vid);
% aqui terminan los 2 bucles
% detenemos la CAPTURA
stop(vid);
%FLUSHDATA remueve la imagen del motor de adquisicion y la almacena en el buffer
flushdata(vid);
% borramos todo(como en cualquier programa)

% Esta función se ejecuta cuando se presiona el botón "siguiente"
function siguiente_Callback(hObject, eventdata, handles)
%Se llama la siguiente ventana llamada "Menú"
Menu



% Esta función se ejecuta cuando se mueve el slider1
function slider1_Callback(hObject, eventdata, handles)
global valorS1
valorS1=get(hObject,'Value');
```

```
%Esta función posee la apariencia del slider1
function slider1_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% Esta función se ejecuta cuando se mueve el slider2
function slider2_Callback(hObject, eventdata, handles)
global valorS2
valorS2=get(hObject,'Value');

%Esta función posee la apariencia del slider2
function slider2_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% Esta función se ejecuta cuando se mueve el slider3
function slider3_Callback(hObject, eventdata, handles)
global valorS3
valorS3=get(hObject,'Value');


%Esta función posee la apariencia del slider3
function slider3_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% Esta función se ejecuta cuando se mueve el slider4
function slider4_Callback(hObject, eventdata, handles)
global valorS4
valorS4=get(hObject,'Value');

%Esta función posee la apariencia del slider4
function slider4_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

---------- **CÓDIGO DE MENÚ** ----------

```
function varargout = Menu(varargin)
% Inicialización de la primer función
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Menu_OpeningFcn, ...
                   'gui_OutputFcn',  @Menu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% Fin de la primer función


% Esta función se ejecuta antes de que la ventana sea visible
function Menu_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
close(captura);


% Las salidas de esta función se muestran directamente en la línea de comandos
function varargout = Menu_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
%Se cargan las imágenes y se despliegan en cada cuadro de la ventana
axes(handles.axes1)
background = imread('abcdario.jpg');
axis off;
imshow(background);
%%%%%%%%%%%%%%%%%%%%
axes(handles.axes3)
background = imread('numeros.jpg');
axis off;
imshow(background);
%%%%%%%%%%%%%%%%%%%
axes(handles.axes2)
background = imread('dias.jpg');
axis off;
imshow(background);
%%%%%%%%%%%%%%%%%%%
axes(handles.axes4)
background = imread('tiempo.jpg');
axis off;
imshow(background);
%%%%%%%%%%%%%%%%%%%%
axes(handles.axes5)
background = imread('preguntas.jpg');
axis off;
imshow(background);
%%%%%%%%%%%%%%%%%%%
axes(handles.axes6)
background = imread('respuesta.jpg');
axis off;
imshow(background);
%%%%%%%%%%%%%%%%%%%
axes(handles.axes7)
background = imread('normas.jpg');
axis off;
```

```
imshow(background);
%%%%%%%%%%%%%%%%%%%%
axes(handles.axes8)
background = imread('verbos.jpg');
axis off;
imshow(background);
%%%%%%%%%%%%%%%%%%%%
axes(handles.axes9)
background = imread('comida.jpg');
axis off;
imshow(background);
%%%%%%%%%%%%%%%%%%%%
axes(handles.axes10)
background = imread('ropas.jpg');
axis off;
imshow(background);
%%%%%%%%%%%%%%%%%%%%
axes(handles.axes11)
background = imread('hogar.jpg');
axis off;
imshow(background);
%%%%%%%%%%%%%%%%%%%%
axes(handles.axes12)
background = imread('otros.jpg');
axis off;
imshow(background);

% Esta función se ejecuta cuando se presione el botón “Abcdario”
function pushbutton1_Callback(hObject, eventdata, handles)
FigA;


% Esta función se ejecuta cuando se presiona el botón “Días de la semana”
function pushbutton2_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=71
transicionVentana=71;
FigDinamicas;

% Esta función se ejecuta cuando se presiona el botón “Números”
function pushbutton3_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=72
transicionVentana=72;
FigDinamicas;

% Esta función se ejecuta cuando se presiona el botón “Regresar a calibración”
function pushbutton4_Callback(hObject, eventdata, handles)
captura;


% Esta función se ejecuta cuando se presiona el botón “Preguntas”
function pushbutton5_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=73
transicionVentana=73;
FigDinamicas;

% Esta función se ejecuta cuando se presiona el botón “Respuestas”
function pushbutton6_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=74
transicionVentana=74;
FigDinamicas;

% Esta función se ejecuta cuando se presiona el botón “Normas de cortesía”
function pushbutton7_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=75
```

```
transicionVentana=75;
FigDinamicas;

% Esta función se ejecuta cuando se presiona el botón "Verbos"
function pushbutton8_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=76
transicionVentana=76;
FigDinamicas;

% Esta función se ejecuta cuando se presiona el botón "Comida"
function pushbutton9_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=77
transicionVentana=77;
FigDinamicas;

% Esta función se ejecuta cuando se presiona el botón "Ropa"
function pushbutton10_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=78
transicionVentana=78;
FigDinamicas;

% Esta función se ejecuta cuando se presiona el botón "Hogar"
function pushbutton11_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=79
transicionVentana=79;
FigDinamicas;

% Esta función se ejecuta cuando se presiona el botón "Otros"
function pushbutton12_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=80
transicionVentana=80;
FigDinamicas;

% Esta función se ejecuta cuando se presiona el botón "Tiempo"
function pushbutton13_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=81
transicionVentana=81;
FigDinamicas;
```

---------- **CÓDIGO DE VENTANA CON SEÑA** ----------

```
function varargout = FigLunes(varargin)
% Inicialización de la primer función
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @FigLunes_OpeningFcn, ...
                   'gui_OutputFcn',  @FigLunes_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% Finalización de la primera función


% Esta función se ejecuta cuando la ventana es visible
function FigLunes_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
close(MenuDias)



% Las salidas de esta función van directo a la línea de comandos
function varargout = FigLunes_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
axes(handles.axes1)
background = imread('Lunes.jpg');
axis off;
imshow(background);


% Esta función se ejecuta cuando se presiona el botón "Quiero intentarlo"
function pushbutton1_Callback(hObject, eventdata, handles)
global transicion transicionVentana
transicion=24
transicionVentana=24;
video;
```

---------- **CÓDIGO DE ALGORITMO DE EVALUACIÓN** ----------

```
function varargout = video(varargin)
% Se inicializa la primera función del programa, no editar
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @video_OpeningFcn, ...
                   'gui_OutputFcn',  @video_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% Fin de la función


% Esta función se ejecuta momentos antes de que la ventana sea visible
function video_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

global transicionVentana
% Dependiendo de la ventana anterior, manda la acción de cerrar.
if(transicionVentana==1)
close(FigA);
elseif(transicionVentana==2)
close(FigB);
elseif(transicionVentana==3)
close(FigC);
elseif(transicionVentana==4)
close(FigD);
elseif(transicionVentana==5)
close(FigF);
elseif(transicionVentana==6)
close(FigG);
elseif(transicionVentana==7)
close(FigH);
elseif(transicionVentana==8)
close(FigI);
elseif(transicionVentana==9)
close(FigK);
elseif(transicionVentana==10)
close(FigL);
elseif(transicionVentana==11)
close(FigM);
elseif(transicionVentana==12)
close(FigN);
elseif(transicionVentana==13)
close(FigO);
elseif(transicionVentana==14)
close(FigP);
elseif(transicionVentana==15)
close(FigQ);
elseif(transicionVentana==16)
close(FigR);
elseif(transicionVentana==17)
close(FigT);
elseif(transicionVentana==18)
```

```
close(FigU);
elseif(transicionVentana==19)
close(FigV);
elseif(transicionVentana==20)
close(FigW);
elseif(transicionVentana==21)
close(FigX);
elseif(transicionVentana==22)
close(FigY);
elseif(transicionVentana==23)
close(FigZ);
elseif(transicionVentana==24)
close(FigLunes);
elseif(transicionVentana==25)
close(FigMartes);
elseif(transicionVentana==26)
close(FigMiercoles);
elseif(transicionVentana==27)
close(FigJueves);
elseif(transicionVentana==28)
close(FigViernes);
elseif(transicionVentana==29)
close(FigDomingo);
elseif(transicionVentana==30)
close(FigUno);
elseif(transicionVentana==31)
close(FigDos);
elseif(transicionVentana==31)
close(FigTres);
elseif(transicionVentana==33)
close(FigCuatro);
elseif(transicionVentana==34)
close(FigCinco);
elseif(transicionVentana==35)
close(FigSeis);
elseif(transicionVentana==36)
close(FigSiete);
elseif(transicionVentana==37)
close(FigOcho);
elseif(transicionVentana==38)
close(FigOnce);
elseif(transicionVentana==39)
close(FigDoce);
elseif(transicionVentana==40)
close(FigTrece);
elseif(transicionVentana==41)
close(FigCatorce);
elseif(transicionVentana==42)
close(FigDespues);
elseif(transicionVentana==43)
close(FigHora);
elseif(transicionVentana==44)
close(FigJamas);
elseif(transicionVentana==45)
close(FigNoches);
elseif(transicionVentana==46)
close(FigNunca);
elseif(transicionVentana==47)
close(FigCuando);
elseif(transicionVentana==48)
close(FigQue);
elseif(transicionVentana==49)
close(FigAceptar);
elseif(transicionVentana==50)
close(FigQuizas);
elseif(transicionVentana==51)
close(FigBuen);
elseif(transicionVentana==52)
close(FigNoche);
```

```
elseif(transicionVentana==53)
close(FigMal);
elseif(transicionVentana==54)
close(FigNombre);
elseif(transicionVentana==55)
close(FigGusto);
elseif(transicionVentana==56)
close(FigDormir);
elseif(transicionVentana==57)
close(FigNecesitar);
elseif(transicionVentana==58)
close(FigPagar);
elseif(transicionVentana==59)
close(FigChile);
elseif(transicionVentana==60)
close(FigDulce);
elseif(transicionVentana==61)
close(FigVestido);
elseif(transicionVentana==62)
close(FigBrasier);
elseif(transicionVentana==63)
close(FigRopaInterior);
elseif(transicionVentana==64)
close(FigRopa);
elseif(transicionVentana==65)
close(FigBano);
elseif(transicionVentana==66)
close(FigCama);
elseif(transicionVentana==67)
close(FigCorazon);
elseif(transicionVentana==68)
close(FigAvion);
elseif(transicionVentana==69)
close(FigPsicologo);
elseif(transicionVentana==70)
close(FigOperacion);

end

% Las salidas de esta función van directo a la línea de comandos
function varargout = video_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
global bandera1 promedio1 promedio2 promedio3 videoA valorS1 valorS2 valorS3 valorS4 transicion
transicionVentana Letras
bandera1=0;
sens1=promedio1;
sens2=promedio2;
sens3=promedio3;
videoA=videoinput('winvideo',1,'MJPG_640x480');
% Se configura las opciones de adquision de video

set(videoA, 'FramesPerTrigger', Inf)
set(videoA, 'ReturnedColorspace', 'rgb');
videoA.FrameGrabInterval =11;
%con start(vid) se activa la adquisicion, pero todavia no se toma la primera foto
start(videoA)
contadorFlush=0;
while(bandera1==0)
% Se captura la imagen y se procesa para acondicionarla
data = getsnapshot(videoA);
data1=imadjust(data,[valorS1 valorS2 valorS3;1 1 1],[valorS4 valorS4 valorS4;1 1 1]);
foto_R=data1(:,:,1);
foto_G=data1(:,:,2);
foto_B=data1(:,:,3);
h2=fspecial('average',[5,5]);
media2R=imfilter(foto_R,h2);
media2G=imfilter(foto_G,h2);
media2B=imfilter(foto_B,h2);
foto2(:,:,1)=media2R;
```

```
foto2(:,:,2)=media2G;
foto2(:,:,3)=media2B;
diff_im = rgb2gray(foto2);
% Se defina la tolerancia y se realiza el proceso de binarizado
tol=50;
for i=1:480
   for j=1:640
      if((media2R(i,j))<=(sens1+tol)&&(media2R(i,j))>=(sens1-
tol)&&(media2G(i,j))<=(sens2+tol)&&(media2G(i,j))>=(sens2-
tol)&&(media2B(i,j))<=(sens3+tol)&&(media2B(i,j))>=(sens3-tol))
          diff_im(i,j)=255;
      else
         diff_im(i,j)=0;
      end
   end
end
diff_im = im2bw(diff_im);
% Se realiza el proceso de etiquetado
diff_im2 = bwareaopen(diff_im,2000);
diff_im1 = bwareaopen(diff_im,200);
BW_filled = imfill(diff_im1,'holes');
BW_filled1 = imfill(diff_im2,'holes');
[bw2,num2]= bwlabel(BW_filled1);
[bw,num]= bwlabel(BW_filled);
stats = regionprops(bw, 'BoundingBox', 'Centroid');
stats2 = regionprops(bw2, 'BoundingBox', 'Centroid');

data1=fliplr(data);
imshow(data1);
hold on
centroide=[0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0];
angulo=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
centroMan=[0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0];
% Se marcan con una "x" los cuerpos encontrados
for k=1:num
bc = stats(k).Centroid;
plot(abs(bc(1)-640),bc(2),'-m+')
centroide(k,1)=bc(1);
centroide(k,2)=bc(2);
end
for p=1:num2
bd = stats2(p).Centroid;
plot(abs(bd(1)-640),bd(2), '-m+')
centroMan(p,1)=bd(1);
centroMan(p,2)=bd(2);
end
% Se trazan las líneas y se obtiene el ángulo de las mismas
for f=1:num2
    for g=1:num
        distance=((centroide(g,1)-centroMan(f,1))^2+(centroide(g,2)-centroMan(f,2))^2)^0.5;
        if((30<distance) && (distance<200))
         x1=[centroide(g,1) centroMan(f,1)];
         y1=[centroide(g,2) centroMan(f,2)];
         line(abs(x1-640),y1,'LineWidth',2,'Color',[.2 .8 .8]);
         m=(centroide(g,2)-centroMan(f,2))/(centroide(g,1)-centroMan(f,1));
         yy=sign(centroide(g,2)-centroMan(f,2));
         xx=sign(centroide(g,1)-centroMan(f,1));
         if( xx == 1 && yy == -1)
              angulo(g)=-atan(m)*(180/3.1416)
         elseif( xx == -1 && yy == -1)
              angulo(g)=180-atan(m)*(180/3.1416)
         elseif( xx == -1 && yy == +1)
              angulo(g)=180-atan(m)*(180/3.1416)
         elseif( xx == 1 && yy == 1)
              angulo(g)=360-atan(m)*(180/3.1416)
         end
        end
    end
end
```

```
% Se evalúa y se compara con el error definido para cada seña
if(norm(Letras(transicion,1:15)-angulo)<=Letras(transicion,16))
bandera1=1;
stop(videoA);
flushdata(videoA);
Felicidades;
end
hold off
contadorFlush=contadorFlush+1;
% Se reinicia la variable de video cada 40 ciclos para que no se sature
if(contadorFlush==40)
    flushdata(videoA);
    contadorFlush=0;
end
end

stop(videoA);
flushdata(videoA);

% Esta función se ejecuta cuando se presiona el botón detener
function pushbutton1_Callback(hObject, eventdata, handles)
global bandera1 videoA
bandera1=1;
stop(videoA);
flushdata(videoA);


% Esta función se ejecuta cuando se presiona el botón "Volver a la calibración"
function pushbutton3_Callback(hObject, eventdata, handles)
global bandera1 videoA
bandera1=1;
stop(videoA);
flushdata(videoA);
captura;
```

---------- **CÓDIGO DE TRAYECTORIAS** ----------

```
function varargout = Trayectoria(varargin)
% Inicialización de la primer función, no debe editarse
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Trayectoria_OpeningFcn, ...
                   'gui_OutputFcn',  @Trayectoria_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% Fin de la primer función

% Esta función se ejecuta antes de que la ventana sea visible.
function Trayectoria_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

% Las salidas de esta función van directo a la línea de comandos
function varargout = Trayectoria_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
% Se declaran las variables globales
global bandera1 promedio1 promedio2 promedio3 video valor valorS1 valorS2 valorS3 valorS4
bandera1=0;
sens1=promedio1;
sens2=promedio2;
sens3=promedio3;
centroMan=zeros(2,2);
video=videoinput('winvideo',1,'MJPG_320x240');
% Se configura las opciones de adquision de video
set(video, 'FramesPerTrigger', Inf);
set(video, 'ReturnedColorspace', 'rgb');
video.FrameGrabInterval =8;
% Con start(vid) se activa la adquisicion, pero todavia se toma la primera foto
start(video)
contador=0;
ii=2;
% El arreglo "trayectorias" contiene los puntos de la trayectoria a cumplir
trayectorias=[145,95;149,95;153,96;157,97;161,99;165,102;169,107;172,112;174,115;174,117;174,121;17
5,125;174,129;174,133;171,139;167,145;163,149;159,151;155,153;151,153;147,154;143,154;139,154;135,1
53;131,151;127,149;123,145;119,139;117,134;116,129;115,125];
while(contador<300)
data = getsnapshot(video);
data1=imadjust(data,[valorS1 valorS2 valorS3;1 1 1],[valorS4 valorS4 valorS4;1 1 1]);

% de la imagen en escala de grises de la imagen adquirida en data
foto_R=data1(:,:,1);
foto_G=data1(:,:,2);
foto_B=data1(:,:,3);
h2=fspecial('average',[5,5]);
media2R=imfilter(foto_R,h2);
media2G=imfilter(foto_G,h2);
media2B=imfilter(foto_B,h2);
foto2(:,:,1)=media2R;
foto2(:,:,2)=media2G;
foto2(:,:,3)=media2B;
diff_im = rgb2gray(foto2);
```

```
% El proceso de binarizado y etiquetado es el mismo que en la ventana de evaluación
tol=50;
for i=1:240
   for j=1:320
      if((media2R(i,j))<=(sens1+tol)&&(media2R(i,j))>=(sens1-
tol)&&(media2G(i,j))<=(sens2+tol)&&(media2G(i,j))>=(sens2-
tol)&&(media2B(i,j))<=(sens3+tol)&&(media2B(i,j))>=(sens3-tol))
          diff_im(i,j)=255;
      else
         diff_im(i,j)=0;
      end
   end
end
diff_im = im2bw(diff_im);
diff_im2 = bwareaopen(diff_im,1000);
[bw2,num2]= bwlabel(diff_im2);
stats2 = regionprops(bw2, 'BoundingBox', 'Centroid');
data1=fliplr(data);
imshow(data1);

hold on

if(num2>0)
bd = stats2(1).Centroid;
plot(abs(bd(1)-320),bd(2),'-g+','LineWidth',2)

end
% Se traza la figura que debe encerrar a la cabeza
rectangle('Position',[135 30 50 70],'Curvature',1,'LineWidth',1.5)
plot(abs(trayectorias(1,1)-320),trayectorias(1,2),'-o','LineWidth',2,'MarkerSize',5,
'MarkerEdgeColor','r','MarkerFaceColor',[0,0,0])

for k=ii:30
plot(abs(trayectorias(k,1)-320),trayectorias(k,2),'-o','LineWidth',2,'MarkerSize',3,
'MarkerEdgeColor','b')
hold on
end

if(num2>0)
distancia=((bd(1)-trayectorias(ii,1))^2+(bd(2)-trayectorias(ii,2))^2)^0.5;

if(distancia<=20)
    ii=ii+1;
end
% Cuando se alcanzan todos los puntos de la trayectoria se depliegan ventanas y se resetean
variables
if(ii==30)
    contador=300;
    objetivo=1;
    FelicidadesT;
    stop(video);
    flushdata(video);
end
end

centroMan(:,:);
contador=contador+1;
hold off
end
if(objetivo==0)
Intenta_de_nuevo;
end
stop(video);
flushdata(video);

%Esta función se ejecuta si se presiona el botón "detener"
function Detener_Callback(hObject, eventdata, handles)
global bandera1 video
bandera1=1;
```

```
stop(video)
flushdata(video);
```
----------**CÓDIGO DE VENTANA DE FELICITACIÓN**----------

```
function varargout = Felicidades(varargin)
% Se inicializa la primera función
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Felicidades_OpeningFcn, ...
                   'gui_OutputFcn',  @Felicidades_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% Fin de la primera función


% Esta función se ejecuta justo antes de que la ventana sea visible
function Felicidades_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);


% Las salidas de esta función van directamente a la línea de comandos
function varargout = Felicidades_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
unam1=imread('like.jpg');
imshow(unam1)



% Esta función se ejecuta cuando se presiona el botón "continuar"
function pushbutton1_Callback(hObject, eventdata, handles)
global transicionVentana
%Dependiendo de la seña en cuestión se abre una nueva ventana(s).
if(transicionVentana==1)
FigB;
elseif(transicionVentana==2)
FigC;
elseif(transicionVentana==3)
FigD;
elseif(transicionVentana==4)
FigF;
elseif(transicionVentana==5)
FigG;
elseif(transicionVentana==6)
FigH;
elseif(transicionVentana==7)
FigI;
elseif(transicionVentana==8)
FigK;
elseif(transicionVentana==9)
FigL;
elseif(transicionVentana==10)
FigM;
elseif(transicionVentana==11)
FigN;
elseif(transicionVentana==12)
FigO;
elseif(transicionVentana==13)
FigP;
```

```
elseif(transicionVentana==14)
FigQ;
elseif(transicionVentana==15)
FigR;
elseif(transicionVentana==16)
FigU;
elseif(transicionVentana==18)
FigV;
elseif(transicionVentana==19)
FigW;
elseif(transicionVentana==20)
FigX;
elseif(transicionVentana==21)
FigY;
elseif(transicionVentana==22)
FigZ;
elseif(transicionVentana==23)
close Felicidades;
close video;
elseif(transicionVentana==24)
close Felicidades;
close video;
Trayectoria;
elseif(transicionVentana==25)
close Felicidades;
close video;
Trayectoria;
elseif(transicionVentana==26)
close Felicidades;
close video;
Trayectoria;
elseif(transicionVentana==27)
close Felicidades;
close video;
Trayectoria;
elseif(transicionVentana==28)
close Felicidades;
close video;
Trayectoria;
elseif(transicionVentana==29)
close Felicidades;
close video;
Trayectoria;
else
close Felicidades;
close video;
end


% Esta función se ejecuta cuando se presiona el botón “cerrar”
function pushbutton2_Callback(hObject, eventdata, handles)
%Se cierran las ventanas abiertas en ese momento
close Felicidades;
close video;
```

BIBLIOGRAFÍA

[1] Organización Mundial de la Salud (OMS), «Organización Mundial de la Salud, página oficial,» [En línea]. Available: http://www.who.int/topics/disabilities/es/. [Último acceso: 8 Agosto 2016].

[2] Comisión Nacional de Fomento Educativo (CONAFE), Discapacidad auditiva, guía didáctica para la inclusión en educación inicial y básica., Cd. de México: SEP, 2010.

[3] Instituto Nacional de Estadística y Geografía (INEGI), «INEGI, sitio web versión Beta, censo de población y vivienda 2010,» 2010. [En línea]. Available: http://www.beta.inegi.org.mx/proyectos/ccpv/2010/. [Último acceso: 8 Agosto 2016].

[4] DGDC, UNAM, «Manos que hablan,» *UNAMirada a la Ciencia,* p. 1, 2016.

[5] YouTube, «Resultados de búsqueda,» [En línea]. Available: https://www.youtube.com/results?search_query=lenguaje+de+se%C3%B1as. [Último acceso: 12 08 2016].

[6] INEGI , «Instituto Nacional de Geografía y Estadística, Indicadores sobre sociedad de la información, 2013 a 2015,» [En línea]. Available: http://www3.inegi.org.mx/sistemas/temas/default.aspx?s=est&c=19007. [Último acceso: 12 8 2016].

[7] El Universal, «El Universal, TECHBIT,» 7 7 2015. [En línea]. Available: http://www.eluniversal.com.mx/articulo/techbit/2015/07/7/ipn-desarrolla-un-guante-traductor. [Último acceso: 12 8 2016].

[8] T. S. Andrea Núñez, «TICbeat, SignAloud, guantes que traducen el lenguaje de signos a palabras,» 25 4 2016. [En línea]. Available: http://www.ticbeat.com/tecnologias/signaloud-guantes-que-traducen-el-lenguaje-de-signos-a-palabras/. [Último acceso: 13 8 2016].

[9] «TabletZona, Desarrollan una funda para tablets capaz de entender y traducir el lenguaje de signos,» 23 10 2014. [En línea]. Available: http://tabletzona.es/2014/10/23/desarrollan-una-funda-para-tablets-capaz-de-entender-y-traducir-el-lenguaje-de-signos/. [Último acceso: 13 8 2016].

[10] «Leap Motion,» [En línea]. Available: https://www.leapmotion.com/. [Último acceso: 14 8 2016].

[11] «Muy Interesante, Kinect se convierte en un traductor del lenguaje de signos,» [En línea]. Available: http://www.muyinteresante.es/innovacion/articulo/sordomudos-videoconsolas-integracion-kinect-se-convierte-en-un-traductor-del-lenguaje-de-signos. [Último acceso: 14 8 2016].

[12] «Buen Diario, Nuevo traductor en Internet: español-lenguaje de señas.,» 15 8 2013. [En línea]. Available: http://www.buendiario.com/nuevo-traductor-en-internet-espanol-lenguaje-de-senas/. [Último acceso: 15 8 2016].

[13] «Infogr.am, WINDOWS VS LINUX VS APPLE,» [En línea]. Available: https://infogr.am/Windows-vs-Linux-vs-Apple-limhoff_1359931857. [Último acceso: 15 8 2016].

[14] Forbes, «Forbes, México; Android e iOS dominarán el mercado (casi) para siempre,» 5 9 2013. [En línea]. Available: http://www.forbes.com.mx/android-e-ios-dominaran-el-mercado-casi-para-siempre/#gs.CrGZY=k. [Último acceso: 15 8 2016].

[15] Universiada de la República, Introducción a la Ingeniería de Software, Instituto de la computación, 2002.

[16] W. M. SABOGAL, «Lo que hay que saber del Español,» *El País,* 27 11 2010.

[17] Duolingo, «Duolingo: aprende Inglés, Francés y otros idiomas gratis,» [En línea]. Available: https://www.duolingo.com/. [Último acceso: 17 8 2016].

[18] Microsoft, «Microsoft, Tienda.,» [En línea]. Available: https://www.microsoft.com/es-mx/store/p/duolingo-aprende-idiomas-gratis/9wzdncrcv5xn. [Último acceso: 18 8 2016].

[19] L. G. Brown, « A survey of image registration techniques,» *ACM Computing Surveys,* vol. 24, nº 4, p. 325, 1992.

[20] National Instruments, «National Instruments, Tienda,» [En línea]. Available: http://www.ni.com/vision/software/vbai/. [Último acceso: 18 8 2016].

[21] OpenCV, [En línea]. Available: http://opencv.org/. [Último acceso: 18 8 2016].

[22] Matlab, «MathWork, documentation,» [En línea]. Available: http://www.mathworks.com/help/images/ref/imfilter.html. [Último acceso: 22 8 2016].

[23] Buy T Shirts Online, «The perception of the color,» [En línea]. Available: http://www.buytshirtsonline.co.uk/colour-perception/. [Último acceso: 24 8 2016].

[24] IntoRobotics, 26 11 2013. [En línea]. Available: https://www.intorobotics.com/9-opencv-tutorials-hand-gesture-detection-recognition/. [Último acceso: 15 8 2016].

[25] Matlab, «MathWorks,» [En línea]. Available: https://www.mathworks.com/help/images/ref/imadjust.html. [Último acceso: 27 08 2016].