Sistema para control de brazo robótico y reconocimiento autónomo de objetos

Ronaldo Canizales, Boris Franco, Alejandra Molina, Rodolfo Monge

Departamento de Electrónica e Informática, Universidad Centroamericana José Simeón Cañas, Ant. Cuscatlán, El Salvador

00001513@uca.edu.sv
00015413@uca.edu.sv
00099713@uca.edu.sv
00092195@uca.edu.sv

***Abstract*—El presente artículo describe el desarrollo de un nuevo sistema de software para el control de un brazo robótico propiedad del Departamento de Electrónica e Informática (DEI) de la Universidad Centroamericana José Simeón Cañas (UCA). Sus funcionalidades se resumen en los siguientes módulos: imitación de movimientos del usuario en tiempo real, grabación y reproducción de secuencias de movimientos y el reconocimiento autónomo de objetos. Para esto, se utilizaron los sensores ópticos de realidad aumentada conocidos como Kinect for Windows V1 y Leap Motion, el primero, encargado de la detección de los ángulos y coordenadas del antebrazo, brazo y bíceps; mientras que el segundo detecta los movimientos de las manos del usuario. Para la detección de objetos se emplearon algoritmos de visión por computadora y de minería de datos. Además, el software desarrollado es capaz de realizar mediciones y cálculos para ambos brazos del usuario. El sistema garantiza la compatibilidad si en un futuro se elabora la extremidad faltante.**

Si bien el sistema es compatible con el brazo original, se diseñó y trabajó con un tablero indicador que emula ambas extremidades.

Para comprobar el funcionamiento del sistema, este fue sometido a una serie de pruebas en las que se efectuaron mediciones, alterando diferentes variables, por ejemplo, la distancia de los objetos, la intensidad de la luz y la interferencia de otros objetos dentro del campo visual de los sensores.

Palabras claves—brazo robótico, Kinect, Leap Motion, realidad aumentada, visión por computadora.

I. INTRODUCCIÓN

El brazo robótico del Departamento de Electrónica e Informática de la UCA consiste en una extremidad superior derecha controlada por medio del reconocimiento de gestos. Sus piezas se obtuvieron a través de un proceso de impresión 3D y su estructura está basada en el proyecto de código abierto InMoov, diseñado en Francia por Gaël Langevin.

El dispositivo está formado por cinco partes: mano y muñeca, antebrazo, bíceps, hombro, pecho y espalda [1]; como se muestra en la Fig. 1.

Para el control de los movimientos de cada parte, el brazo robótico emplea la tecnología Arduino, que consiste en una plataforma electrónica integrada por una serie de placas microcontroladoras, un lenguaje de programación y un entorno de desarrollo de software [2].

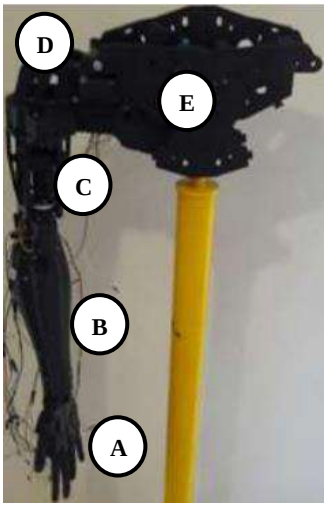

Fig. 1. Brazo robótico del DEI y sus partes, a) mano y b) antebrazo, c) bíceps, d) hombro, e) pecho y espalda. Adaptado de [1].

El brazo robótico utiliza una placa de tipo Arduino UNO para controlar los servomotores encargados de los movimientos del hombro y del bíceps. Mientras que para operar los servomotores encargados de la flexión y extensión de los dedos de la mano se utiliza una placa Arduino Mega 2560.

Originalmente, el software del brazo robótico se limitaba a la realización de una única tarea: la imitación de movimientos en tiempo real. No obstante, mediante el desarrollo e implementación de una nueva plataforma de escritorio, se ha logrado incrementar el número de funcionalidades del sistema que controla al dispositivo, estas son: el reconocimiento autónomo de objetos y la grabación y posterior reproducción de secuencias de movimientos.

La realidad aumentada es un término aplicado a todas aquellas tecnologías capaces de superponer o combinar información alfanumérica, simbólica y gráfica con la visión que una persona tiene sobre el mundo real [3]. En la actualidad existen en el mercado, numerosos dispositivos que la implementan, ejemplo de ello son los sensores ópticos Kinect for Windows V1 y Leap Motion. Ambos son utilizados dentro del sistema de control del brazo robótico, específicamente para la detección y grabación de los movimientos de las extremidades superiores del usuario. El primero de ellos, se encarga de obtener las coordenadas de nueve articulaciones de la parte superior del cuerpo: muñecas, codos, hombros, pecho, cuello y cabeza. Mientras que el segundo sensor se ocupa de detectar los

movimientos de ambas manos del usuario y de sus correspondientes dedos.

Dentro del sistema se desarrolló un módulo que permite el reconocimiento autónomo de objetos, es decir, que en base a una imagen capturada por el sensor Kinect, el software es capaz de clasificar y distinguir una serie de objetos respecto de su entorno. Para lograrlo, se requirió del uso de técnicas pertenecientes a una disciplina conocida como "visión por computadora", que tiene por objetivo el análisis de imágenes capturadas por medio de una cámara, con la finalidad de obtener información de los objetos presentes en esa escena [4]. Además que pretende desarrollar sistemas autónomos capaces de realizar (y en muchos casos superar) las funciones correspondientes a la visión humana. La implementación de algoritmos de visión por computadora se realizó a través de la librería OpenCV.

Las imágenes digitales son el elemento central dentro de la visión por computadora. Estas se definen como una matriz conformada por una serie de elementos denominados píxeles. Cada uno de ellos contiene el nivel de iluminación o el color de un punto en la escena [5]. De esta forma, una imagen con una resolución de 35 x 35 es una matriz que contiene 35 columnas y 35 filas, dando un total de 1255 elementos o píxeles, como puede apreciarse en la Fig. 2.

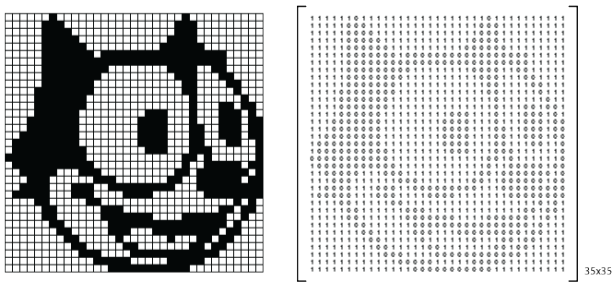

Fig. 2. Representación de una imagen digital mediante una matriz de 35x35. Adaptado de [6].

Para el reconocimiento de objetos, el sistema también hace uso de herramientas matemáticas de minería de datos que permiten predecir a qué clase pertenecen cada uno de los objetos identificados dentro de una imagen.

Dentro de la minería de datos, se denomina clasificación al proceso encargado de encontrar un modelo (o función matemática) que describe y distingue distintas clases de datos, con el propósito de ser capaz de usar dicho modelo para predecir la clase de objetos cuya etiqueta de clase es desconocida. El modelo resultante está basado en el análisis del conjunto de datos de entrenamiento, es decir, datos cuya etiqueta de clase es conocida [7].

El modelo resultante puede ser representado en varias formas, por ejemplo, mediante reglas de clasificación o a través de árboles de decisión, un ejemplo de este último, generado a través de una herramienta estadística conocida como Rattle perteneciente al lenguaje de programación R, se muestran en la Fig. 3.

Contrario a los modelos de clasificación, que analizan las etiquetas de clases de objetos, dentro de la minería de datos existen las técnicas de agrupamiento o clustering, que analizan los datos del objeto sin consultar etiquetas de clase. En general, las etiquetas de clase no están presenten en los datos de entrenamiento, simplemente porque éstos no se conocen a priori.

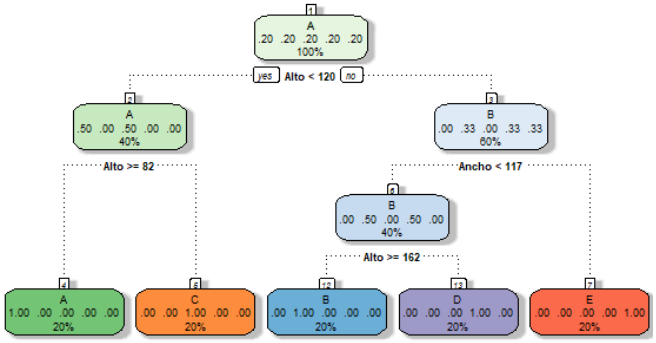

Fig. 3. Ejemplo de un árbol de decisión, generado con la herramienta Rattle del lenguaje R.

El clustering puede ser utilizado para generar dichas etiquetas de clase. Una técnica de agrupamiento muy conocida y comúnmente usada es el algoritmo de k-means, conocida también como clustering basado en centroides.

Cuando se conoce desde un inicio la etiqueta de clase de cada dato del conjunto de entrenamiento, entonces se dice que el proceso de aprendizaje del clasificador es supervisado (ya que se sabe a qué clase pertenece cada objeto). Lo contrario sucede con el aprendizaje no supervisado (o clustering) donde las etiquetas de clase son desconocidas y tampoco se sabe la cantidad de clases existentes.

El sistema descrito en este artículo, fue desarrollado utilizando a un tablero indicador compuesto por ledes cuya estructura interna está basada en el brazo robótico original. De este modo, se garantiza la compatibilidad para ambos dispositivos.

La comunicación entre el Leap Motion, Kinect y el brazo robótico o el tablero indicador se lleva a cabo utilizando la tecnología Bluetooth, que es un estándar de comunicación inalámbrico empleado para la interconexión de dispositivos electrónicos.

Referente al presente artículo, ser realiza una descripción del diseño de cada uno de los componentes que integran al sistema de control del brazo robótico del DEI, detallando el funcionamiento tanto de su software como de su hardware. También, se presentan los resultados de algunas de las pruebas que se efectuaron para comprobar el funcionamiento del sistema. Finalmente se incluyen las conclusiones resultantes del desarrollo de proyecto.

II. MATERIALES Y MÉTODOS

A. Materiales

1) El sensor Kinect

Es un dispositivo desarrollado por Microsoft y utilizado para la detección de movimientos y el reconocimiento de gestos. Permite que una persona interactúe con el sistema o las aplicaciones por medio del reconocimiento movimientos del cuerpo y comandos de voz; eliminando la necesidad de utilizar dispositivos de entrada como un control o un teclado.

A través de los años se han lanzado múltiples versiones del Kinect. En este trabajo se utilizó un sensor Kinect for Windows Version 1 (V1). Su estructura se puede apreciar en la Fig. 4.

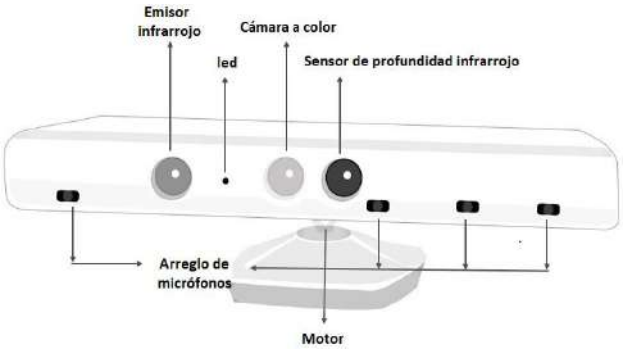

Fig. 4. Partes del sensor Kinect. Adaptado de [8].

El rango de detección en el que trabaja el sensor se encuentra entre 0.8 m y 4 m. El dispositivo es capaz de detectar objetos fuera de ese intervalo, pero no se garantiza que las coordenadas de las articulaciones de la persona estén en concordancia con los datos de la realidad.

Para programarlo, se requiere utilizar el Kinect for Windows SDK, que consiste en un conjunto de herramientas de software utilizadas para el desarrollo y ejecución de aplicaciones compatibles con el dispositivo. Contiene los controladores o drivers necesarios para interactuar con el sensor. Para el desarrollo del presente trabajo, se utilizó la versión 1.8 del SDK.

2) El sensor Leap Motion

Es un dispositivo utilizado para el reconocimiento de gestos desarrollado por la empresa Leap Motion Inc. Es capaz de detectar, con una precisión de 0.01 mm y sin una latencia visible, los movimientos de ambas manos y de sus correspondientes dedos, transmitiendo todos estos datos a una computadora mediante una conexión USB. La Fig 5 muestra la estructura externa del sensor Leap Motion.


Fig. 5. Estructura externa del sensor Leap Motion

En su interior, el sensor posee tres ledes y dos cámaras monocromáticas. Los ledes se encargan de iluminar la zona de interacción mediante la emisión de luz infrarroja con una longitud de onda de 850 nm. Las cámaras son las encargadas de capturar las imágenes de los objetos que reflejan la luz infrarroja emitida por los ledes para luego enviarlas a una computadora y que estas sean analizadas por el software de Leap Motion.

Para el desarrollo de aplicaciones, se requiere utilizar el Leap Motion SDK, que contiene las librerías y servicios necesarios para operar el sensor. Para el desarrollo del sistema que controla al brazo robot, se utilizó el Leap Motion V2 Software.

3) Librería OpenCV

Es una librería de software utilizada dentro de las áreas de visión por computadora y de aprendizaje de máquina. Nativamente está desarrollada en el lenguaje de programación C++, sin embargo posee implementaciones para C, Python, Java y MATLAB y se encuentra disponible para los sistemas operativos Windows, Mac OS, GNU/Linux y Android. Para el sistema, se trabajó con la versión 3.4.1, específicamente su implementación para C++.

Algunas de las actividades en las que la librería OpenCV ha sido implementada incluyen: detección de intrusos en labores de videovigilancia, ayuda a robots en la navegación y recolección de objetos en almacenes.

4) Tablero indicador

El tablero indicador está conformado internamente por cuatro placas Arduino, su distribución se encuentra ilustrada en la Fig. 6.

Fig. 6. Interacción entre el sistema y tablero indicador.

Las placas Arduino Mega tienen anexada una pantalla led donde se visualizan cada uno de los ángulos designados a dicha placa (Fig. 7), es decir, la placa Arduino que reciba el paquete de datos del Leap Motion mostrará los ángulos de abertura de los dedos mientras que la placa que reciba el paquete de datos del sensor Kinect mostrará los ángulos de abertura de los brazos.

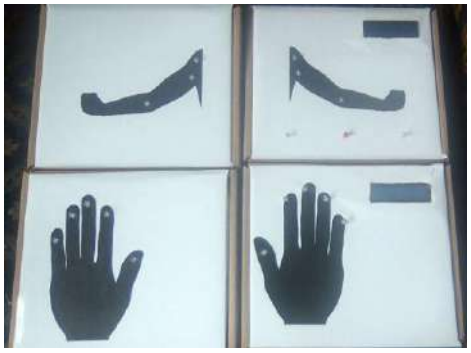
Fig. 7. Tablero indicador

Cada placa Arduino contiene un software que sigue la estructura base que se muestra en la Fig. 8. Dicha estructura está diseñada para ser compatible tanto con el tablero indicador como con la extremidad superior original.

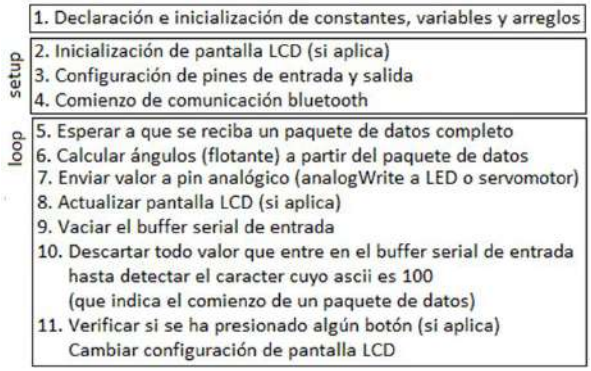

Fig. 8. Estructura base del software de las placas Arduino.

5) Plataforma de escritorio

La plataforma de escritorio es el componente central, a través de ella fluye toda la información generada por el sistema. Se trata de una aplicación de escritorio desarrollada en C#.

Los ángulos y coordenadas de distintos puntos de las extremidades superiores del usuario, brindados por ambos sensores son registrados en archivos de valores separados por comas. Estos son generados automáticamente por la plataforma.

6) Figuras utilizadas para módulo de clasificación

Cinco figuras fueron elegidas para realizar la detección por color, estas son: cuadro grande, cuadro pequeño, cruz, triángulo y círculo; mostradas en la Fig. 9

Fig. 9. Figuras a clasificar en base a su color.

Para la clasificación de objetos en base a su distancia, se eligieron tres siluetas de cartón correspondientes a un rectángulo, una cruz y un triángulo; además de una botella opaca y un peluche esférico. Todos estos objetos se muestran en la Fig. 10.

Fig. 10. Figuras utilizadas para la clasificación en base a distancia.

B. Métodos

1) Imitación de movimientos en tiempo real

El primer proceso realizado por el sistema es la imitación en tiempo real de los movimientos realizados por el usuario, cuya estructura es mostrada en la Fig. 11.

Fig. 11. Proceso de imitación en tiempo real.

La imitación de los movimientos de las manos se hace de manera simultánea a la de los brazos pero aislada de ésta. Es decir que son dos procesos distintos que siguen la misma estructura y se ejecutan al mismo tiempo, pero de forma independiente uno del otro.

2) Cálculos realizados con datos provistos por los sensores

La extremidad robótica original y el tablero indicador necesitan dos tipos de ángulos para que puedan operar los servomotores y los ledes respectivamente. El primer tipo corresponde a los ángulos de abertura de cada dedo respecto a la palma de la mano. Mientras que el segundo hace referencia a los brazos, específicamente los ángulos de abertura horizontal y vertical de cada hombro y a los ángulos de abertura de cada codo.

Para calcular los ángulos relativos a la mano se utilizan los vectores *Hand.Direction* y *Finger.Direction*, provistos por el Leap Motion, que devuelven un vector tangente a la palma de la mano y un vector gradiente a la punta de cada dedo, es decir un vector posicionado en la punta de los dedos con la dirección hacia donde apuntan, respectivamente. Una representación visual de estos vectores se aprecia en la Fig. 12.

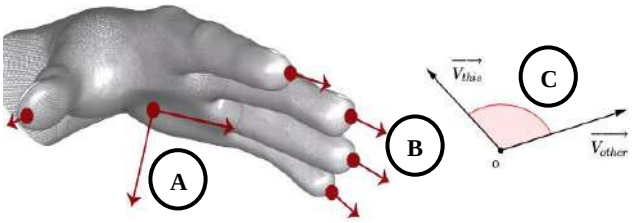

Fig. 12. a) Vector Hand.Direction, b) vectores Finger.Direction, c) ángulo devuelto por la operación Vector.AngleTo. Adapado de [1] y [9].

Se utiliza la operación *Vector.AngleTo* para calcular el ángulo de abertura de un vector respecto a otro. Dicho ángulo es devuelto en radianes, siempre es positivo y su magnitud es menor o igual a pi radianes o 180°. Los ángulos correspondientes a los dedos índice y meñique se limitan a [0°, 150°] mientras que los demás se limitan a [0°, 160°] esto con la finalidad de garantizar la compatibilidad con la extremidad robótica del DEI.

Para el cálculo de los ángulos relativos al brazo se utilizan las articulaciones humanas: muñecas, codos y hombros tanto izquierdos como derechos y un punto llamado *ShoulderCenter* que está ubicado ligeramente arriba del punto central de una línea imaginaria que une ambos hombros. Dichos puntos están resaltados de color verde en la Fig. 13.

Fig.13. Articulaciones reconocidas por el Kinect V1. Adaptado de [X].

En el cálculo de los ángulos relativos al brazo del usuario se utilizan tres ecuaciones fundamentales de la matemática. Para su visualización, se toma como base los triángulos de la Fig. 14.

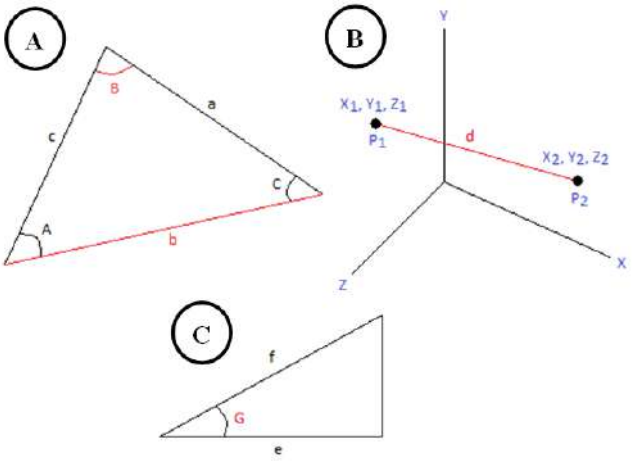

Fig. 14. a) Triángulo oblicuángulo, b) dos puntos en R3, c) triángulo rectángulo.

La primera fórmula es la ley de los cosenos (Ec. 1) que se utiliza para relacionar la longitud y los ángulos de cualquier triángulo, sea éste rectángulo o no. La segunda fórmula es la correspondiente a la distancia entre dos puntos cualesquiera en el espacio (Ec. 3) y finalmente el coseno (Ec. 4), es decir, la relación entre el cateto adyacente y la hipotenusa de un triángulo rectángulo.

$$b^2 = a^2 + b^2 - 2acCos(B) \quad \text{(Ec. 1)}$$

$$B = Cos^{-1}\left(\frac{a^2+c^2-b^2}{2ac}\right) \quad \text{(Ec. 2)}$$

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad \text{(Ec. 3)}$$

$$Cos(G) = \frac{e}{f} = \frac{cateto\ ayacente}{hipotenusa} \quad \text{(Ec. 4)}$$

Para el cálculo del ángulo de abertura de un codo (Ec.5) se toman las coordenadas de un hombro, un codo y una muñeca y los llamaremos A, B y C respectivamente. Con estos tres puntos se puede formar un triángulo oblicuángulo en el espacio, donde los vectores AB, BC y el ángulo B corresponden a la longitud del bíceps, antebrazo y al ángulo de abertura del codo respectivamente. Primero se usa (Ec. 3) para calcular dichas longitudes y finalmente (Ec. 2) para encontrar el ángulo entre ellas.

$$\theta_{AberturaCodo} = Cos^{-1}\left(\frac{d_{ab}^2+d_{bc}^2+d_{ac}^2}{2d_{ab}d_{bc}}\right)$$

$$d_{ab} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2}$$

$$d_{ab} = \sqrt{(x_c - x_b)^2 + (y_c - y_b)^2 + (z_c - z_b)^2}$$

$$d_{ac} = \sqrt{(x_c - x_a)^2 + (y_c - y_a)^2 + (z_c - z_a)^2} \quad \text{(Ec. 5)}$$

En el cálculo de los ángulos de abertura horizontal y vertical de los hombros se toman las coordenadas de un hombro central, hombro derecho y codo derecho, indicados con los puntos A, B y C respectivamente dentro de la Fig. 15.

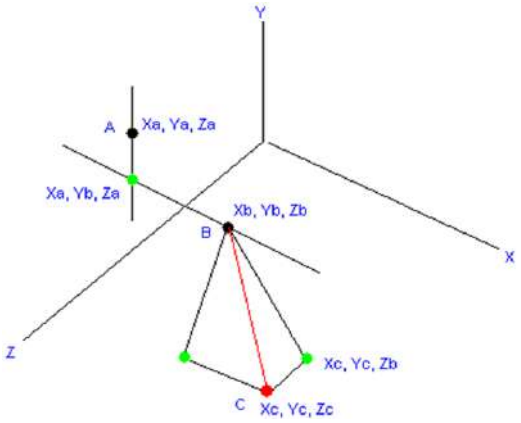

Fig. 15. Coordenadas de un hombro central, hombro derecho y codo derecho.

El punto resaltado en color verde que se encuentra más a la derecha en la Fig. 13 corresponde a una proyección del punto C en una traza del eje Z a la altura de la coordenada Z del punto B, en otras palabras, desplazar el codo respetando la altura original, y colocarlo en un punto donde no esté ni delante ni detrás del cuerpo. Esto con el fin de deshacernos de la abertura horizontal y quedarnos solamente con la vertical. Dicho punto se encuentra resaltado en color rojo en la Fig. 16. Cabe mencionar que al hacer esta proyección, se pasa del espacio XYZ al plano XY.

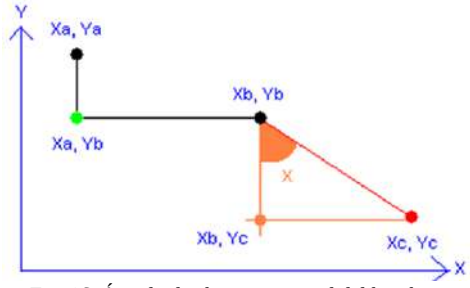

Fig. 16. Ángulo de abertura vertical del hombro.

El punto resaltado en color naranja en la Fig. 14 corresponde a aquella coordenada que forma un triángulo rectángulo cuya altura es igual (en valor absoluto) que la diferencia de alturas entre el codo y el hombro de la persona. El ángulo formado entre el cateto vertical y la hipotenusa representa la abertura vertical del hombro y puede ser calculado usando (Ec. 6) en dicho triángulo rectángulo.

$\theta_{AberturaVerticalHombro} = Cos^{-1}\left(\frac{d_{CatAdy}}{d_{Hipotenusa}}\right)$, donde:

$$d_{CatetoAdyacente} = |y_c - y_b|$$

$$d_{Hipotenusa} = \sqrt{(x_c - x_b)^2 + (y_c - y_b)^2} \quad \text{(Ec. 6)}$$

Para el cálculo el ángulo de abertura horizontal del hombro se proyectan los puntos A y C de la Fig. 17 en una traza del eje Y a la altura de la coordenada Y del punto B, es decir, sería como estar viendo a la persona desde arriba.

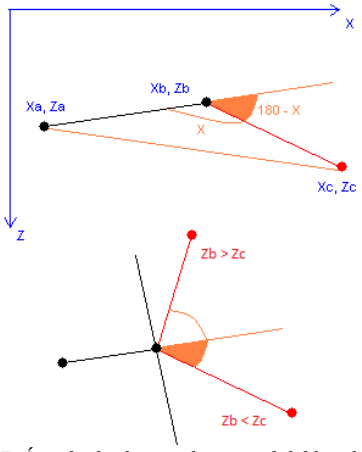

Fig. 17. Ángulo de abertura horizontal del hombro.

La magnitud del ángulo que interesa se encuentra resaltado de color naranja en la Figura X, correspondiente al ángulo suplementario (cuya suma da 180°) al ángulo A'BC' del triángulo proyectado, cuya expresión matemática corresponde a la Ec. 7. Cabe notar que el signo depende de las coordenadas Z de los puntos B y C, es decir, el signo depende de si el codo de la persona se encuentra más cerca o más lejos del sensor Kinect que su hombro.

$$\theta_{AbHorizontalHombro} = \pi - Cos^{-1}\left(\frac{d_{ab}^2+d_{bc}^2-d_{ac}^2}{2d_{ab}d_{bc}}\right), \text{ donde:}$$

$$d_{ab} = \sqrt{(x_b - x_a)^2 + (z_b - z_a)^2}$$

$$d_{ab} = \sqrt{(x_c - x_b)^2 + (z_c - z_b)^2}$$

$$d_{ac} = \sqrt{(x_c - x_a)^2 + (z_c - z_a)^2} \qquad \text{(Ec. 7)}$$

3) Ensamblaje de paquetes de datos que se envían al tablero indicador

La extremidad superior original maneja los ángulos como cifras enteras, tanto los de las manos como los de los brazos. En cambio, la nueva plataforma de escritorio es capaz de manejar los ángulos de los dedos hasta con seis cifras decimales y los ángulos de los brazos hasta con dos cifras decimales. El inconveniente se presenta al enviar dichas cantidades a las placas Arduino vía Bluetooth, pues el buffer serial sólo permite recibir datos enteros de un byte de longitud. Para solventarlo, se utiliza el proceso ilustrado en la Fig. 18, donde se convierte un flotante (cuyo valor sea mayor o igual a cero y menor a mil) a cinco bytes con un valor que en este caso oscila entre 0 y 99.

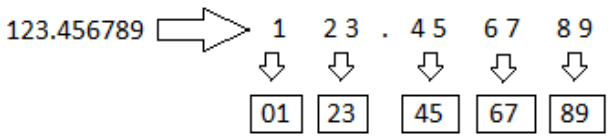

Fig. 18. Conversión de un flotante a un arreglo (conjunto) de bytes.

El objetivo es descomponer el número flotante en diez dígitos organizados en cinco parejas: la primera toma el valor de "01" si el ángulo alcanza el valor de la centena o "00" en caso contrario, mientras que las demás parejas representan (de izquierda a derecha) a las "decenas y unidades", "décimas y centésimas", "milésimas y diez milésimas" y "cien milésimas y millonésimas".

4) Grabación y reproducción de movimientos

La grabación de secuencias de movimientos se realiza en archivos de valores separados por coma (CSV) de forma que estos puedan reproducirse en el futuro. El proceso se esquematiza en la Fig. 19.

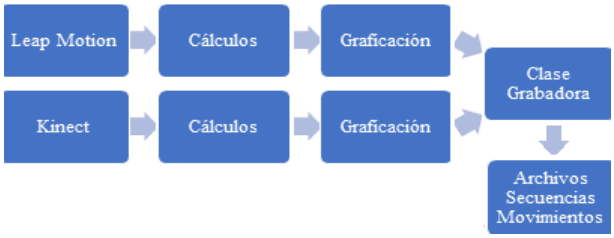

Fig. 19. Proceso general de grabación de secuencias de movimientos.

La grabación comparte ciertas similitudes con la imitación de movimientos en tiempo real. La diferencia radica en el tratamiento posterior que reciben los ángulos y coordenadas de las extremidades superiores de la persona, ya que en la grabación tanto los ángulos como las coordenadas se almacenan en archivos de valores separados por comas, mientras que para la imitación solamente los ángulos se envían al tablero indicador.

Los archivos de secuencias de movimientos están conformados por registros de 225 datos. La cantidad de registros depende de la duración de la grabación.

El proceso de reproducción puede dividirse en las cuatro etapas que se muestran en la Fig. 20.

Fig. 20. Proceso general de reproducción de movimientos pre-grabados.

Durante la primera etapa la plataforma de escritorio lee, interpreta y carga en memoria todos los datos contenidos en una secuencia pre-grabada. Luego, se ofrece al usuario la posibilidad de generar una animación a partir de dicha secuencia pre-grabada.

Una vez que el usuario ha visualizado la animación, es posible que esté conforme con ella o que desee remover algunos cuadros de la secuencia pre-grabada. Por esto, se ha implementado la etapa llamada depuración de secuencia pre-grabada, su uso es opcional. Finalmente la secuencia pre-grabada se encuentra lista para ser enviada al tablero indicador.

11) Reconocimiento autónomo de objetos

La cuarta funcionalidad que ofrece la plataforma de escritorio es el reconocimiento de objetos. Es decir, la diferenciación de estos respecto a su entorno. Siendo dicha tarea la que aporta el componente autónomo al sistema.

Este proceso (esquematizado en la Fig.21) puede realizarse bajo dos modalidades de detección: por color y por distancia.

Fig. 21. Esquema general del proceso de reconocimiento de objetos.

Cabe aclarar que la modalidad de detección por distancia se refiere a que el objeto se encuentre en el rango óptimo de detección del Kinect pero no así su fondo (los objetos que lo rodean).

El reconocimiento comienza cuando el sensor Kinect toma una fotografía de los objetos que se encuentran en escena. Después, dicha fotografía es procesada por un programa que extrae sus características (por ejemplo área, perímetro, entre otros) desarrollado en C++ utilizando las librerías OpenCV.

La plataforma de escritorio lee las características de cada objeto y las procesa con un algoritmo de clasificación. Dicho algoritmo utiliza herramientas matemáticas de minería de datos (k-means y árboles de decisión) para predecir la clase a la que pertenecen cada uno de los objetos.

III. RESULTADOS Y DISCUSIÓN

El nuevo sistema reproduce fielmente los movimientos del usuario, además que genera una visualización en tiempo real de sus extremidades superiores. Dicha visualización se ve enriquecida con el uso de realidad aumentada, es decir, para facilitar la lectura de valores calculados por el sistema, estos son dibujados nueve veces por segundo sobre cada imagen o fotograma capturado por las cámaras.

Sobre las imágenes provistas por el sensor Kinect se dibuja un total de veintitrés elementos: seis ángulos para cada brazo, nueve articulaciones del cuerpo y ocho segmentos de recta que unen a las articulaciones.

Simultáneamente el sistema realiza un proceso similar con las imágenes provistas por el sensor Leap Motion sobre las que se dibujan hasta un máximo de cuarenta y ocho elementos: cada uno de los diecinueve huesos de conforman cada mano y además, ligeramente arriba de la punta de cada dedo, se dibuja su respectivo ángulo de abertura respecto a la palma de la mano.

La Fig. 22 muestra la visualización de los elementos anteriormente descritos dentro de la interfaz gráfica de la plataforma de escritorio.

Se realizó un total de seis pruebas para comprobar el funcionamiento del módulo de imitación de movimientos en tiempo real dentro de distintos rangos de distancia. Como resultado, se comprobó que tanto el sensor Leap Motion como el Kinect son capaces de realizar las mediciones de forma exacta siempre y cuando el usuario no se aleje o acerque demasiado a las cámaras de los dispositivos. En base a las pruebas efectuadas, el rango óptimo de detección obtenidos para el Kinect fue de 0.7 m a 3.75 m, mientras que para el Leap Motion fue de 20 cm a 50 cm.

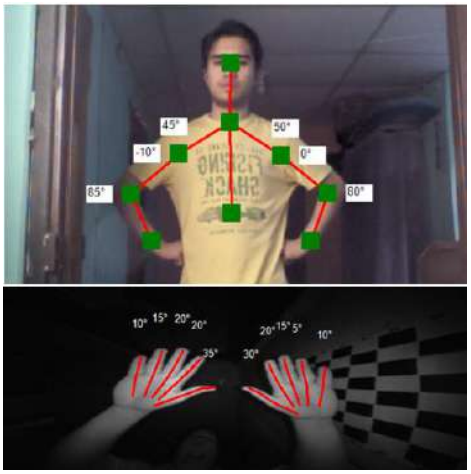

Fig. 22. Articulaciones y huesos de las manos interpretados por la plataforma de escritorio.

No se presentó alteración alguna en la detección de los movimientos al realizar pruebas con variaciones de luminosidad, tanto para exceso de luz (Fig. 23) como para su ausencia (Fig. 24).

Fig. 23. Iluminación excesiva frente a: a) sensor Kinect y b) sensor Leap Motion.

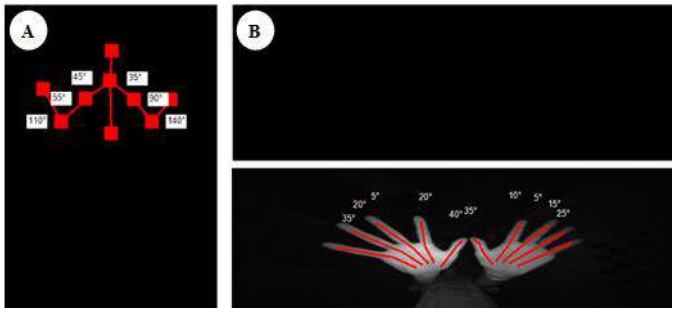

Fig. 24. Sensores operando bajo nula iluminación: a) Kinect, b) Leap Motion.

En el caso del módulo de grabación y reproducción de movimientos el sistema es capaz de generar una animación en base a la secuencia registrada dentro de los archivos de valores separados por coma.

Se efectuaron tres pruebas para observar el comportamiento de dichos archivos ante grabaciones de distintos intervalos de tiempo, los resultados obtenidos se aprecian en la TABLA 1.

TABLA 1. CARACTERÍSTICAS DE ARCHIVOS DE VALORES SEPARADOS POR COMA

Duración	Tamaño archivo	Cantidad registros	Registros Kinect	Registros Leap Motion
1 min	598 KB	525	234	291
5 min	3.09 MB	2919	1336	1583
10 min	5.26 MB	5313	2288	3025

Para todos los casos, el sistema fue capaz de generar los archivos y guardar la información de forma exitosa. De igual modo, la reproducción de las secuencias de movimientos se efectuó si ningún problema.

Por último, se realizaron experimentos con el módulo correspondiente al reconocimiento de objetos. Se llevó a cabo un total de 125 pruebas bajo circunstancias ideales: 35 para el modo de clasificación en base a distancia y 90 para la clasificación en base a color.

Para las pruebas del clasificador de figuras que trabaja con detección en base a distancia, el clasificador aparenta ser perfecto, ya que no se equivocó ni una vez. La TABLA 2 muestra los resultados. Cada fila representa las siete pruebas realizadas con un tipo de figura en específico, pero dentro de cada fila, los datos se colocan en la columna correspondiente al resultado que ofrece el clasificador.

TABLA 2. RESULTADOS DE CLASIFICADOR POR DISTANCIA

	Rectángulo	Cruz	Triángulo	Botella	Peluche
Rectángulo	7	0	0	0	0
Cruz	0	7	0	0	0
Triángulo	0	0	7	0	0
Botella	0	0	0	7	0
Peluche	0	0	0	0	7

Los resultados de las pruebas al clasificador de figuras que trabaja con detección en base a colores, pueden apreciarse en la TABLA 3

TABLA 3. RESULTADOS DE CLASIFICADOR POR COLOR

	Cuadro grande	Cruz	Triángulo	Elipse	Cuadro pequeño
Cuadro Grande	15	0	0	0	3
Cruz	0	18	0	0	0
Triangulo	0	0	18	0	0
Elipse	0	0	0	18	0
Cuadro Pequeño	0	0	0	1	17

En la primera fila pueden contabilizarse las 18 pruebas realizadas a la figura Cuadro grande, pero 3 de ellas están en la columna de la figura Cuadro pequeño mientras que todas las demás están en la columna correcta, es decir, la del Cuadro grande.

Esto significa que en tres de las 18 ocasiones que se le presentó un Cuadro grande al clasificador, éste arrojó una respuesta errónea, específicamente confundió la figura con un Cuadro pequeño.

IV. CONCLUSIONES

La imitación en tiempo real, grabación y reproducción de secuencias de movimientos tanto para brazos como para manos, se realizan de manera exitosa para ambas extremidades superiores.

El módulo de clasificación de figuras ofrece resultados fiables, ya que la exactitud de la detección en base al color es del 95% mientras que en base a la distancia es del 100%.

Mediante la utilización de cinco cifras decimales en el cálculo de los ángulos de las manos y dos cifras en el caso de los brazos se utiliza el 100% de la precisión de ambos tipos de servomotores. Mientras que anteriormente se utilizaban 90% y 9.7% de la precisión de los brazos y manos respectivamente.

El tiempo de retraso desde que el usuario realiza una acción hasta que esta se visualiza en el tablero indicador es de 109.5 ms aproximadamente.

La grabación de movimientos tiene un inmenso potencial en futuras aplicaciones, por ejemplo: corrección de la postura, traducción de lenguaje de señas, terapias de recuperación de la motricidad fina y gruesa de personas que han sufrido accidentes, entre otros.

La implementación de métodos numéricos o de redes de neuronas artificiales permitiría suavizar las secuencias de movimientos capturadas por los sensores, evitando así ordenarle al brazo que realice movimientos bruscos que podrían dañarlo

RECONOCIMIENTOS

Agradecemos a nuestro director de tesis, el Mgtr. Guillermo Cortés y también al Msc. Daniel Sosa, sin su valiosa ayuda el presente trabajo no hubiera sido posible.

REFERENCIAS

[1] FLAMENCO, F., MARROQUÍN, R., PORTILLO, J. y RIVAS, G. (2017). *Brazo robot controlado por gestos* (Tesis de pregrado). UCA, Antiguo Cuscatlán, El Salvador.

[2] RAMOS, E., CASTRO, C. y JAWORSKI, P. (2012). *Arduino and Kinect Projects*. New York: Apress.

[3] AUKSTAKALNIS, S. (2017). *Practical Augmented Reality: a Guide to the Technologies, Applications and human factors for AR and VR*. Estados Unidos: Pearson.

[4] SZELISKI, R. (2010). *Computer Vision: Algorithms and Applications*. Recuperado de http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

[5] ALEGRE, E., PAJARES, G., DE LA ESCALERA, A. (2016). *Conceptos y Métodos en Visión por Computador*. Recuperado de https://intranet.ceautomatica.es/sites/default/files/upload/8/files/ConceptosyMetodosenVxC.pdf

[6] PESCO, D. y BORTOLOSSI, H. (Sin fecha). *Matrices and Digital Images*. Recuperado del sitio de Internet de Fluminense Federal University, Institute of Mathematics and Statistics: http://dmuw.zum.de/images/6/6d/Matrix-en.pdf

[7] HAN, J. & KAMBER, M. (2006). Data Mining: Concepts and Techniques. San Francisco: Morgan Kaufmann.

[8] JANA, A. (2012). *Kinect for Windows SDK Programming* Guide. Birmingham: Packt Publishing.

[9] LEAP MOTION INC. (Sin fecha). API Overview. Recuperado de https://developer-archive.leapmotion.com/documentation/v2/csharp/